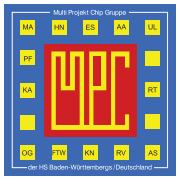
MPC

MULTI PROJEKT CHIP GRUPPE
BADEN-WÜRTTEMBERG

Herausgeber: Hochschule Ulm **Ausgabe:** 44 **ISSN** 1862-7102 **Workshop:** Reutlingen Juli 2010



Cooperating Organisation
Solid-State Circuit Society Chapter
IEEE German Section

**Tagungsband zum Workshop der Multiprojekt-Chip-Gruppe Baden-Württemberg,
Reutlingen, 9. Juli 2010**

Inhaltsverzeichnis

Constraint-driven Design - Eine Wegskizze zum analogen Designflow der nächsten Generation [1]

Jürgen Scheible

Robert Bosch GmbH, AE/EIL, Tübinger Str. 123, 72762 Reutlingen

07121 / 35-2763, juergen.scheible@de.bosch.com

Der Entwurf analoger integrierter Schaltkreise ist bis heute durch einen interaktiven Entwurfsstil gekennzeichnet. Das Backend dieses Prozesses bildet der Layoutentwurf, der üblicherweise mit der SDL-Methode (*schematic driven layout*) durchgeführt wird. Als Ziel wird i.a. – in Analogie zu den im Digitaldesign existierenden Lösungen - eine vollautomatische Layoutsynthese auch im Analogbereich angestrebt.

Die hier für das Analogdesign vorgeschlagene Methodik hat nicht eine solche Layoutsynthese zum Inhalt, sondern stellt einen realistischeren Zwischenschritt dar. Die Kernaussage besteht darin, dass zunächst eine Methode bereitzustellen ist, bei der alle die Schaltungsfunktion beeinflussenden Randbedingungen (c*onstraints*) rechnergestützt *prüfbar* sein müssen. Erst auf dieser Basis wird es gelingen, in einem weiteren Schritt, analoges Layout zu synthetisieren.

Ausgehend von einer grundsätzlichen Betrachtung zu Contraints, wird diese These aus einer Betrachtung der historischen Entwicklung der EDA-Werkzeuge hergeleitet. Die Extrapolation dieser Historie lässt eine Wegskizze für einen neuen „constraint-driven“ Designflow erkennen.

1. Was sind Constraints?

1.1. Layout als Optimierungsproblem

Die Aufgabe des Layoutentwurfs einer elektronischen Schaltung kann beschrieben werden als Optimierungsproblem unter Einhaltung von Randbedingungen. Grundsätzlich sind also *Optimierungsziele* und *Randbedingungen* zu unterscheiden.

Eine anschauliche Beschreibung dieses Sachverhalts findet sich in [2]: Optimierungsziele dienen der Verbesserung der Leistungsparameter einer Schaltung, ihres Gebrauchswertes, ihrer Zuverlässigkeit etc. Typische Beispiele für Optimierungsziele sind minimale Chipfläche *A* oder minimale Gesamtverdrahtungslänge *L*. Der Grad ihrer Erfüllung ist also ein Kriterium für die Qualität des entwickelten Layouts.

Da Optimierungsziele algorithmisch häufig schwer fassbar sind, ist ein oft gewählter Ansatz die Abbildung in eine Kostenfunktion *K*, mit der sich die Ziele über Wichtungsfaktoren w_i gewichten und so die Gesamtkosten *K* quantifizieren lassen. Für das hier angedeutete Beispiel mit den Zielen *A* und *L* ergibt sich für die zu minimierende Kostenfunktion $K = w_1*A + w_2*L$.

Neben diesen graduellen Qualitätskriterien stellen Randbedingungen harte Kriterien dar, deren Einhaltung zwingend notwendig ist. Bereits die Verletzung einer Randbedingung kann eine Schaltung unbrauchbar machen. Diese Randbedingungen werden als *Constraints* bezeichnet.

Das Problem der Layoutsynthese kann demnach als eine Suche in einem n-dimensionalen Lösungsraum dargestellt werden. Dabei entspricht jeder zu Beginn des Layoutentwurfs vorhandene Freiheitsgrad einem von n Parametern, durch die ein n-dimensionaler Lösungsraum aufgespannt wird.

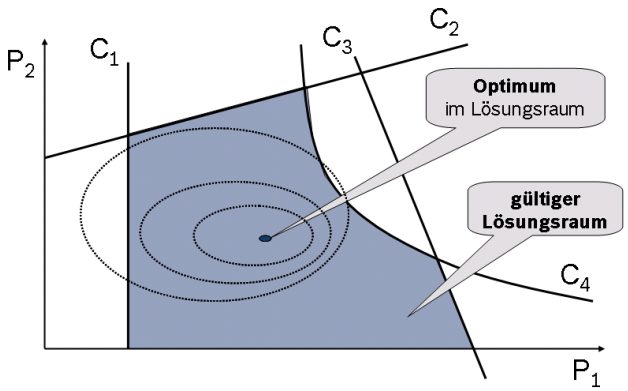


Bild 1: 2-dimensionaler Lösungsraum mit vier Constraints.

Die Constraints bewirken dabei eine Einschränkung des gültigen Lösungsraums. Innerhalb dieses Lö-

[1] Originalquelle siehe [1]

sungsraums wird nun eine hinsichtlich der verfolgten Optimierungsziele optimale Lösung (z.B. mit minimalen Gesamtkosten *K*) gesucht.

Bild 1 zeigt eine Darstellung für den extrem vereinfachten Fall von nur zwei Parametern P_1 und P_2 und vier Constraints C_1 bis C_4.

1.2. Klassifizierung von Constraints

Die Begriffe Constraints und Randbedingungen werden im Folgenden synonym verwendet. Randbedingungen lassen sich in folgende Kategorien einteilen [2]:

- **Technologische Randbedingungen** werden aus der benutzten Halbleitertechnologie abgeleitet. Ihre Einhaltung gewährleistet die Herstellbarkeit des Schaltkreises.
- **Elektrische Randbedingungen** gewährleisten das angestrebte elektrische Verhalten und werden daher oft auch als funktionale Randbedingungen bezeichnet.
- **Entwurfsmethodische Randbedingungen** werden künstlich eingeführt, um die Lösung des Layoutproblems einer rechnergestützten Bearbeitung besser zugänglich zu machen (z.B. HV-Verdrahtung, Standardzellenraster).

Die Verletzung entwurfsmethodischer Randbedingungen ist demnach durchaus tolerierbar, da hierdurch weder Herstellbarkeit noch Funktion gefährdet sind. Im Sinne der Definition aus Kap. 1.1 sind also nur technologische und entwurfsmethodische Randbedingungen immer zwingend einzuhalten. Diese Randbedingungen werden daher hier als *primäre* Randbedingungen bezeichnet. Im Folgenden seien ausschließlich primäre Randbedingungen bzw. Constraints gemeint.

1.3. DRC und LVS - der entscheidende Unterschied

Nach heutigem Stand der Technik kann ein Layoutergebnis mit DRC vollständig hinsichtlich Einhaltung der bestehenden technologischen Randbedingungen verifiziert werden. D.h. die Herstellbarkeit einer mikroelektronischen Schaltung lässt sich durch EDA-Werkzeuge vollständig absichern.[2]

Bei elektrischen Randbedingungen ist die Situation anders. Die Funktion einer Schaltung lässt sich mit LVS nur partiell absichern. Eine Vielzahl elektrischer Randbedingungen ist nicht als Information im Schaltplan enthalten und wird heute nur durch Anwendung von Expertenwissen in zusätzlichen (manuellen, visuellen oder halbautomatischen) Prüfschritten verifiziert. Elektrische Randbedingungen ergeben sich i.a. aus den parasitären Eigenschaften der integrierten Schaltkreise. Hier nur einige Beispiele:

- Maximale Spannungsabfälle auf Leitungen,
- Maximale Verlustleistungsdichte,
- Schirmungsmaßnahmen zur Vermeidung unerwünschter Kopplungen,
- Symmetrievorgaben zur Erzielung elektrischen Gleichlaufs von Parametern hinsichtlich
  - Fertigungstoleranzen, Driften,
  - Leitungsparasiten,
  - Wärmeverteilung (→ Isothermen),
  - Piezoeffekten (→ Isobaren).

Ein Hauptziel in der Weiterentwicklung von EDA-Werkzeugen muss daher sein, diese Lücke zu schließen. Warum das so ist und was letztlich vom erfolgreichen Erreichen dieses Zieles abhängt, soll im nächsten Kapitel hergeleitet werden.

2. EDA-Evolution für den Layoutentwurf analoger ICs

2.1. Historie

In [2] findet sich eine detaillierte Darstellung der historischen Entwicklung von Entwurfswerkzeugen für die Mikroelektronik beginnend in den 60-er Jahren. Kennzeichnend für diese Entwicklung sind vor allem die Erfolge, die in der zunehmenden Automatisierung des Entwurfs von Digitalschaltungen erzielt wurden.

An dieser Stelle soll nur auf die für den Layoutentwurf analoger ICs entscheidenden Aspekte eingegangen werden.

Die erste wirksame Rechnerunterstützung für den Entwurf analoger Layouts wird markiert durch die Verbreitung spezialisierter Grafik-Editoren während der 80-er Jahre. Diese Editoren wurden zwar stetig weiter entwickelt. Ihre grundlegenden Funktionen („*polygon pushing*“) bilden aber bis heute die Grundlage für alle Werkzeuge des analogen Layoutentwurfs.

Wichtige Meilensteine waren die Möglichkeiten zur rechnergestützten Verifikation der prozesstechnischen Entwurfsregeln (DRC) und die rechnergestützte Prüfung auf Konformität des Layoutergebnisses mit einem gegebenen Schaltplan bzw. einer Netzliste (LVS) im Zeitraum von etwa 1985 - 1995.

Erst als diese automatisierte Verifikation erreicht war, konnten sich auf dieser Basis weitere Funktionen zur

[2] Die für hochmoderne DSM-Prozesse notwendigen DfM-Verfahren und die damit verbundenen Probleme seien an dieser Stelle der Einfachheit halber außer Acht gelassen.

Unterstützung des analogen Layoutentwurfs durchsetzen, wie z.B. Bauelementgeneratoren, Autorouter, Kompaktoren.

2.2. Der heutige SDL-Flow

Aber bleiben wir bei der Chronologie: Seit Mitte der 90-er Jahre hat sich für den Entwurf analogen Layouts zunehmend der SDL-Flow (*schematic driven layout*) etabliert, bei dem die Layoutentstehung direkt durch den umzusetzenden Schaltplan gesteuert wird (Stichwort „*correct by construction*“) [3].

Bild 2 zeigt im linken Teil diese Entwicklung auf einem grob angedeuteten Zeitstrahl.

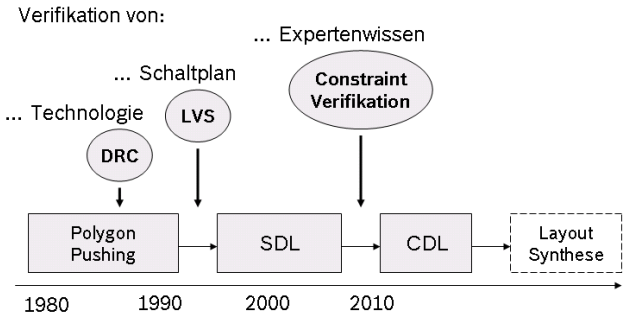


Bild 2: EDA-Evolution für den Layoutentwurf analoger ICs.

Entscheidend ist hierbei die Erkenntnis, dass für eine erfolgreiche Automatisierung von Entwurfsschritten zunächst die Fähigkeit zur automatischen Verifikation der Ergebnisse dieser Entwurfsschritte – ungeachtet der Art und Weise wie diese Ergebnisse erzielt wurden – vorhanden sein muss.

Am Beispiel der SDL-Methodik wird dies anschaulich deutlich. Der Layoutentwickler wird bereits bei der Erstellung des Layouts geführt, indem die Richtigkeit des entstehenden Layouts hinsichtlich folgender Aspekte erzwungen und fortlaufend geprüft wird („*correct by construction*“):

- Auswahl der Bauelementtypen,
- Dimensionierung der Bauelemente,
- Elektrische Verbindungen (Netze).

Dies sind exakt die Aspekte, welche im vorausgegangenen Schritt der EDA-Evolution durch den LVS einer vollständigen rechnergestützten Prüfung zugänglich gemacht werden konnten (s. Bild 2). Es sind auch diejenigen Aspekte, bei welchen in der letzten Dekade die deutlichsten Fortschritte hinsichtlich einer weiteren Automatisierung der Layoutgenerierung erzielt wurden: Bauelementgeneratoren und Autorouter.

2.3. Schlussfolgerung für die weitere Entwicklung

Der Autor ist der Überzeugung, dass hierbei ein Grundprinzip erkennbar ist, welches allgemein auf die EDA-Entwicklung anzuwenden ist:

Die Fähigkeit zur rechnergestützten (automatischen) Analyse ist stets eine notwendige Voraussetzung für eine erfolgreiche Rechnerunterstützung (Automatisierung) der Synthese!

Bild 2 zeigt im rechten Teil das Ergebnis aus der Anwendung dieses Prinzips. Voraussetzung für eine erfolgreiche automatische Layoutsynthese ist demnach, dass alle elektrischen Randbedingungen, welche heute im Allgemeinen über die im Schaltplan enthaltenen Informationen hinaus gehen und nur in Form von Expertenwissen vorliegen und Berücksichtigung finden, zunächst rechnergestützt prüfbar sein müssen.

In Analogie zum SDL-Flow, welcher auf Basis eines LVS möglich wurde, wird die umfassende rechnergestützte Verifikation von Constraints die Grundlage schaffen für einen Constraint-driven Layoutflow (CDL). Dieser stellt seinerseits einen notwendigen Meilenstein dar für die Entwicklung einer erfolgreichen automatischen Synthese analogen Layouts.

Die EDA-Evolution für das Layout analoger Schaltungen wird also in folgenden Schritten ablaufen:

1. Constraint Verifikation
2. Constraint driven Layoutdesign
3. automatische Layoutsynthese

3. Schritte zur Constraint-Verifikation

3.1. Voraussetzungen

Heute verfügbare Verifikationswerkzeuge für den IC-Entwurf konzentrieren sich meist auf die schnelle Bearbeitung weniger spezieller Verifikationsaufgaben für eine quantitativ hohe Anzahl von Verifikationsinstanzen. Darüber hinaus finden sich in den heute gängigen IC-Entwurfsumgebungen zwar verschiedene Managementsysteme zur Verwaltung von Constraints. Diese erlauben aber nur die Beschreibung der durch die jeweilige Entwurfsumgebung unterstützten Constraint-Typen und sind darüber hinaus durchweg proprietär und zueinander inkompatibel.

Um das in Kap. 1.3 angedeutete Ziel einer vollständigen Absicherung aller für die Funktionsfähigkeit eines ICs relevanten Randbedingungen zu erreichen, sind daher insbesondere für die Verifikation von analogen Schaltkreisen neuartige Ansätze notwendig, die es erlauben, eine Vielzahl von komplex zusammenhängen-

den und domainübergreifenden Randbedingungen berücksichtigen zu können.

In [4] wird ein neuartiges Verifikationsrahmenwerk namens *Constraint-Engineering-System* (CES) vorgestellt, das diese Forderungen erfüllt. Als notwendige Komponenten einer solchen umfassenden Constraint Verifikation werden genannt:

1. **Abbildbarkeit**
   Constraints müssen vollständig und eindeutig in der Entwurfsumgebung abbildbar sein.
2. **Maschinelle Verarbeitbarkeit**
   Constraints und Verifikationsaufgaben müssen einheitlich, in abstrahierter Form, maschinell verarbeitbar und werkzeugübergreifend dargestellt werden.
3. **Flexibilität**
   Relevante Constraints müssen zu jedem Zeitpunkt im Entwurfsablauf adressierbar sein.
4. **Transparenz**
   Es bedarf einer Möglichkeit zum transparenten Zugriff auf alle relevanten Designinformationen einer Verifikationsaufgabe.

Hauptmerkmal des CES ist die Fähigkeit, für die Bearbeitung komplexer Verifikationsaufgaben beliebige Verifikations- und Simulationswerkzeuge problemspezifisch und flexibel miteinander kombinieren zu können. Für weitere Details sei auf [4] verwiesen.

Für den Fortgang der Argumentation an dieser Stelle ist die Eigenschaft der Abbildbarkeit von Bedeutung.

3.2. Abbildung von Constraints

Um den Entwurf einer integrierten Schaltung mit Entwurfswerkzeugen unterstützen zu können, müssen die für den Entwurf relevanten Schaltungseigenschaften in die Toolumgebung transferiert werden. Hierzu bedarf es grundsätzlich einer geeigneten Abbildung dieser Eigenschaften. Diese Abbildung geschieht im Allgemeinen in mindestens zwei Schritten:

1. Abbildung von der physischen Ebene in eine Metaebene.
2. Abbildung von der Metaebene in die Toolebene.

Am Beispiel DRC wird dies anschaulich klar. In einem ersten Schritt werden die durch den Herstellprozess gegebenen Bedingungen (z.B. optische Abbildungsgenauigkeiten, Justiertoleranzen etc.) zunächst in textuelle Designregeln abgebildet (i.a. Mindestabstände und -breiten, minimale Überlappungen etc.). Diese Metabeschreibung dient als Grundlage für eine zweite Abbildung: die toolspezifische Programmierung von Rulefiles.

Wichtig hierbei ist die Rückwirkung der zweiten Abbildung auf die erste. Bereits bei der textuellen Beschreibung der Designregeln (Metaebene) sind die toolspezifischen Möglichkeiten und Grenzen der anschließenden Rulefile-Erstellung und der sie interpretierenden Algorithmen hinreichend zu berücksichtigen, um eine lückenlose Abbildung der physischen Realität in das Rulefile und damit eine vollständige Prüfbarkeit mit dem DRC sicher zu stellen. Einfach ausgedrückt: Designregeln müssen so formuliert sein, dass sie in einem DRC-Rulefile programmierbar sind.

Dasselbe gilt sinngemäß für alle weiteren Constraints, d.h. insbesondere auch für die hier interessierenden elektrischen Randbedingungen. Dies soll anhand eines einfachen Beispiels veranschaulicht werden.

Zur Beschreibung eines Constraints, mit dem die Funktion einer Schaltung hinsichtlich der Auswirkungen eines parasitären Leitungswiderstands abgesichert werden soll, sind folgende Abbildungen notwendig:

1. Abbildung der elektrischen Randbedingung (diese ergibt sich durch Schätzung, Berechnung oder Simulation auf der physikalisch-funktionalen Ebene) in einen *schaltungsspezifischen* Constraint. Ergebnis: die Netzverbindung bekommt als Vorgabe einen maximalen Spannungsabfall (in Volt).
2. Abbildung dieses schaltungsspezifischen Constraints in einen *layoutspezifischen* Constraint. Ergebnis: die Netzverbindung bekommt als Vorgabe einen maximalen Widerstandswert (in Ohm). Dies ist ein wichtiger zweiter Schritt, da im Layout die Angabe eines Spannungsabfalls in Volt nicht direkt verwertbar ist.
3. Zur Berechnung von Leitbahnwiderständen im Layout bedarf es schließlich noch einer formelmäßigen Beschreibung der elektrischen Widerstandswerte in Abhängigkeit von der Layoutgeometrie der Leitbahnelemente. Erst mit dieser dritten Abbildung sind wir für das vorliegende Beispiel auf der (grafischen) Layoutebene angelangt.

3.3. Denken in „Freiheitsgraden“

Wir wollen dieses Beispiel nutzen, um das in Kap. 1.1 zur Erläuterung des prinzipiellen Unterschieds von Optimierungszielen und Randbedingungen eingeführte Lösungsraummodell etwas näher zu betrachten und besser zu verstehen.

Leitbahnen in integrierten Schaltungen bestehen aus Mikrostrip-Leitungen in verschiedenen Metallebenen, die über Durchkontaktierungen (*Vias*) miteinander verbunden werden. Die geometrischen Eigenschaften dieser Leitbahnelemente können als *Freiheitsgrade* aufgefasst werden:

- Länge, Breite, Fläche,

- Technologieebene (Layer),
- Abstände,
- (relative und absolute) Koordinaten,
- Richtungen (z.B. horizontal, vertikal, diagonal).

In analoger Weise lassen sich für die Bauelemente einer Schaltung ebenso Freiheitsgrade definieren, z.B.:

- Höhe, Breite, Fläche, Form
- Lage und Form der Anschlusspins,
- Abstände,
- (relative und absolute) Koordinaten,
- Orientierung,
- Symmetrieeigenschaften.

Diese Freiheitsgrade können als Parameter des in Kap. 1.1 angesprochenen Lösungsraums aufgefasst werden. Mit der erfolgreichen Bestimmung von Parameterwerten für alle vorhandenen Freiheitsgrade wird ein Punkt im Lösungsraum definiert, was einem (möglichen) Layoutergebnis entspricht.

Die in Kap. 3.1 genannten Eigenschaften eines EDA-Systems zur umfassenden Verifikation von Constraints beruhen also letztlich auf der erfolgreichen Implementierung von Mechanismen zur Abbildung der Constraints auf die Abstraktionsebene dieser Freiheitsgrade und der Beschreibung ihrer Relationen.

4. Kontinuierlicher Layoutflow

4.1. Nachteile des klassischen sequentiellen Layoutflows

Im Freiheitsgradmodell lässt sich der Vorgang des Layoutentwurfs verstehen als die schrittweise Reduzierung von anfänglich vorhandenen Freiheitsgraden. Mit der Bestimmung jedes Parameters nimmt die Zahl der verfügbaren Freiheitsgrade ab bis zum Schluss ein Punkt des Lösungsraums erreicht wird, d.h. alle Parameter bestimmt und somit kein Freiheitsgrad mehr vorhanden ist.

Dieser Vorgang ist für den Layoutentwurf analoger Schaltkreise nicht nur durch einen interaktiven und weitgehend manuellen Entwurfsstil gekennzeichnet. Üblicherweise ist das Vorgehen auch untergliedert in Entwurfsschritte, die sequentiell abgearbeitet werden. Diese Schritte sind klassischerweise:

- Floorplanning
- Bauelemente generieren
- Platzierung
- Verdrahtung (Routing)
- Verifikation

Die erwähnte „schrittweise“ Reduzierung von Freiheitsgraden ist also durchaus wörtlich zu verstehen in dem Sinne, dass in jedem dieser Entwurfsschritte jeweils eine ganze Gruppe von Freiheitsgraden auf einmal verschwindet. Im linken Teil von Bild 3 wird die stufenweise Reduktion der Freiheitsgrade veranschaulicht.

Zum Beispiel werden beim Routing eines Netzes gleichzeitig Layer, Leitbahnbreite, Länge, Richtung und absolute Koordinaten festgelegt. Dies ist durch die heute üblichen Routingwerkzeuge unumgänglich. In vielen Fällen müssen die hierbei getroffenen Entscheidungen zu einem späteren Zeitpunkt ganz oder teilweise revidiert werden, da zum Zeitpunkt der Verlegung der betreffenden Leitbahn manche der hierfür relevanten Randbedingungen noch nicht bekannt oder sichtbar waren. Die Folge sind Entwurfszeit verlängernde Rekursionen.

Diese Nachteile treten grundsätzlich bei allen genannten, klassischen Designphasen auf.

4.2. Idee eines kontinuierlichen Layoutflows

Viel günstiger wäre es, wenn im Verlauf des Layoutentwurfs immer nur diejenigen Parameter bestimmt (d.h. diejenigen Freiheitsgrade eliminiert) werden müssten, deren Bestimmung zum jeweiligen Zeitpunkt zweifelsfrei möglich ist.

Der hierfür vorgeschlagene methodische Ansatz besteht darin, von den klassischen sequentiellen Designphasen zu einem *kontinuierlichen Design* überzugehen. Hierbei soll das Layout zunächst auf einer symbolischen Ebene entstehen und erst nach und nach physikalisch realisiert werden, indem man die bestehenden Freiheitsgrade einzeln, d.h. zu verschiedenen Zeitpunkten festlegen kann.

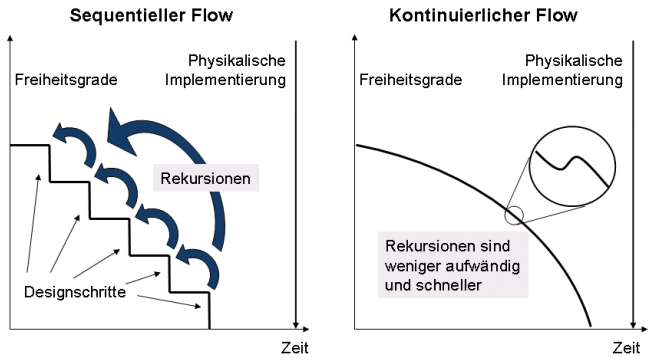


Bild 3: Vergleich sequentieller Layoutflow vs. kontinuierlicher Layoutflow.

Bild 3 veranschaulicht den Unterschied zum klassischen sequentiellen Flow: Die Reduktion der Freiheitsgrade erfolgt kontinuierlich. Das physikalische Layout „kristallisiert“ erst ganz am Schluss mit Verschwinden der letzten Freiheitsgrade. Die Zwischenergebnisse der vorherigen Phasen auf höheren Abstraktionsebenen können als „symbolisches“ Layout verstanden werden

Im oben genannten Beispiel des Routings könnte es beispielsweise sinnvoll sein, sich zu einem frühen Zeitpunkt zunächst nur für einen Layer zu entscheiden. Danach könnte der Layouter dann auf Basis weiterer Erkenntnisse eine Vorzugsrichtung festlegen und noch später, wenn z.B. bestimmte Stromflüsse in Teilnetzen feststehen, die Leitbahnbreite usw.

Die durch die Umsetzung einer Constraint Verifikation erfolgte Aufschlüsselung der Layoutelemente in ihre Freiheitsgrade und die dabei erfolgte Beschreibung ihrer Abhängigkeit von den (technologischen und elektrischen) Constraints bieten sich Chancen zur Weiterentwicklung in dieser Richtung. Ein Constraint-driven Designflow könnte demnach gleichzeitig mit den Eigenschaften eines in diesem Sinne *kontinuierlichen* Layoutflows ausgestattet werden.

4.3. Vorteile eines kontinuierlichen Layoutflows

4.3.1. Effizienz und Qualität

Durch diese neuartige Vorgehensweise ließen sich entscheidende Nachteile im bisherigen sequentiellen Verfahren überwinden, die daraus erwachsen, dass die als Expertenwissen vorliegenden Constraints jeweils zur (meist manuellen) Steuerung bestimmter Designphasen verwendet werden. Dadurch kommt es oft zwangsläufig dazu, dass Constraints, die erst in nachfolgenden Phasen Eingang finden, nicht mehr ausreichend berücksichtigt werden können, da durch Festlegungen in vorausgegangenen Phasen bereits günstige Lösungen „verbaut“ wurden. Dies führt entweder zu qualitativ ungünstigen Ergebnissen oder – falls die Lösung unakzeptable Nachteile hat - zu Rekursionen, die die Entwurfszeit verlängern.

Kennzeichnend für den vorgeschlagenen kontinuierlichen Layoutentwurf ist somit die Möglichkeit, sämtliche vorhandenen Freiheitsgrade explizit, d.h. *einzeln* und zu *beliebigen Zeitpunkten* bestimmen zu können anstatt wie bisher infolge der Bindung an die bekannten Designphasen nur implizit, d.h. *gruppiert* und nur zu *bestimmten Zeitpunkten*, wodurch sich im Ergebnis die beschriebenen Qualitäts- und Effizienzsteigerungen ergeben.

4.3.2. Unterstützung von Reuse

Die Erfahrung lehrt, dass die Wiederverwendung (*reuse*) fertigen Layouts in den meisten Fällen sehr schwierig ist. Häufige Gründe für das Scheitern von Reuse auf Layoutebene sind die folgenden:

- Die Schaltung ist zu applikationsspezifisch.
- Schaltungsänderungen, auch wenn diese anscheinend nur sehr gering sind, haben große Auswirkungen auf das Layout.
- Die Halbleitertechnologie wechselt.
- Die Form des Layoutblocks ist ungünstig und passt nicht in den geforderten Floorplan.

Die Ursache all dieser Hinderungsgründe kann auf einen Nenner gebracht werden: Fertiges Layout hat keine Freiheitsgrade!

Das im kontinuierlichen Design enthaltene Freiheitsgradmodell eröffnet hier völlig neue Wege zur Wiederverwendbarkeit vorhandener Layoutlösungen, indem man bereits vorhandenes Layout so wiederverwendet, dass nur die durch Constraints bestimmten Freiheitsgrade übernommen werden. Der Kopiervorgang geschieht somit auf einer höheren (symbolischen) Abstraktionsebene. Dadurch lassen sich auch Layouts von topologisch ähnlichen Schaltungen, selbst wenn sie in anderen Prozessen vorliegen, in Form von „Templates“ wiederverwenden. Der entscheidende Vorteil liegt darin, dass notwendige projektspezifische Anpassungen, an denen die Wiederverwendung fertigen Layouts in den meisten Fällen scheitern, nunmehr problemlos möglich sind.

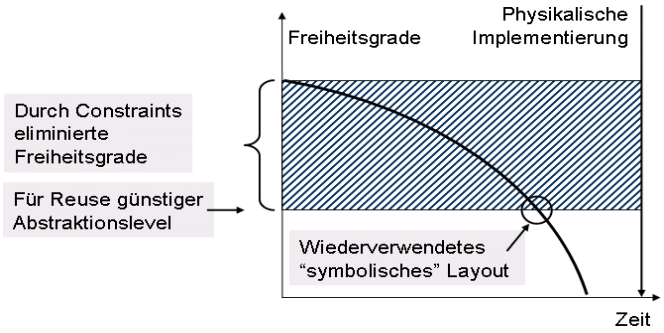


Bild 4: Reuse im kontinuierlichen Layoutflow.

In Bild 4 wird versucht, dies zu veranschaulichen. Der obere schraffierte Teil kennzeichnet die durch Constraintvorgaben eingeschränkten Freiheitsgrade. Dies bedeutet anschaulich, dass der Reuse nun genau auf dem Grad der Abstraktion erfolgt, wo nur noch die für die elektrische Funktion unwichtigen (d.h. nicht durch Constraintvorgaben eingeschränkte) Freiheitsgrade übrig sind, während das Expertenwissen aber vollständig in die Wiederverwendung einfließt.

An dieser Stelle soll noch auf ein neu auf dem EDA-Markt erschienenes Entwurfswerkzeug hingewiesen werden, das diesem Prinzip in einigen Aspekten recht nahe kommt. Es handelt sich um das Tool *1STONE* der Firma IPGEN Rechte GmbH.

Dieser neuartige Ansatz ermöglicht es, „einen Full-Custom-Entwurfsprozess, unter Berücksichtigung einer Vielzahl von Rahmenbedingungen, mittels einer speziellen Skriptsprache zu automatisieren“ [5]. Die in Skriptform vorliegende Layoutbeschreibung stellt dabei den für Reuse optimal geeigneten Abstraktionslevel dar. Die einzuhaltenden Constraints sind dort im Allgemeinen zwar nicht explizit beschrieben, aber implizit enthalten. Variationen der für die Funktion unrelevanten Freiheitsgrade lassen sich programmieren oder durch manuelle Nacharbeiten am entstandenen Layout einbringen

4.3.3. Chancen zur Weiterentwicklung

Ein weiterer Vorteil eines Constraint-driven Designflows liegt darin, dass die Entwicklung automatischer Layoutsyntheseverfahren nicht mehr auf ad hoc Ansätze angewiesen ist, sondern sich nunmehr schrittweise verfolgen lässt, da die Abhängigkeit der Schaltungsfunktionalität von der Menge der Constraints in herunter gebrochener Form vorliegt.

5. Literatur

[1] Scheible, J.: Constraint-driven Design – Eine Wegskizze zum Designflow der nächsten Generation. Analog `08, 10. GMM/ITG-Fachtagung „Entwicklung von Analogschaltungen mit CAE-Methoden“, Siegen, 02.-04.04.2008, S. 153-158, VDE-Verlag.

[2] Lienig, J.: Layoutsynthese elektronischer Schaltungen. Berlin, Heidelberg: Springer, 2006.

[3] Scheible, J.: Device Generatoren und Autorouter für den Layoutentwurf analoger IC-Schaltungen in modernen Mischprozessen. Analog `96, 4. GMM/ITG-Diskussionssitzung „Entwicklung von Analogschaltungen mit CAE-Methoden“, Berlin, 1.-2.10.1996, S. 311-318.

[4] Freuer, J.; Jerke, G.; Schäfer, A.; Hahn, K.; Brück, R.; Nassaj, A.; Nebel, W.: Ein Verfahren zur Verifikation hochkomplexer Randbedingungen beim IC-Entwurf. Analog `06, 9. ITG/GMM-Fachtagung „Entwicklung von Analogschaltungen mit CAE-Methoden“, Dresden, 27.-29.09.2006, S. 75-80, VDE-Verlag.

[5] Wittmann, R.; Rosendahl, D.: Ausführbare Entwurfsablaufbeschreibungen für einen sicheren und effizienten Entwurfsablauf. Silicon Saxony Workshop „Entwurf von integrierten Analog- / Mixed-Signal- / HF-Schaltungen, Dresden, 10.05.2007, S. 16-17.

M16C als Soft-Core-Prozessor

Ch. Kielmann, J. Skorka, I. Schoppa
HTWG Konstanz, Brauneggerstr. 55, 78462 Konstanz
Email: {chkielma, juskorka, ischoppa}@htwg-konstanz.de

Die Synthese eines Prozessors ist eine große Herausforderung und anspruchsvolle Aufgabe, insbesondere dann, wenn es sich um den Nachbau eines kommerziellen Prozessors unter der Einhaltung der Kompatibilität auf der Befehlsebene handelt. Im Rahmen zweier Bachelorarbeiten wird an der HTWG Konstanz die zentrale Prozessoreinheit (CPU) des universellen 16-Bit-Mikrocontrollers M16C von Renesas als Soft-Core-Prozessor implementiert. Dazu entsteht ein Referenz-Modell der CPU in der algorithmischen Verhaltensbeschreibung in VHDL. Basierend darauf erfolgt die Implementierung der CPU in der strukturellen VHDL-Beschreibung auf der Registertransferebene zwecks der Synthese mit der Spartan-3-Familie von Xilinx. Ein Projektteam entwickelt eine Testumgebung mit zahlreichen Assemblerprogrammen, mit denen der Befehlssatz des Soft-Core-Prozessors validiert werden kann.

1. Einführung

Der Einsatz universeller Mikrocontroller in hoch spezialisierten Anwendungen ist manchmal durch verfügbare Ressourcen eines Mikrocontrollers begrenzt, z.B. durch den Typ und die Anzahl verfügbarer Schnittstellen. Für solche Fälle hat ein Hardware-Entwickler heute die Möglichkeit, selbst einen stark spezialisierten, an die Anforderung der Applikation angepassten Mikrocontroller mit einem programmierbaren Logikbaustein FPGA/CPLD auf der Grundlage eines Soft-Core-Prozessors mit all den gewünschten Eigenschaften, applikationsspezifischen Hardware-Komponenten und Schnittstellen zu realisieren.

1.1. Soft-Core-Prozessoren

Unter dem Begriff *Soft-Core-Prozessoren* versteht man heute Prozessoren, die in einer Hardwarebeschreibungssprache wie VHDL oder Verilog modelliert sind, und die in programmierbaren Logikbausteinen (FPGA/CPLD) zum Einsatz kommen. Auf der Grundlage von Soft-Core-Prozessoren lassen sich dann in FPGA/CPLD-Bausteinen eingebettete Systeme vollständig aufbauen, einschließlich der Daten- und Programmspeicher sowie Peripheriekomponenten.

Die meisten FPGA/CPLD-Hersteller bieten Soft-Core-Prozessoren für eigene Produktlinien an, die auf die jeweilige FPGA/CPLD-Architektur unter Verwendung primitiver Komponenten gezielt optimiert worden sind. Die strukturorientierten Beschreibungen solcher Prozessoren bestehen in der Regel aus einer Liste mit instanziierten Komponenten. Zu dieser Gruppe gehören beispielsweise 32-Bit-CPUs wie Nios [2], MicroBlaze [9] und ARM [1], oder auch 8-Bit-CPUs wie PicoBlaze [10], M8C [3] und LatticeMico8 [4].

Eine weitere Gruppe bilden Soft-Core-Prozessoren, die technologieunabhängig, meistens durch verhaltensorientierte Beschreibung von Registertransferoperationen modelliert sind, und somit für den Einsatz in verschiedenen FPGA/CPLD-Architekturen vorgesehen sind. Als Beispiele kann man hier 8-Bit-CPUs wie 6502, 8051 und HC11, oder auch leistungsstarke 32-Bit-CPUs wie SPARC und DLX nennen.

1.2. Protokollkonverter

Im Rahmen eines industriellen Kooperationsprojektes soll ein PROFIBUS-Protokollkonverter und Schnittstellenwandler entwickelt werden, der Feldgeräte, insbesondere digitale Sensoren mit PROFIBUS-DP-Netzwerken verbindet. In der Anlehnung an ein Referenz-Design sind in den System-Anforderungen u.a. die Realisierung mit einem FPGA und der Einsatz eines Soft-Core-Prozessors auf der Grundlage eines M16C-Mikrocontrollers festgelegt.

Die Bezeichnung M16C ist die allgemeine Bezeichnung einer Familie universeller 16-Bit-Mikrocontroller [5], die ursprünglich von Mitsubishi Electric entwickelt wurden, und heute von Renesas Technology weiter entwickelt und hergestellt werden. Diese Mikrocontroller basieren auf einer 16-Bit-CPU mit mehreren CISC- und einigen RISC-Merkmalen, und sind in zahlreichen Versionen mit unterschiedlichsten Schnittstellen (z.B. UART, SPI, ADC, DAC) sowie Komponenten (z.B. CRC, DMAC) ausgestattet [5]. Die M16C-Mikrocontroller sind weltweit stark verbreitet und werden in

zahlreichen Applikationen in verschiedenen Anwendungsbereichen eingesetzt.

Das Bild 1 zeigt schematisch das vereinfachte Blockschaltbild des PROFIBUS-Protokollkonverters mit den wesentlichen Hardware-Modulen und seinen Schnittstellen.

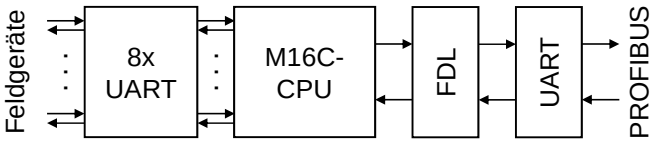


Bild 1: Blockschaltbild des PROFIBUS-Protokollkonverters.

Auf der einen Seite ist der Protokollkonverter mit acht asynchronen, seriellen Schnittstellen (UART) ausgestattet, über die er mittels RS485-Verbindungen mit digitalen Sensoren kommuniziert. Auf der anderen Seite verfügt er über ein Interface zum PROFIBUS, das u.a. ein Fieldbus-Data-Link-Modul (FDL) und eine asynchrone, serielle Schnittstelle (UART) umfasst. Der M16C als Soft-Core-Prozessor (CPU) übernimmt in diesem Protokollkonverter die Steuerung der einzelnen Komponenten sowie die Umsetzung eines proprietären Kommunikationsprotokolls der digitalen Sensoren auf den PROFIBUS-Standard [6, 7].

2. CPU

2.1. Architektur

Ursprünglich wurde die M16C-Mikrocontroller-Familie von Mitsubishi für den Einsatz in mobilen Telefonen entwickelt. Dabei lehnt sich die Architektur des Prozessors stark an die H8-Familie von Hitachi an. Der M16C-Mikrocontroller zeichnet sich u.a. durch einen geringen Energieverbrauch und gute EMI/EMS-Werte aus [15]. Die Prozessorarchitektur mit den Maschinenbefehlen ermöglicht eine hohe Codedichte. Die M16C-CPU wurde mit einer dreistufigen Fließbandorganisation entwickelt, und unterstützt überlappende Ausführung von Maschinenbefehlen. Die Architektur des M16C-CPU hat insgesamt 91 Befehle und kann bis zu 22 Interruptquellen mit sieben Prioritätsstufen verarbeiten. Die Interruptquellen sind in 17 interne und 5 externe Interrupts aufgeteilt. Der Prozessor gewährleistet eine schnelle Interruptverarbeitung durch einen schnellen Wechsel von Registerbänken.

Die M16C-Befehlsformate lassen sich generell in vier Klassen aufteilen: Generic, Quick, Short und Zero, so wie in Bild 2 schematisch dargestellt. Maschinenbefehle, die mit 8-Bit-Registern arbeiten, nutzen dabei die Formate Short und Zero. Das Short-Format hat eine Länge von einem Byte, bestehend aus den Feldern Opcode, sowie den Source- und Destination-Modis. Das Zero-Format hat ebenfalls eine Länge von einem Byte, welches Informationen über den Opcode und dem Destination-Modus enthalten. Das Quick-Format hat eine Länge von zwei Bytes. Das erste Byte enthält den Opcode, und im zweiten Byte stehen ein Immediate-Wert und der Destination-Modus. Das Befehlsformat Generic ermöglicht den Datentransfer zwischen Registern innerhalb der CPU, oder zwischen der CPU und dem Speicher. Das Generic-Format besteht ebenfalls aus zwei Bytes, und ist analog zum Quick-Format aufgebaut, allerdings mit dem Unterschied, dass dort der Immediate-Wert durch den Source-Modus ersetzt werden kann.

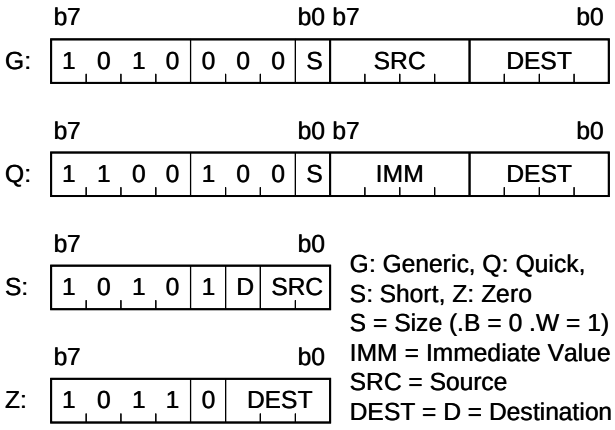


Bild 2: Befehlsformate der M16C-CPU [16].

Weitere Besonderheiten der M16C-CPU sind ein in Hardware realisierter 16x16-Bit-Multiplizierer und eine Instruction-Queue. Bei der Instruction-Queue handelt es sich um einen vierstufigen 4-Byte-Buffer, mit dem der Durchsatz in der Ausführung der Maschinenbefehle durch Prefetching erhöht wird.

Die M16C-Familie unterstützt zahlreiche Adressierungsarten, die sich in drei Gruppen unterteilen lassen: General-, Special- und Bit-Modus. Die General- und Bit-Modis unterscheiden sich darin, dass der Bit-Modus nur für die Bitmanipulation vorgesehen ist. Mit dem General- und dem Bit-Modus kann ein Adressbereich von $0000 bis $FFFF adressiert werden. Der Special-Modus kann bis $FFFFF adressieren. Nachfolgend eine Auflistung der Adressierungsarten [16].

General-Modus: registerdirekte, absolute, Adressregister-indirekte und relative, SB- und FB-relative, unmittelbare und SP-relative Adressierungen.

Bit-Modus: registerdirekte, absolute, Adressregister-indirekte und relative, SB- und FB-relativ und FLG-direkte Adressierungen.

Special-Modus: 20-Bit-absolute, Adressregister-relative mit 20-Bit-Displacement, 32-Bit-Adress-register-indirekte, 32-Bit-registerdirekte, Kontrollregis-terdirekte sowie PC-relative Adressierungen.

2.2. Referenzmodell

Das Referenzmodell dient sowohl der schnellen Überprüfung als auch der umfassenden Verifikation der Registertransferbeschreibung von der M16C-CPU. Das Referenzmodell beruht auf der algorithmi-schen Verhaltensbeschreibung. Das Bild 3 zeigt in der Übersicht einzelne VHDL-Software-Module. Es ist zu erkennen, dass sich das Referenzmodell und die Re-gistertransferebene ein gemeinsames Package sowie eine Entity teilen. In einem zentralen Package sind al-le gemeinsam verwendeten Konstanten und Datenty-pen definiert. Die Entity M16CPU enthält die Schnitt-stellenbeschreibung mit generischen Konstanten des Soft-Core-Prozessors und wird bei der Implementie-rung sowohl des Referenzmodells als auch in der Re-gistertransferbeschreibung gemeinsam benutzt.

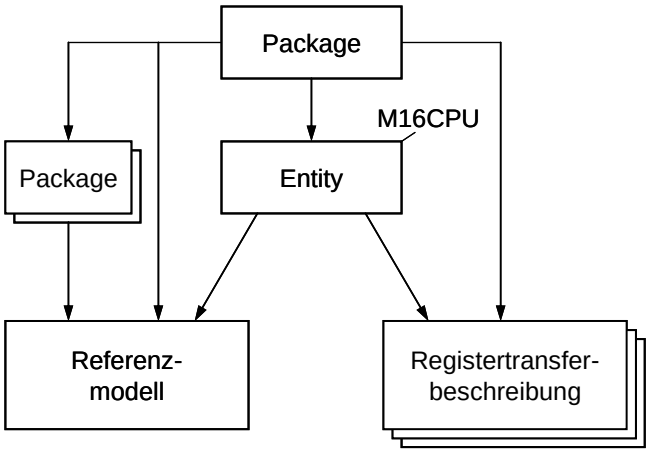


Bild 3: Software-Module und deren Abhängigkeiten.

Für die Registertransferbeschreibung sind weitere Software-Module vorgesehen, die nach funktionellen Aspekten bspw. in PC-, ALU-, Registerfile-Beschrei-bungen gegliedert sind. In den Abschnitten 5 und 6 wird näher auf die Umsetzung der Registertransferbe-schreibung eingegangen. Das Referenzmodell ist in einer Datei enthalten, und wird um weitere Pakete z.B. mit abstrakten Datentypen oder Konvertierungs-funktionen erweitert. Die Grundstruktur des Refe-renzmodells beruht im Wesentlichen auf einer Haupt-schleife, innerhalb deren sich eine CASE-Anweisung befindet. Mit Hilfe dieser Anweisung werden Maschi-nenbefehle mit den dazugehörigen Adressierungsar-ten dekodiert und weiter in entsprechenden Unterpro-grammen in einzelnen Schritten ausgeführt.

3. Werkzeuge

Für die Entwicklung des Soft-Core-Prozessors stehen folgende Werkzeuge zur Verfügung:

IAR Embedded Workbench: Der Workbench stellt uns eine Umgebung bereit, mit der die Testprogram-me entwickelt werden. Hierzu wird die Software von IAR System in der Version 3.40 Kickstart verwendet. In diesem Softwarepaket sind Assembler, Linker, C-Compiler sowie ein Simulator enthalten, mit dem die Testprogramme ausgeführt werden.

Mentor Graphics Modelsim: Um die Testumgebung des Soft-Core-Prozessor zu entwickeln, wurde der VHDL-Simulator ModelSim XE III 6.2g eingesetzt. Die Software enthält einen VHDL-Compiler und einen Si-mulator. Der Simulator stellt eine Umgebung bereit, innerhalb welcher der Soft-Core-Prozessor ausgeführt und auf Richtigkeit hin laut Spezifikation überprüft werden kann.

Xilinx ISE Project Navigator: Zur Synthese der Re-gistertransferbeschreibung des Soft-Core-Prozessors wird der ISE Project Navigator von Xilinx eingesetzt. Als Zieltechnologie dient eine Spartan 3 XC3S200 der Firma Xilinx. Für die Konfiguration des Blocks RAM wird der Xilinx Core Generator verwendet. Die Initiali-sierungsdateien für den Block RAM werden mittels der Software Data2Bram unter der Version Da-ta2Mem erstellt.

Entwicklungsboard: Als Entwicklungsboard wird das Spartan 3 Starter Kit-Board Rev. E von der Firma Xilinx eingesetzt [14].

4. Testumgebung

Die M16C-CPU hat 91 verschiedene Befehle und 21 Adressierungsarten. Für jeden der 91 Befehle und 21 Adressierungsarten wurde ein eigenes Testprogramm geschrieben. Zum Testen wurden insgesamt 550 Testprogramme mit 550 Testdateien entwickelt, was in der Summe 28287 Assemblercodezeilen ergibt.

Die Testumgebung, die schematisch in Bild 4 zu se-hen ist, besteht aus drei Entities:

- die Entity `M16CPU` mit VHDL-Beschreibungen des Prozessorkerns in zwei Versionen: algorith-mische Verhaltensbeschreibung und Register-transferbeschreibung.

- die Entity `Dual_Port_RAM` enthält eine verhaltensorientierte VHDL-Beschreibung des Dual-Port-Speichers. Diese Beschreibung umfasst u.a. Unterprogramme zum Lesen von Assemblerdateien im HEX-Format.
- die Entity `M16CPU_tb` ist die steuernde Instanz für den gesamten Test. Sie erzeugt u.a. Steuersignale für die Entity `M16CPU` und für die Entity `Dual_Port_RAM`.

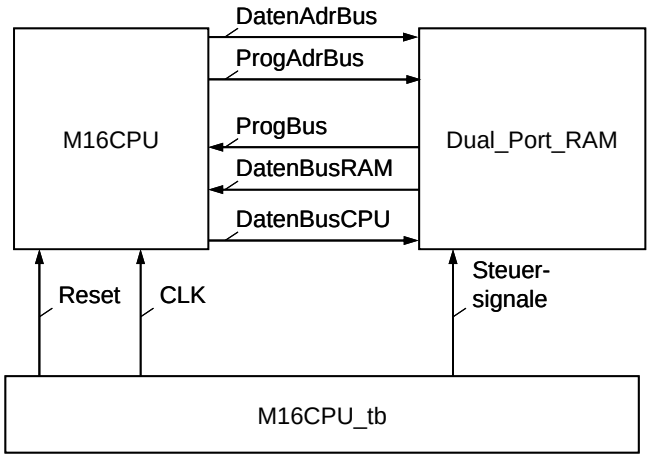


Bild 4: Testumgebung des Soft-Core-Prozessors.

4.1. Testablauf

Der M16CPU wird mit der Software ModelSim getestet. Die Entity M16CPU_tb lädt dazu Testprogramme in den Dual_Port_Speicher, die durch die M16CPU ausgeführt werden. Als Ergebnis eines Testprogramms dienen definierte Werte im Speicher, welche in der Entity M16CPU_tb mittels einer Testdatei verglichen kann. Der gesamte Testablauf von Laden bis zum Vergleich der Werte erfolgt voll automatisiert.

4.2. Testprogramme

Die Testprogramme wurden in Gruppen aufgeteilt um bestimmte Einheiten des Prozessorkerns zu testen. Der Befehl MOV testet sämtliche Adressierungsarten des Prozessors und damit die Adresseinheit.

Das Beispielprogramm im Listing 1 zeigt ein Testprogramm für den MOV-Befehl mit absoluter Adressierungsart. Zunächst wir die Funktion getestet indem ein Wert in den Speicher kopiert wird und durch die Testumgebung verifiziert wird. Anschließend wird das setzen der Flags überprüft.

```
/*
 *  Test für Befehl MOV\Format
 *  G\SRC_Immediate\dest_absolute\Byte
 *
 *  Voraussetzungen:
 *  der STC-Befehl muss funktionieren
 */
      NAME    main
      PUBLIC  main
      ASEGN   RESET:CONST, 0xFFFFC
      DC32    main
      RSEG    CODE
main
      // unter 005FH soll der Wert 00FBH stehen
      MOV.B  #0FBH,005FH

      // Test des Sign-Flags
      MOV.B  #0FFH,006FH
      // Das S-Flag sollte gesetzt sein
      STC FLG,0064H
      // speichert das FLG an der Stelle 0064H
      // Das Bit(4) bei 0064H muss gesetzt sein

      // Test des Zero-Flags
      MOV.B  #00H,007FH
      // Das Z-Flag sollte gesetzt sein
      STC FLG,0074H
      // speichert das FLG an der Stelle 0074H
      // Das Bit(3) bei 0062H muss gesetzt sein

      NOP
      END     main
```

Listing 1: Assemblercode eines Testprogramms.

Zum Überprüfen des Testprogramms ist eine Testdatei notwendig wie im Listing 2 auszugsweise gezeigt, welche die Spezifizierten Adress-Werte-Paare enthält.

Die Datei beschreibt an welcher Speicheradresse ein definierter Wert vorzufinden ist. Führt der Soft-Core-Prozessor ein Schreibzugriff auf die Speicheradresse aus, wird anschließend überprüft ob der Wert mit dem erwarteten Wert übereinstimmt.

```
-- Adresse    Wert
  a"00005f"   w"FB"
  a"000064"   w"08"
  a"000074"   w"04"
```

Listing 2: Adress-Werte-Paare einer Testdatei.

In dem Bild 5 wird die Anbindung des Soft-Core-Prozessor (M16CPU) schematisch dargestellt. Der Speicher für den M16CPU wird als Dual Port RAM beschrieben. Hierzu wird der Block RAM, der durch den FPGA zu Verfügung gestellt, wird verwendet.

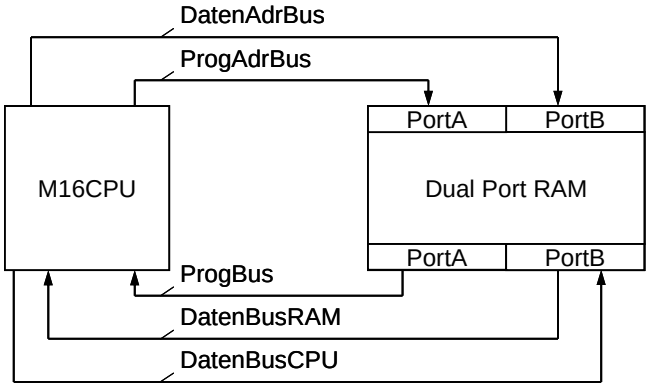


Bild 5: Blockschaltbild M16CPU und Speicher.

5. Soft-Core-Architektur

5.1. Steuerwerk

Der Soft-Core-Prozessor M16C wird mit einem mikroprogrammierbaren Steuerwerk ausgestattet, wodurch der Verbrauch an FPGA-Ressourcen (Slices) reduziert werden kann. Außerdem ermöglicht eine solche Lösung eine flexible Umsetzung von Maschinenbefehlen auf Sequenzen von Mikroinstruktionen. Jede Mikroinstruktion aktiviert einige Registertransferoperationen, die im gleichen Takt stattfinden.

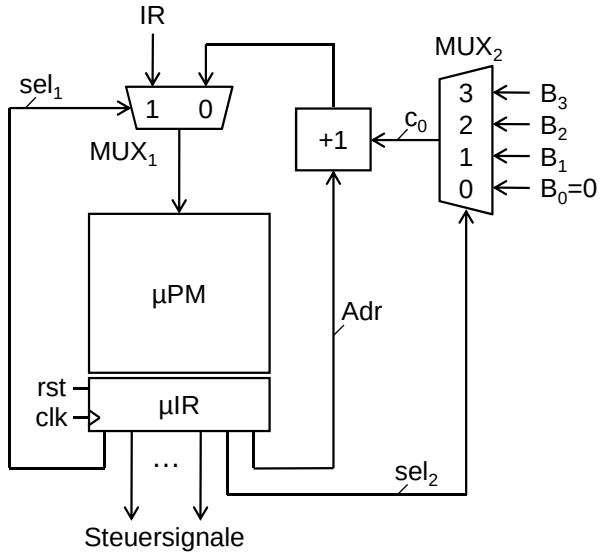


Bild 6: Aufbau eines mikroprogrammierbaren Steuerwerks.

Das Bild 6 zeigt die Grundstruktur des Steuerwerks auf der Registertransferebene. Der Mikroprogrammspeicher µPM mit dem dazugehörigen Mikroinstruktionsregister µIR wird durch synchrone Block RAMs im FPGA [8] realisiert. Neben dem Block RAM enthält das Steuerwerk auch einen Adressgenerator auf der Basis zweier Multiplexer und eines Inkrementierers. Im Steuerwerk wird sowohl Steuersignalen fürs Rechenwerk als auch Folgeadresse und Auswahlsignale für steuerwerksinterne Abläufe bereitgestellt.

Der Multiplexer MUX_1 selektiert mit Hilfe des Auswahlsignals sel_1 eine von zwei Quellen zum Adressieren des Mikroprogrammspeichers. Die erste Quelle ist der Operationscode aus dem Befehlsregister IR, der als Startadresse der dazugehörenden Sequenz von Mikroinstruktionen interpretiert wird. Die zweite Quelle ist die Adresse Adr der momentan ausgeführten Mikroinstruktion. Diese Adresse kann mit Hilfe des Inkrementierers beim Bedarf modifiziert werden, so dass bedingte Verzweigungen und somit auch Schleifen im Mikroprogrammcode möglich sind. Dazu wird der Inkrementierer mit dem Signal c_0 aus dem Multiplexer MUX_2 gesteuert. Der Multiplexer MUX_2 selektiert mit Hilfe des Steuersignals sel_2 eine von mehreren Eingangsbedingungen B_0 bis B_3. In der Abhängigkeit vom aktuellen Stand der selektieren Bedingung B_{sel2} wird die Adresse Adr entweder um 1 erhöht (bei B_{sel2}=1) oder nicht modifiziert (bei B_{sel2}=0). Für jede bedingte Verzweigung im Mikroprogrammcode werden im Mikroprogrammspeicher zwei benachbarte Speicherzellen (mit den Adressen k und k+1) mit zwei Mikroinstruktionen belegt, die Steuersignale und Folgeadressen für die Bedingungen B_{sel2}=0 und B_{sel2}=1 enthalten. Die Eingangsbedingung B_0 ist permanent mit dem Wert 0 belegt und für unbedingte Übergänge vorgesehen. Die Bedingungen B_1, B_2 und B_3 können bspw. durch Zustandsbits aus dem Flagregister FLG der CPU beeinflusst werden.

5.2. Rechenwerk

Die Einheiten des Rechenwerks sind wie Bild 7 ersichtlich, über einen internen Bus verbunden. Nachfolgend werden alle Komponenten in Ihrem Aufbau und Funktionalität beschrieben.

Registerbänke: Die M16C-CPU hat zwei Registerbänke, in denen sich vier universelle 16-Bit-Register R0 bis R3, zwei 16-Bit-Adressregister A0 bis A1 und ein 16-Bit-Frame-Base-Register FB befinden. Über das Steuerbit B im Flagregister FLG lassen sich die Registerbänke auswählen. Die Registerbänke werden im Soft-Core-Prozessor mit Hilfe von Look-Up-Tabellen realisiert [13].

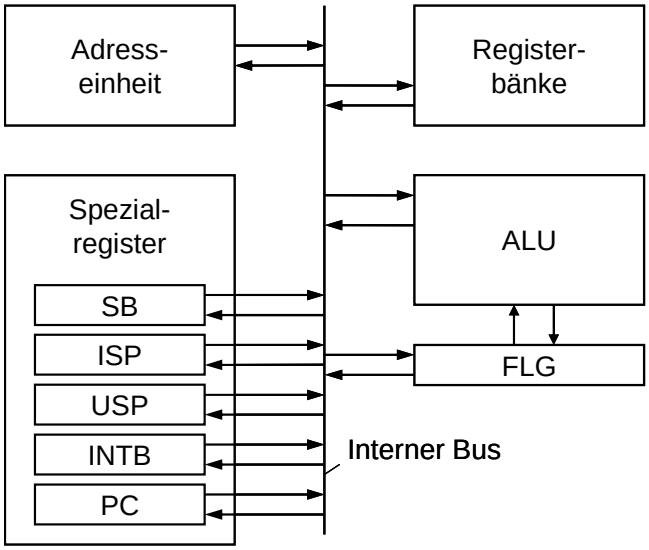


Bild 7: Struktur des Rechenwerks.

ALU: Die arithmetisch/logische Einheit ALU ist für 16-Bit-Operationen ausgelegt und stellt die gängigen

Funktionen zur Verfügung: Additionen und Subtraktionen mit und ohne Carry, Schiebe- und Rotationsoperationen, logische Operationen, vorzeichenlose und –behaftete Multiplikationen und Divisionen sowie spezielle Operationen wie die Berechnung der Summe von Produkten (RMPA) oder bedingte Bittransferoperationen (BM*Cnd*).

Adresseinheit: Die Implementierung sämtlicher Adressierungsarten erfolgt einheitlich in einer Adresseinheit, die als eine separate Komponente in dem Prozessorkern enthalten ist.

Spezial Register: Die Register SB, USP, ISP, FLG, INTB werden über Signale realisiert. Der Programm Counter ist ein 20-Bit-Addierer mit Lademöglichkeiten.

6. Literatur

1. Actel: *ARM Cortex-M1 Handbook*, Actel Corp., 2009.
2. Altera: *Nios II Processor Reference Handbook*, Altera Corp., 2009.
3. Cypress: *PSoC Programmable System-on-Chip TRM*, Cypress Semiconductor, 2010.
4. Lattice: *LatticeMico8 Microcontroller User's Guide*, Lattice Semiconductor Corp. 2010.
5. Renesas: *Renesas MCU M16C Family (R32C/M32C/M16C/R8C)*, Renesas Electronics Corp., 2008.
6. DIN EN 61158-x-3:2008-9: *Industrielle Kommunikationsnetze – Feldbusse – Teile 1-3 bis 5-3*, Hrsg. DIN Deutsches Institut für Normung, Beuth-Verlag 2008.
7. DIN EN 61784-1:2008-10: *Industrielle Kommunikationsnetze – Profile – Teil 1*, Hrsg. DIN Deutsches Institut für Normung, Beuth-Verlag 2008.
8. Xilinx: *Using Block RAM in Spartan-3 Generation FPGAs*, Application Note 463, Xilinx Inc., 2005.
9. Xilinx: *MicroBlaze Processor Reference Guide*, Xilinx, Inc., 2008.
10. Xilinx: *PicoBlaze 8-bit Embedded Microcontroller User Guide*, Xilinx, Inc., 2010.
11. IAR Systems: M16X/*R8C IAR Assembler Reference Guide for Renesas M16C/1X 3X, 6X and R8C Series of CPU Cores*, IAR Systems AB., 2004.
12. IAR Systems: *IAR Embedded Workbench IDE User Guide*, IAR Systems AB., 2009.
13. Xilinx: *Using Look-Up Tables as Distributed RAM in Spartan-3 Generations FPGAs*, Application Note 464, Xilinx, Inc., 2005.
14. Xilinx: *Spartan-3 FPGA Starter Kit Board User Guide*, User Guide 130, Xilinx, Inc., 2008.
15. Renesas: *New CISC CPU Architecture Builds on the Past and Adopts the Latest Technology to Enable Future Embedded System Advances*, Renesas Technology America, 2007.
16. Mitsubishi: *M16C/60 Series Software Manual*, Mitsubishi Electric Corp., 1997.

PLB-to-WB Bridge
Eine Brücke zwischen proprietärer und Open-Source Hardware

Christian Hättich, Prof. Dr.-Ing. Frank Kesel
Hochschule Pforzheim, Tiefenbronner Straße 65, 75175 Pforzheim
Studiengang Embedded Systems
christian.haettich@gmx.de, frank.kesel@fh-pforzheim.de

Zusammenfassung

Wie im Umfeld der Softwareentwicklung findet sich auch im Bereich der Hardware der Open-Source-Gedanke. Mit Hardware sind in diesem Zusammenhang synthetisierbare Peripherien gemeint, mit denen einfache digitale Systeme bis hin zu komplexen Systems-on-Chip erstellt werden können.
Auf der proprietären Seite ist Xilinx einer von mehreren Anbietern, der nicht nur FPGAs, sondern auch synthetisierbare IP Cores zusammen mit einem umfangreichen Softwarepaket kommerziell anbietet. Auf der anderen Seite bietet die Open Cores Community solche IP Cores unter einer Open-Source-Lizenz an und empfiehlt als Bussystem den Wishbone Bus (WB). Um proprietäre und frei erhältliche IP Cores miteinander zu kombinieren, ist eine Busbrücke nötig, da meist unterschiedliche Schnittstellen verwendet werden.
Diese Ausarbeitung beschreibt eine PLB-to-WB Bridge, die eine solche Kombination ermöglicht. Neben dem Aufbau der PLB-to-WB Bridge werden auch das Bus Functional Model und ein Beispielsystem vorgestellt.

Schlüsselbegriffe: *Systembusse, Systems-on-Chip, PLB, Wishbone, Bus Bridge, PLB-to-WB Bridge, Bus, Brücke, IP Cores*

1 Einleitung

Schon seit den Anfängen der Rechnersysteme spielen verschiedene Bussysteme eine entscheidende Rolle in der Datenkommunikation zwischen einzelnen Komponenten innerhalb von Rechnersystemen oder auch zwischen mehreren Rechnersystemen.
Prinzipiell lassen sich Busse in *serielle* und *parallele* Busse unterscheiden [1]. Bekannte Beispiele für serielle Busse sind u. a. der Universal Serial Bus (USB), Serial Advanced Technology Attachment (SATA) oder der Inter-Integrated Circuit (I^2C). Bei den parallelen Bussen bilden Peripheral Component Interconnect (PCI), Industry Standard Architecture (ISA) oder der Accelerated Graphics Port (AGP) populäre Beispiele.
Je nach den Komponenten, die über einen Bus kommunizieren, können parallele Bussysteme auch in die Kategorien *Systembusse* und *Peripheriebusse* und serielle Bussysteme in *Rechnernetze* und *Prozessbusse* unterteilt werden [4], wobei dieser Artikel den Fokus auf die System- und Peripheriebusse für *Systems-on-Chip* richtet.
Xilinx setzt in seinen auf MicroBlaze und PowerPC basierenden Systemen den von IBM entwickelten *Processor Local Bus* (PLB) ein, der einer von drei Bussen der CoreConnect Bus Architecture[1] ist. Mit diesem werden in einem typischen System-on-Chip der Speicher sowie verschiedene Peripherieblöcke mit dem Prozessor verbunden. Ein solches System kann mit der Software *Xilinx Platform Studio* (XPS), das ein Teil des *Embedded Development Kit* (EDK) ist, auf einer grafischen Oberfläche schnell und einfach zusammengestellt werden. Hierfür müssen jedoch für die einzelnen *Intellectual Property (IP) Cores* sowie auch für die Software die jeweiligen Lizenzen erworben werden. Neben diesen proprietären IP Cores bietet die OpenCores Community[2] Peripherien an, die frei[3] erhältlich sind.
Das Ziel des in diesem Beitrag beschriebenen Projekts ist die Verwendung von WB-kompatiblen IP Cores in einem MicroBlaze basierenden System-on-Chip.

1.1 Processor Local Bus

Der PLB ist ein komplexer synchroner Bus, der aus einem separaten Adressbus, einem Schreibe- und Lesebus sowie aus Steuerleitungen besteht. Dabei wird das Pipelining von Adressen, überlappende Lese- und Schreibtransfers sowie eine Datenbusbreite von 32 Bit bis zu 128 Bit unterstützt [2]. Als Transfertypen kommen *Single Transfers*, *Line Transfers* und *Burst Transfers* in verschiedenen Ausprägungen zum Einsatz. Mit dem PLB ist es möglich, maximal 16 Slaves mit maximal 16 Masters zu verbinden. Ein sogenannter Arbiter (Schiedsrichter) entscheidet darüber, welcher Mas-

[1] Die CoreConnect Bus Architecture besteht aus dem Processor Local Bus, dem On-chip Peripheral Bus und dem Device Control Register Bus.
[2] http://Opencores.org
[3] Meist durch die Open-Source-Lizenzen GPL, LGPL oder BSD geschützt.

ter den Bus nutzen darf.

1.2 Wishbone Bus

Der WB ist wesentlich einfacher strukturiert als der PLB und besteht aus einem Adressbus, einem Schreibe- und Lesebus sowie aus Steuerleitungen. Die WB-Spezifikation definiert dabei weder die Bustopologie noch elektrische Vorgaben. Die Größe des Adressbusses wie auch des Datenbusses ist modular definiert und lediglich dem Datenbus ist eine Grenze von 64 Bit gesetzt [3]. Auch die Transfertypen sind wesentlich einfacher gestaltet als beim PLB. Die Spezifikation sieht die folgenden drei Arten vor[4]: *Single Read/Write Cycles*, *Burst Read/Write Cycles* und *Read-Modify-Write (RMF) Cycles* [3].

1.3 Vergleich

Beide Bussysteme sind synchrone Busse, die sich in ihrer Komplexität unterscheiden, wie Tabelle 1 verdeutlicht.

Kategorie	PLB	WB
Anzahl Slave Verbindungen	36	13
Anzahl Master Verbindungen	33	13
Anzahl unterschiedlicher Transfertypen	8	5
Adress Pipelining	✓	
Überlappende Transfers	✓	

Tabelle 1: Unterschiede zwischen PLB und WB. Bei den Verbindungen wurden einfach alle Top Level Ports gezählt.

1.4 Busbrücken und -hierarchien

Eine Busbrücke wird verwendet, um zwei gleiche oder verschiedene Bussysteme miteinander zu verbinden. Der Grund für den Einsatz einer Busbrücke kann verschiedener Natur sein. Zum einen können damit Bushierarchien gebildet werden (siehe [4]) und zum anderen erlaubt eine Busbrücke die Verwendung von Peripherien mit unterschiedlichen Busschnittstellen in einem System, wie es auch das Ziel dieses Projektes ist. In Abbildung 1 ist eine Bushierarchie dargestellt. Dabei werden in der Regel langsamere Peripherien zusammen auf einem langsameren Bus (hier der ISA-Bus) und schnellere Peripherien auf einem schnelleren Bus (hier der PCI-bus) gruppiert. Die Verbindung zwischen den Bussen wird dann durch eine Busbrücke hergestellt. Der Vorteil der Hierarchie ergibt sich daraus, dass die Brücke einen Puffer beinhaltet. Schreibende Zugriffe über die Brücke können dadurch schnell erfolgen, auch wenn eine langsame Peripherie angesprochen wird. Dies gilt jedoch nicht für Lesezugriffe.

[4] Wishbone Classic Bus Cycles nach Revision B.3

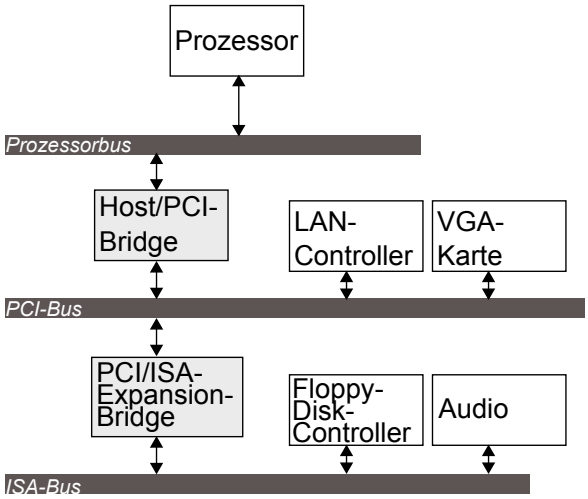


Abbildung 1: Beispiel für eine Bushierarchie: Langsame Peripherien sind mit dem ISA–Bus verbunden und schnellere Peripherien mit dem PCI-Bus (vgl. [4]).

2 Aufbau der PLB-to-WB Bridge

Die PLB-to-WB Bridge bindet einen WB an einen PLB an und ermöglicht die Verwendung von WB kompatiblen IP Cores in einem MicroBlaze System. Die Architektur der Brücke basiert grundlegend auf First In - First Out Speichern (FIFOs), die sowohl als Puffer als auch zur Entkoppelung der beiden Taktdomänen dienen. Mit der Xilinx Anwendung *Core Generator* kann ein solches FIFO erstellt und konfiguriert werden. Dabei stehen für die Implementierung vier verschiede-

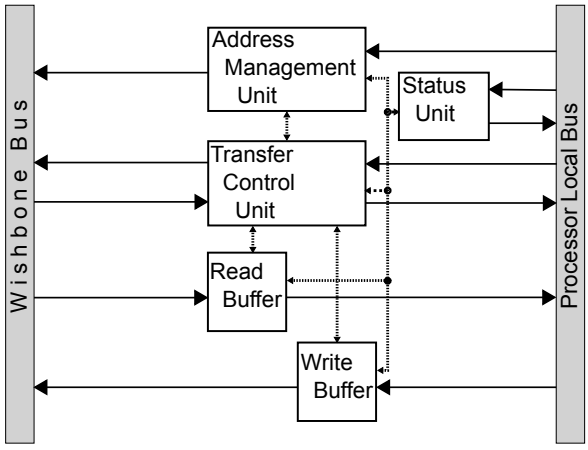
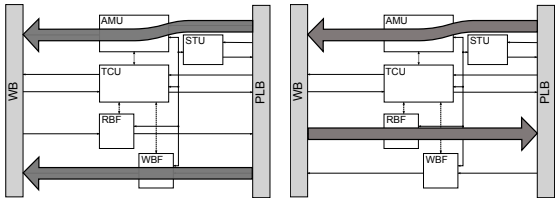


Abbildung 2: Architektur der PLB-to-WB Bridge

ne Arten für FIFOs zur Verfügung, die aber alle mit einem separaten Lese-und-Schreib-Taktsignal konfiguriert werden können. Durch die Verwendung der FIFOs wird die Entkoppelung der beiden Taktdomänen in die FIFOs verlagert und muss im restlichen Design nicht mehr beachtet werden.
Die PLB-to-WB Bridge implementiert auf der PLB-



(a) Schreibtransfer (b) Lesetransfer

Abbildung 3: Datenfluss eines Schreib- und eines Lesetransfers

Adressbereich zugeordnet, über den die WB Peripherien angesprochen werden können. Bei einem Schreibtransfer wird das Wort zusammen mit der Adresse von einem PLB Master an die Brücke übergeben, die dann das Wort an die angegebene Adresse auf der WB-Seite schreibt (Abb. 3(a)). Durch die Pufferung kann dies (innerhalb eines Taktzyklus, wenn die Puffer nicht voll sind) sehr schnell durchgeführt werden. Bei einem Lesetransfer wird erst die Adresse an die Brücke übergeben. Diese liest das gewünschte Wort auf der WB-Seite und stellt es dann für den PLB Master bereit (Abb. 3(b)). Ein solcher Lesetransfer dauert wesentlich länger als ein Schreibtransfer, da der PLB Master auf die Brücke warten muss. Zusätzlich zu dem Adressbereich, der auf dem WB abgebildet wird, gibt es einen zweiten Adressbereich, in dem vier Statusregister der Brücke angesprochen werden können.
Abbildung 2 zeigt den Aufbau der PLB-to-WB Bridge, deren Komponenten im Folgenden erläutert werden.

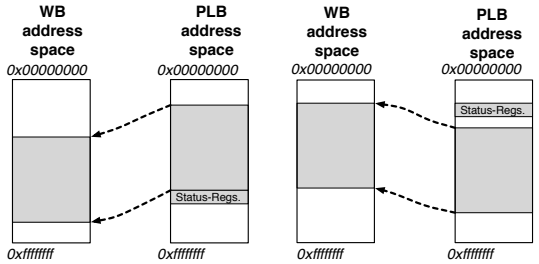
2.1 Komponenten

2.1.1 Status Unit

Eine Problematik bei gepufferten Schreibtransfers ist, dass nachträglich Busfehler auf der WB-Seite entstehen können. Hierfür wurde eine Status Unit (STU) implementiert, die die Information (Adresse und Datum des fehlgeschlagenen Schreibtransfers) in den Statusregistern auf der PLB-Seite zur Verfügung stellt. Die STU erzeugt einen Interrupt und die Software kann dann durch einen Registerzugriff entscheiden, ob der Schreibtransfer wiederholt oder abgebrochen werden soll.
Des Weiteren bietet die PLB-to-WB Bridge eine Interrupt-Schnittstelle auf der WB-Seite an. Diese Schnittstelle ermöglicht es, einen Interrupt Request (IRQ) auf die PLB-Seite weiterzuleiten. Diese Aufgabe wird auch von der STU übernommen. Zusätzlich erkennt die STU, ob der WB zurückgesetzt wurde. In einem solchen Fall sind die gepufferten Daten in der Brücke nicht mehr gültig und durch einen Registerzugriff kann die Software dann einen *Soft Reset* durchführen.

2.1.2 Address Management Unit

Neben der STU gibt es auch eine Address Management Unit (AMU), die die Pufferung der Adressen und das Adress-Pipelining implementiert. Zusätzlich zur



(a) Positiver Versatz (b) Negativer Versatz

Abbildung 4: Beispiel für Versetzung der Adressbereiche (von PLB nach WB)

Pufferung besteht auch die Möglichkeit, eine positive oder negative Versetzung der Adressbereiche statisch zu definieren. Hiermit wird von jeder Adresse, die auf der WB-Seite angelegt wird, ein fester Wert addiert (Abb. 4(a)) oder subtrahiert (Abb. 4(b)). Je nach Anforderungen ist dadurch ein flexibler Systementwurf möglich.

2.1.3 Schreib- und Lesepuffer

Die beiden Schreib- (WBF) und Lesepuffer (RBF) in Abbildung 2 sind durch FIFOs implementiert, weshalb keine zusätzliche Hardware an dieser Stelle erforderlich ist.

2.1.4 Transfer Control Unit

In der Mitte (Abb. 2) befindet sich die Transfer Control Unit (TCU), die alle anderen Komponenten steuert. Zusätzlich interagiert die STU mit fast allen[5] Steuersignalen auf WB- und PLB-Seite. Hierfür gibt es einen Automaten auf WB-Seite, der dort die jeweiligen Bus-Transfers durchführt, und zwei Automaten auf der PLB-Seite. Auf dieser Seite werden zwei Automaten implementiert, da die PLB-Spezifikation sogenannte überlappende Transfers beschreibt. Dabei können ein Lese- und Schreibtransfer gleichzeitig durchgeführt werden. Einen sogenannter Watchdog-Timer auf WB-Seite verhindert zusätzlich ein Blockieren der Bridge, falls auf eine nicht gültige Adresse zugegriffen wird. Der Zeitraum, nachdem der Watchdog den Transfer abbricht, ist parametrisierbar.

[5] Lediglich die Bestätigung einer zweiten Adresse beim Pipelining wird durch die AMU selbtständig durchgeführt.

3 Entwickeln mit dem Bus Functional Model

Das Bus Functional Model (BFM) wurde, wie auch der PLB, von IBM entwickelt und ermöglicht eine einfache und dezidierte Simulation von Buszyklen, bei der nicht das gesamte System mit dem Prozessor simuliert werden muss, was die Simulationsdauer verringert. Außerdem ist es nicht nötig, einen Assembler- oder C-Code zu schreiben, der die gewünschten Buszyklen erzeugt. Im Allgemeinen besteht das BFM aus drei Einheiten:

- OPB Toolkit
- PLB Toolkit
- DCR Toolkit[6]

Mit diesen können Simulationsmodelle für die gesamte Core-Connect-Architektur erstellt werden. Da in diesem Projekt nur der PLB zum Einsatz kommt, wird das PLB Toolkit im folgenden Abschnitt näher beschrieben.
Das PLB Toolkit, welches bei Xilinx speziell auf das EDK angepasst erhältlich ist, beinhaltet folgende Komponenten:

- PLB Master
- PLB Slave
- PLB Monitor
- BFM Synchronization Bus
- xilbfl: Bus Functional Language Compiler

Mit dem *PLB Master* und dem *PLB Slave* können Peripherien am PLB abstrakte simuliert werden. Der *PLB Monitor* hingegen dient lediglich der Überwachung des richtigen Verhaltens aller Peripherien, die an den PLB angeschlossen sind. Das Verhalten und Setup des PLB Master, des PLB Slave und des PLB Monitor kann mit der sogenannte Bus Functional Language (BFL) bestimmt werden. Der BFL-Quellcode wird danach mit dem BFL-Compiler *xilbfl* übersetzt. Das Resultat ist eine für den Simulator[7] passende Script-Datei, die vor dem Starten der Simulation ausgeführt wird, um die entsprechende Peripherie zu konfigurieren.

```
1  set_device (
2     path=/system_tb
3          /dut
4          /plb_bfm_master_32
5          /plb_bfm_master_32
6          /master,device_type=plb_master)
7  configure (msize=00)
8
9  mem_update (addr=f2000000,data=01112233)
10 mem_update (addr=f2000004,data=44556677)
11 mem_update (addr=f2000008,data=8899aabb)
12 mem_update (addr=f200000c,data=ccddeeff)
13
14 mem_update (addr=f3000000,data=01112233)
15 mem_update (addr=f3000004,data=44556677)
16 mem_update (addr=f3000008,data=8899aabb)
17 mem_update (addr=f300000c,data=ccddeeff)
18
19 write (addr=f2000000,size=0001, be=0001)
20 write (addr=f3000000,size=1010, be=0011)
21 read  (addr=f2000000,size=0000, be=1111)
22 read  (addr=f3000000,size=0000, be=0011)
```

Programmauflistung 1: BFL-Beispiel für einen PLB Master

Programmauflistung 1 zeigt ein Beispiel für einen PLB Master. In den Zeilen eins bis sieben werden der Pfad der Peripherie und dessen Größen angegeben. Danach folgt in den Zeilen neun bis 17 die Definition des Speicherinhalts, der dann in den Zeilen 19 und 20 an die Adresse `0xf2000000` und `0xf3000000` geschrieben wird. Stimmt bei einem Lesezyklus (Zeilen 21 und 22) das gelesene Wort mit dem Wort aus dem Speicherinhalt nicht überein, wird eine Fehlermeldung ausgegeben.
Zusätzlich kann ein PLB Monitor zur Überwachung genutzt werden. Zeigt irgendein Signal einer PLB-Peripherie ein falsches Verhalten, wird dies ebenfalls durch eine Fehlermeldung ausgegeben. Ein typisches

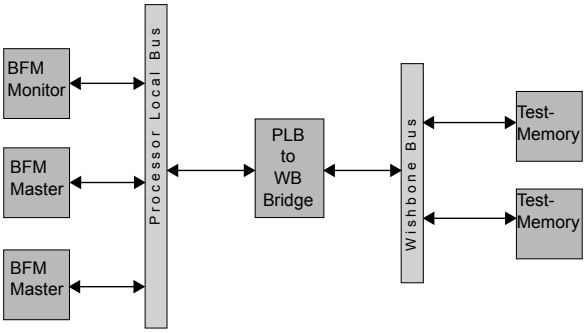


Abbildung 5: Beispielsystem zur Simulation der PLB-to-WB Bridge

Simulationssystem für dieses Projekt ist in Abbildung 5 zu sehen. Mit den beiden BFM Masters werden Buszyklen auf dem Testspeicher auf der WB-Seite initiiert. Dieser Testspeicher ist ein für dieses Projekt speziell

[6] Das DCR Toolkit ist für Xilinx Platform Studio nicht verfügbar.
[7] Zum Zeitpunkt des Verfassens dieses Beitrags wird nur der ModelSim Simulator unterstützt, der auch bei diesem Projekt zum Einsatz kommt.

entwickelter Speicher, mit dem verschiedene Situationen, wie WB Errors oder WB Retries, simuliert werden können. Zudem erlaubt dieser Speicher die Konfiguration der Lese- und Schreibverzögerung.
In der Mitte von Abbildung 5 ist die Busbrücke zu erkennen, die die Buszyklen umsetzen muss.
Aufgrund der inkrementellen Vorgehensweise bei diesem Projekt sind mehrere solcher Testsysteme, bei denen die Brücke mit mehreren Testfällen auf ihr richtiges Verhalten getestet wird, entstanden. Durch eine Makefile-gesteuerte Struktur können alle Testfälle mit einem Aufruf durchlaufen werden.

4 Xilinx XPS

Die Integration der PLB-to-WB Bridge sowie weiterer WB-Peripherien in den Xilinx-Arbeitsablauf zur Generierung von Systems-on-Chip ist ebenso Teil dieses Projekts. Ein WB Peripheral Repository beinhaltet alle WB-Peripherien, wobei der Aufbau und die Struktur des Repository von Xilinx vorgegeben sind. Wird dann der Pfad zu diesem Repository in der Projektkonfiguration angegeben, können alle Peripherien, die sich darin befinden mit dem XPS genutzt werden. Die Ordnerstruktur des Repository ist in Abbildung 6 ersichtlich. Für jede Peripherie existiert ein Ordner in *pcores* (für die Hardware) und ein Ordner in *drivers* (für die Software). In *hdl->vhdl* bzw. in *src* befinden sich die Quellcodedateien und in den Ordnern *data* die Dateien, die von XPS genutzt werden, um die Peripherien und die Treiber in das System zu integrieren und zu synthetisieren bzw. zu kompilieren. Von Bedeutung für die Integration von Peripherien ist die *Microprocessor Hardware Specification-Datei* (MHS), die im *data*-Ordner zu finden ist. Anhand dieser Datei wird für XPS definiert, wie das Interface der Peripherie aufgebaut ist, wie die einzelnen Signale verbunden werden sollen und welche Werte den Top Level Generics zugewiesen werden sollen. Des Weiterein ist im *data*-Ordner auch eine *Peripheral Analyze Order-Datei* (PAO) vorhanden, die festlegt, welche HDL-Dateien in welcher Reihenfolge synthetisiert werden.
Eine ähnliche Struktur ist auf der Softwareseite gegeben.

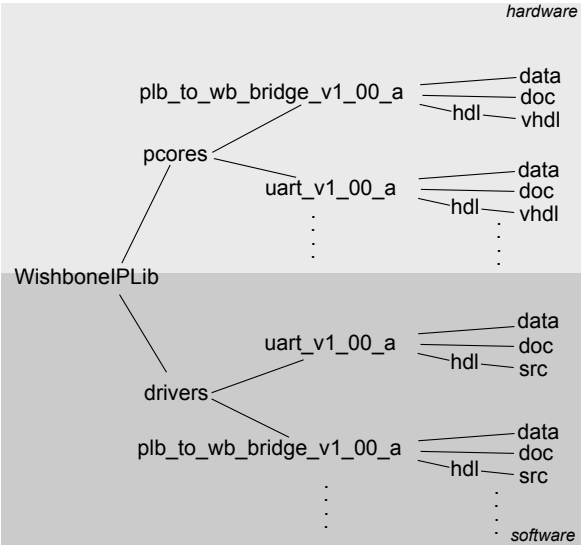


Abbildung 6: Ordnerstruktur des WB Repository

4.1 Problematik bei der Integration von WB-Perihperien

Viele WB IP Cores werden nicht nur durch Generics parametrisiert, sondern auch mithilfe von HDL-Dateien. Eine Konfiguration dieser HDL-Dateien mittels XPS-GUI ist jedoch mit einem höheren Aufwand verbunden und wird in diesem Projekt nicht weiter verfolgt.
Indessen besitzt das XPS auch eine TCL-Schnittstelle (die leider von Xilinx nicht dokumentiert ist), mit der eine Autogenerierung dieser Konfigurationsdateien möglich sein sollte.

4.2 Integrierte Peripherien

In Tabelle 4.2 sind alle IP Cores aufgelistet, die zusätzlich zur PLB-to-WB Bridge in das Repository integriert wurden.

Name	Beschreibung
wb_conbus	Wishbone Bus und Arbiter
uart	Universal Asynchronous Receiver/Transmitter
gpio	General Purpose Input Output
or1200	OpenRisc CPU

Tabelle 2: Integrierte IP Cores, die auf `http://www.opencores.org` frei erhältlich sind.

5 Beispielsystem

In diesem letzten Abschnitt wird ein Beispielsystem, das u. a. die PLB-to-WB Bridge beinhaltet, präsentiert. Dieses System ist zwar sehr einfach aufgebaut (Abb. 7), veranschaulicht aber dennoch die wesentlichen Grundfunktionalitäten der Busbrücke.

5.1 Beschreibung der Hardware

Auf der PLB-Seite befindet sich der *MicroBlaze* Soft Processor von Xilinx, der die Instruktionen und Daten aus dem daneben liegenden BRAM lädt. Der Interrupt-Controller *XIntc* sammelt alle Interrupts ein (wobei sich dies in diesem einfachen Beispielsystem auf den Interrupt der PLB-to-WB Bridge reduziert) und ist über eine weitere Interrupt-Verbindung (in Abbildung 7 zur Vereinfachung nicht eingezeichnet) mit

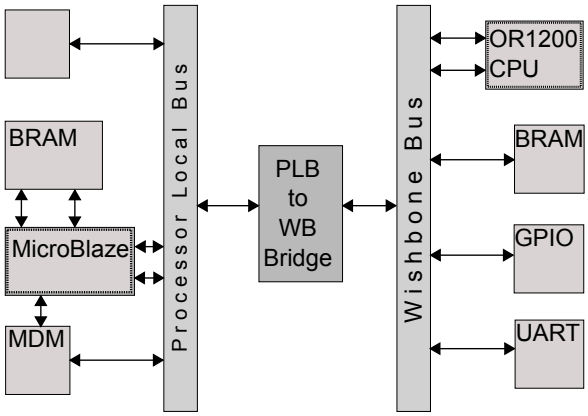


Abbildung 7: Sehr einfaches Beispielsystem für die Xilinx ML501 Evaluation Platform mit einem Virtex-5 FPGA

dem MicroBlaze verbunden. Zusätzlich ist noch eine Hardware-Debug-Einheit auf der PLB-Seite vorhanden, die es erlaubt, den MicroBlaze zu debuggen. Auf der WB-Seite befindet sich ein weiterer Soft-Prozessor: der OR1200, ein 32-Bit-RISC-Prozessor mit einer Harvard-Mikroarchitektur, einer fünfstufigen Pipeline, einer elementaren DSP-Unterstützung und ein Prozessor, der den Einsatz eines virtuellen Speichers ermöglicht. Im Beispielsystem bezieht der OR1200 die Instruktionen und Daten vom BRAM auf der WB-Seite. Zudem finden sich eine General Purpose Input Output-Einheit (GPIO) und eine Universal Asynchronous Receiver Transmitter-Einheit (UART) auf der WB-Seite. Die Interrupt-Ausgänge der GPIO- und UART-Einheit sind mit der Brücke verbunden, wobei der Interrupt-Ausgang der Brücke auf der PLB-Seite mit dem XIntc Interrupt-Controller verbunden ist (die Interrupt-Leitungen sind in der Abbildung 7 zur Vereinfachung nicht eingezeichnet). Das bedeutet, dass alle Interrupts vom MicroBlaze verarbeitet werden.
Die GPIO-Einheit ist auf der ML501 Evaluation Platform mit den LEDs, den Drucktastern und den Schiebeschaltern verbunden, um eine einfache Interaktion mit dem Benutzer zu simulieren. Die UART-Einheit ist mit dem RS232 Serial Port verbunden, über den Testausgaben und Daten seriell übertragen werden können.

5.2 Beschreibung der Software

5.2.1 OR1200

Die Software, die vom OR1200 ausgeführt wird, hat eine sehr einfache Aufgabe zu erfüllen: Sie lässt die LEDs, die über die GPIO-Einheit angeschlossen sind, blinken.

5.2.2 MicroBlaze

Die Software, die vom MicroBlaze ausgeführt wird, soll die Interrupts der GPIO- und UART-Einheit zu verarbeiten: Wird ein Drucktaster betätigt, wird eine Information über die UART-Einheit ausgegeben. Empfängt die UART-Einheit ein Byte, wird dieses als Echo zurückgesendet. Zusätzlich wird dieses empfangene Byte an die LEDs ausgegeben.

5.2.3 Konkurrenz zwischen dem MicroBlaze und dem OR1200

Da der MicroBlaze und der OR1200 die LEDs per GPIO ansteuern, stehen die beiden Prozessoren in Konkurrenz zueinander. In diesem Fall ist dies jedoch unproblematisch, da das Ausgeben eines Werts nur einen Buszyklus umfasst. Würden sich die beiden Prozessoren die UART-Einheit teilen wollen, wäre zusätzliche Hardware, wie eine Hardware-Mutex[8], nötig, da das Senden eines Byte per UART mindestens zwei Buszyklen benötigt: das Lesen des UART-Statusregisters und das Schreiben in das UART-TX-Register.

6 Schlussbetrachtung und Ausblick

Mit der PLB-to-WB Bridge können WB-kompatible IP Cores in MicroBlaze-Systemen verwendet werden und durch die Integration in das XPS ist eine Verwendung der Brücke mit dem Xilinx-Arbeitsablauf möglich.
Einen weiteren Schritt stellt die Implementierung einer *WB-to-PLB Bridge* dar, die den Zugriff von WB Masters auf PLB Slaves ermöglichen könnte. Dadurch könnten auch WB-kompatible IP Cores, die eine Master- und eine Slave-Schnittstelle[9] besitzen, in einem MicroBlaze-System eingesetzt werden. Diese WB-to-PLB Bridge müsste dann eine Slave-Schnittstelle auf der WB-Seite und Master-Schnittstelle auf der PLB-Seite implementieren.

Literatur

[1] Bernd Schürmann *Grundlagen der Rechnerkommunikation* Wiesbaden: Vieweg 2004. – ISBN: 3-528-15562-0

[2] IBM *128-Bit Processor Local Bus Architecture Specifications Version 4.7*

[3] Opencores.org *WISHBONE System-on-Chip (SoC) Interconnection Architecture for Portable IP Cores* Revision B.3

[4] Thomas Flik *Mikroprozessortechnik und Rechnerstrukturen* 7., neu bearbeitete Auflage, Berlin-Heidelberg: Springer 2005. – ISBN: 3-540-22270-7

[8] Mutex steht für *Mutual Exclusion* und verhindert, dass mehrere Tasks gleichzeitig auf einen kritische Speicherbereich zugreifen können.

[9] Meist Peripherien, die einen Zugriff auf den Speicher benötigen.

Entwicklung einer Middleware zwischen Mikroprozessoren und FPGAs

H. Hennig[1], I. Schoppa[2]

[1]SICK AG, Erwin-Sick-Str. 1, 79183 Waldkirch

[2]HTWG Konstanz, Brauneggerstr. 55, 78462 Konstanz

[1]hannes.hennig@htwg-konstanz.de

Im Rahmen einer Bachelorarbeit bei der SICK AG wird das firmeninterne Visualisierungs- und Entwicklungstool SOPAS ET um ein Software-Modul zur automatischen Anbindung von FPGAs an Mikrocontroller in SICK-Sensoren erweitert. Der Kern der Middleware besteht aus mehreren XML-Dateien, welche generische Sensorspezifikationen mit Parameter- und Schnittstellenbeschreibungen sowie Beschreibungen von Kommunikationsprotokollen beinhalten. Ein Generator-Framework setzt solche Sensorspezifikationen in C- und VHDL-Dateien um. Die C-Dateien enthalten Deklarationen und Datenstrukturen sowie Funktionen zur Steuerung der Kommunikationsprotokolle. In den VHDL-Dateien steht das kompatible Gegenstück mit Beschreibungen von Daten-, Steuer- und Statusregistern. Als Kommunikationsprotokolle zwischen FPGAs und Mikrocontrollern sind eine parallele und eine serielle, an das SPI angelehnte, Datenübertragung vorgesehen.

1. Einführung

1.1. Kurzdarstellung der SICK AG

Die SICK AG ist einer der weltweit führenden Hersteller von industrieller Sensorik. Die Unternehmensstruktur setzt sich aus den drei Geschäftsfeldern Prozess-, Fabrik- und Logistikautomation zusammen.

Das Unternehmen zählt weltweit ca. 5.000 Mitarbeiter, davon ca. 1.800 im Ausland. Neben dem Stammsitz in Waldkirch gehören weltweit über 50 Tochtergesellschaften in 30 Ländern zum Konzern. Im Geschäftsjahr 2009 erzielte die SICK AG einen Umsatz von 592,4 Mio. Euro und investierte 10,8% des Umsatzes in die Forschung & Entwicklung neuer Produkte und Technologien.

Die SICK AG bietet ein weitreichendes Produktportfolio für industrielle Automatisierungsaufgaben. Hierzu zählen Laserscanner, Lichtschranken und Kamerasysteme, wie Bild 1 zeigt, aber auch Barcode- und RFID-Scanner, Sicherheitsschalter, Emissionsmesstechnik und viele weitere spezialisierte Sensortypen [4].

Bild 1: Produktauswahl (Quelle [4]).

1.2. SOPAS

Die „SICK Open Platform for Applications and Systems“ (SOPAS) ist eine Software-Plattform, mit der verschiedene Sensor-Produkte konfiguriert, parametrisiert und visualisiert werden können. Diese Plattform stützt sich auf die XML-Beschreibung der Sensoren und darauf aufbauendem generischen Code, die Vereinheitlichung der Kommunikations- und Bussysteme, sowie auf das SOPAS Engineering Tool (ET).

Die Beschreibung der Sensoren basiert auf mehreren XML-Schemata. Zum Einen werden dort interne Datenstrukturen und Parameter eines Sensors mit der Communication Interface Description (CID) beschrieben, die in SOPAS ET visualisiert werden sollen. Für jeden Parameter ist die Beschreibung des Datentyps, der Zugriffsrechte und -bedingungen, sowie der Standardwerte vorgesehen. Zum Anderen wird dort die Visualisierung der Datenstrukturen und Parameter direkt beschrieben, z. B. für die Auswahl von Parametern mit einem Drop-Down-Menü. Des Weiteren bietet sich die Möglichkeit eigene, komplexe grafische Oberflächen in SOPAS ET zu integrieren, z. B. für die Konfiguration und Überwachung des Scan-Bereiches eines Laserscanners, wie das Dialogfenster in Bild 2 zeigt.

Die Beschreibungen der Visualisierung und der Parameter eines Sensors werden von einem Generator-Framework verarbeitet. Die Beschreibung der Visualisierung wird in eine Sensorenbibliothek eingefügt und steht somit SOPAS ET zur Verfügung. Die Parameterbeschreibung der CID wird als Firmware auf dem Sensor integriert, um SOPAS-Mechanismen bereits auf der Ebene des Betriebssystems bereitzustellen.

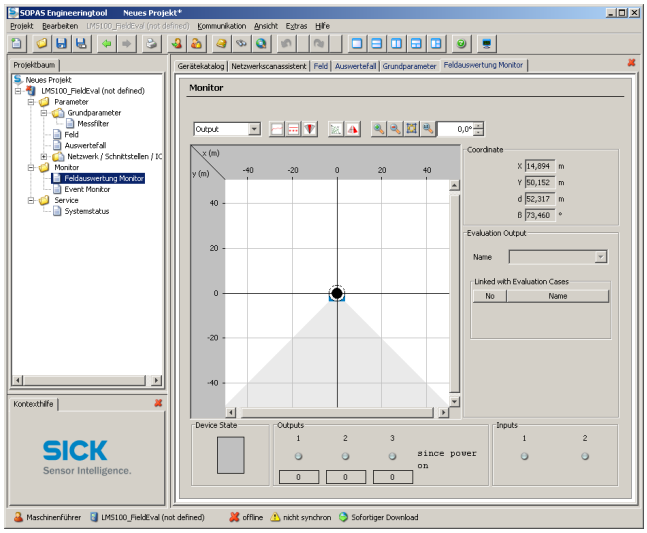


Bild 2: Oberfläche SOPAS ET mit Laserscanner.

Alle Bussysteme mit ihren Kommunikationsprotokollen sind mit Hilfe einer Schichtenarchitektur der SOPAS-Plattform gekapselt, wodurch die physikalischen Übertragungsschichten leicht austauschbar sind. Das Schichtenmodell vereinheitlicht die Kommunikation über diverse Bussysteme, wie CAN, PROFIBUS, IO-Link, Ethernet und USB.

1.3. Motivation

Der Einsatz hybrider Systeme, die aus Mikroprozessoren und FPGAs bestehen, hat mittlerweile eine große Verbreitung im Bereich eingebetteter Systeme erlangt. Dadurch lassen sich Vorteile beider Rechnertechnologien optimal nutzen. Während FPGAs als applikationsspezifische Co-Prozessoren z. B. für digitale Signalverarbeitung spezielle Aufgaben durch parallele Verarbeitung schneller und effektiver als Mikroprozessoren ausführen und zusätzlich den Vorteil der Taktgenauigkeit bieten, lassen sich Applikationen auf Mikroprozessoren mit Hilfe imperativer oder objektorientierter Programmiersprachen (C/C++) flexibel programmieren [1]. Produkte der SICK AG basieren oft auf einer hybriden Architektur, um insbesondere die Vorteile in der digitalen Signalverarbeitung durch den Einsatz von FPGAs ausnutzen zu können.

Die Entwicklung einer Middleware, wie eingangs beschrieben, würde die Integration der SOPAS-Plattform in hybriden Systemen erheblich vereinfachen. Bisher müssen die Parameterräume von Mikroprozessor und FPGA von Hand aneinander angepasst werden. Die Middleware soll das Bit-Mapping automatisieren und dem Entwickler eine flexible Beschreibung der FPGA-Speicherstrukturen über ein Beschreibungsschema ermöglichen.

2. Entwicklungsumgebung

Als Entwicklungsumgebung dienen zwei Plattformen mit zwei unterschiedlichen hybriden Mikroprozessor-FPGA-Architekturen. Die beiden Systeme werden in den folgenden Abschnitten näher erläutert.

2.1. Mikroprozessor-FPGA-System

Das Mikroprozessor-FPGA-System ist ein Sensor-Produkt der SICK AG, dass für die Messung von Füllständen in Flüssigkeitsbehältern verwendet wird. Das Sensor-Produkt hat eine hybride Mikroprozessor-FPGA-Architektur, in der der FPGA messtechnische und der Mikroprozessor rechenintensive Aufgaben bearbeitet. Die Entwicklungsplattform für die zu implementierende Aufgabenstellung besteht aus einem 32-Bit Mikroprozessor und einem FPGA. Beide Komponenten kommunizieren über eine vereinfachte serielle Schnittstelle, die an das Serial Peripherial Interface (SPI) angelehnt ist.

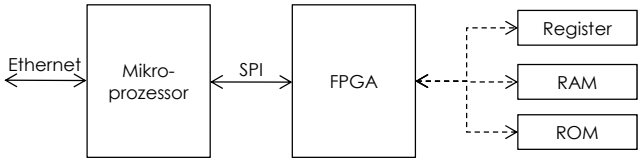


Bild 3: Hybride Mikroprozessor-FPGA-Architektur.

2.2. Soft-Core-Prozessor-FPGA-System

Die zweite Entwicklungsplattform ist der Prototyp einer Zeilenkamera. Der Bildgeber der Zeilenkamera ist ein Halbleiterdetektor der nach CMOS-Technologie gefertigt wird. Detektoren diesen Types werden als Active Pixel Sensor (APS) [10], oder vereinfacht als CMOS-Bildsensor, bezeichnet. Wie bei der konkurrierenden Technologie des Charge-coupled Device (CCD)-Chips [10], werden photoelektrische Dioden zur Messung der Lichtintensität verwendet. Zeilenkameras unterscheiden sich von herkömmlichen digitalen Kameras in der Anordnung der photoelektrischen Dioden auf der Sensoroberfläche. Bei herkömmlichen Digitalkameras sind die Photodioden in Form einer

Matrix auf der Sensoroberfläche aufgebracht, während sie bei Zeilenkameras in einer Reihe angeordnet sind. Zur Bilderzeugung ist daher bei Zeilenkameras entweder die Bewegung der Kamera oder des Objektes notwendig. Typischerweise werden Zeilenkameras zur Inspektion von Oberflächen eingesetzt.

Im Unterschied zu der hybriden Architektur des Füllstandsensors, besteht die Architektur der Zeilenkamera aus einem einzelnen Xilinx Spartan-3A DSP FPGA [8], der als Soft-Core-Prozessor den Xilinx-eigenen MicroBlaze [7] enthält. Neben dem MicroBlaze beinhaltet der FPGA weitere Komponenten, wie Register, Bildverarbeitungsalgorithmen und Sensorik.

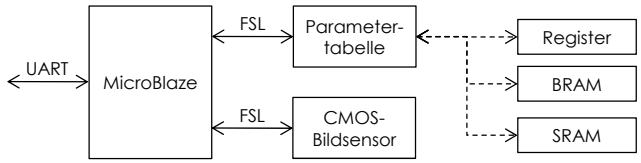


Bild 4: Soft-Core-FPGA-Architektur.

Somit ergibt sich ebenfalls eine implizite Mikroprozessor-FPGA-Architektur, die allerdings physikalisch auf demselben FPGA implementiert ist. Diese Architektur wird als System-on-a-Chip (SoC) oder Ein-Chip-System bezeichnet. Die Kommunikation zwischen dem MicroBlaze und den restlichen Komponenten des FPGAs erfolgt chipintern über den Fast Simplex Link (FSL)-Bus [9].

3. Design

3.1. Architektur

Das Design der Middleware basiert auf der in Bild 5 dargestellten Architektur. Die vorhandene Anbindung von Sensoren an einen Mikroprozessor wird über die CID generiert. Über den Mikroprozessor sollen nun nachgelagerte FPGAs an SOPAS ET angebunden werden. Die C- und VHDL-Komponenten für Mikroprozessor und FPGA werden aus einem gemeinsamen Beschreibungsformat, der *Digital Chip Interface Description* (*DCID*) generiert. Hierzu werden auf beiden Seiten diverse Schnittstellen (*Access Protocol*) für die gegebene physikalische Schicht erzeugt. Auf Seiten des Mikroprozessors werden die beschriebenen Speicherstrukturen des FPGAs über generierte Funktionen und Makros gekapselt. Auf der Seite des FPGAs wird eine *Parametertabelle* generiert, die eine einheitliche Schnittstelle für die untergeordneten Speicherstrukturen, wie Register, RAM und ROM, bereitstellt. Die Kommunikation zwischen Mikroprozessor und FPGA besitzt eine Master-Slave-Architektur, in dem der Mikroprozessor die Kommunikation als Master steuert.

3.2. Datentypen

Um die Code-Generierung auf Seiten des Mikroprozessors und des FPGAs konzeptuell planen zu können, wurden zunächst die in der SOPAS-Plattform verwendeten Basis-Datentypen analysiert und für ihre Eignung sowohl für die VHDL-, als auch die C-

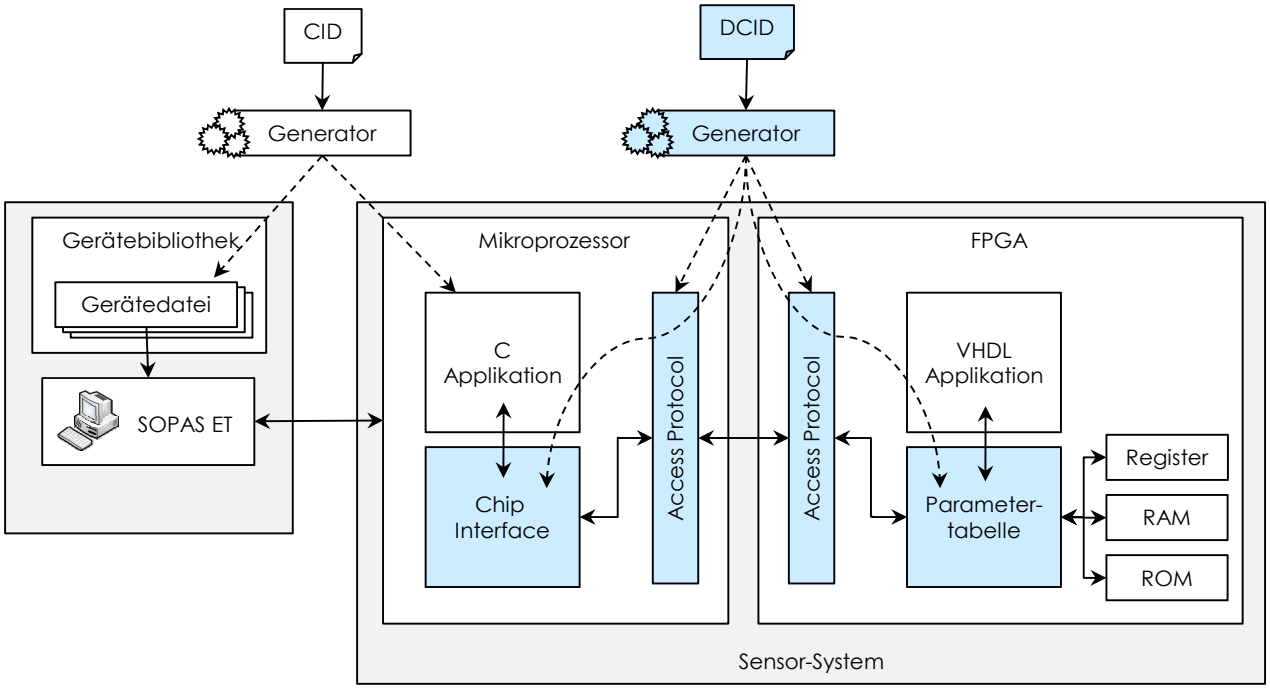


Bild 5: Schematische Architektur der Middleware.

Implementierung der Middleware bewertet. Da die Middleware als Teil der SOPAS-Plattform entwickelt wird und zur Visualisierung der Parameter des FPGAs auf bestehenden SOPAS-Mechanismen basiert, müssen die Datentypen der Middleware zu den Datentypen der SOPAS-Plattform kompatibel sein. Die SOPAS-Plattform unterstützt alle aus Hochsprachen bekannten Datentypen (int, float, char, etc.) zur Definition der Parameter und zur Modellierung der internen Datenstrukturen des Mikroprozessors. Die Analyse ergab, dass für den Mikroprozessor vorzeichenlose und vorzeichenbehaftete Integer-Datentypen, sowie der boolesche Datentyp relevant sind. Die Parameter können mit den Datentypen Int, UInt und Bool definiert werden. Innerhalb der VHDL-Implementierung werden alle Parameter als Bitvektoren (std_logic_vector) angelegt. Die Übersetzung der Bitvektoren in geeignete Datentypen erfolgt auf der Seite des Mikroprozessors.

3.3. Beschreibungsschema

Das *DCID*-Schema wurde für die Entwickler von Hardwarekomponenten konzipiert, die bislang wenig mit SOPAS-Mechanismen arbeiten. Es wird daher losgelöst von bisherigen SOPAS-Schematas als eigenständiges Schema umgesetzt und soll nach Möglichkeit der Beschreibung von Hardware nahe kommen. Das Schema wird nach den Empfehlungen des World Wide Web Consortiums (W3C) [6] als XML-Schema umgesetzt.

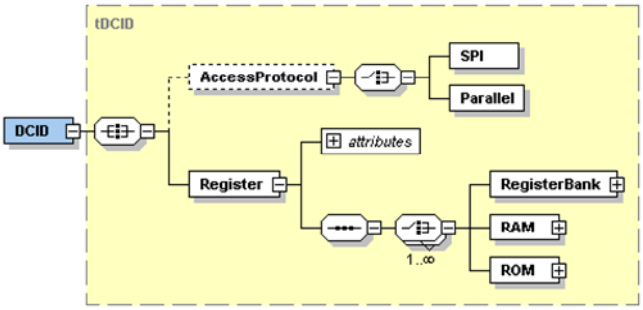


Bild 6: DCID-Schema.

Das *DCID*-Schema gliedert sich in die Beschreibung des *Access Protocols* und der FPGA-internen Speicherstrukturen. Die Speicherstrukturen können als *Registerbank*, *RAM* oder *ROM* beschrieben werden, wie Bild 6 verdeutlicht. Die Beschreibung einer Speicherbank beinhaltet primär den Typ und die Dimensionen des physikalischen Speichers. Für die spätere Adressierung aller Speicherelemente kann für jede Speicherbank eine Startadresse vergeben werden. Listing 1 zeigt die Beschreibung verschiedener Speicherstrukturen exemplarisch.

```
<RegisterBank Name="Registerbank 1"
        MemorySize="16" WordSize="8">
    <!-- Variablenbeschreibung -->
</RegisterBank>
<RAM Name="DRAM1" MemorySize="32" WordSize="8">
    <!-- Variablenbeschreibung -->
</RAM>
```

Listing 1: Beschreibung Speicherbänke.

Innerhalb einer Speicherbank werden *Variablen* und *Funktionen* (*Function*) beschrieben, die abschließend in SOPAS ET dargestellt werden sollen. Zur Typdefinition können die Datentypen Int, UInt und Bool verwendet werden. Weiterhin können mit diesen Datentypen komplexe Strukturen und Arrays modelliert werden. Die Parameter lassen sich zusätzlich in der Bitbreite, sowie mit Zugriffsrechten einschränken und mit einer festen Adresse innerhalb der Speicherstruktur definieren.

3.4. Code-Generator

Das bestehende SOPAS-Generator-Framework wird um die Verarbeitung des erstellten *DCID*-Schemas erweitert. Die im Schema beschriebenen Speicherstrukturen werden zunächst auf eine interne Datenstruktur des Generators abgebildet. Nach der Prüfung der Speicherbänke auf Gültigkeit, Integrität und Adressierung werden C- und VHDL-Module nach einer Konfiguration erzeugt.

4. Implementierung

4.1. Mikroprozessor

Der Anteil der Middleware, der auf dem Mikroprozessor ausgeführt werden soll, besteht aus einer Parametertabelle und einer Zugriffsschnittstelle auf den FPGA, dem *Chip Interface* (vgl. Bild 5). Die Parametertabelle des Mikroprozessors wird in Form von konstanten Makros aufgebaut. Die Makros bilden die Adresse, den Datentyp, die Breite und die Zugriffsmaske eines Parameters ab. Eine Maskierung von Parametern wird notwendig, wenn bspw. auf einzelne Bits eines Bit-Vektors zugegriffen werden soll. Der Zugriff auf die Parameter und damit auf den entfernten FPGA erfolgt über parametrisierte Makros, die als „get“- und „set“-Makro zu jedem Parameter generiert werden. Bild 7 zeigt die Abbildung von einfachen und komplexen Schema-Elementen in C-Quellcode.

Mit dem Typ *Function* der DCID lassen sich objektorientierte Zugriffsfunktionen realisieren, die den direkten Zugriff auf einzelne Parameter kapseln. Bild 8 zeigt die generierte High-Level-Funktion „*kocheKaffee()*“. Für eine *Function* können Übergabe- und Rückgabeparameter definiert werden. Beispiel-

weise kann ein Übergabeparameter in ein Statusregister geschrieben werden, um eine Aktion zu starten. Ein Rückgabeparameter wird aus einem Zustandsregister gelesen, um einen Zustandswechsel bzw. das Ende der gestarteten Aktion anzuzeigen und die Funktion zu terminieren.

```
<Variable Name="TassenZaehler">
    <Int Width="16" DefaultValue="0"/>
</Variable>
```

```
// Simple Variable
#define TassenZaehler             0x04F6
#define TassenZaehler_Mask        0x0001
#define setTassenZaehler(x)       WriteFpga(x, 0x04F6, 0x0001)
#define getTassenZaehler()        ReadFpga(0x04F6, 0x0001, &ptr)
```

```
<Variable Name="Steuerung">
  <Struct>
    <Int ElementName="Tassengroeße" Width="8" DefaultValue="200"/>
    <UInt ElementName="Bruehdauer" Width="8" DefaultValue="50"/>
    <Bool ElementName="Entkalken" DefaultValue="false"/>
    <Bool ElementName="Bereit" DefaultValue="true"/>
  </Struct>
</Variable>
```

```
// Complex Variable
#define Steuerung         0x04F6
#define Steuerung_Mask    0x0001

typedef struct Steuerung {
         Int Tassengroeße,
         UInt Bruehdauer,
         Bool Entkalken,
         Bool Bereit
};

#define setSteuerung(x)   WriteFpga(x, 0x04F8, 3)
#define getSteuerung()    ReadFpga(0x04F6, 0x03FC, &ptr)
```

Bild 7: Exemplarische Makros für Parameter.

```
<Function Name="kocheKaffee">
     <Parameters>
          <Value Name="Staerke">
               <Int DefaultValue="1" Width="2"/>
          </Value>
          <Value Name ="Tassenanzahl">
               <Int Width="3" DefaultValue="1"/>
          </Value>
     </Parameters>
     <ReturnValues>
          <Value Name="Fehler">
               <Bool></Bool>
          </Value>
          <Value Name="Abgeschlossen">
               <Bool></Bool>
          </Value>
     </ReturnValues>
</Function>
```

```
// Abstract function; Name = "kocheKaffee'
typedef struct kocheKaffee_ReturnStruct {
    Bool Fehler,
    Bool Abgeschlossen,
};

kocheKaffee_ReturnStruct kocheKaffee(Int Staerke, Int
Tassenanzahl)
{
    kocheKaffee_ReturnStruct retVal;
    // buffer declaration ...
    WriteFPGA(Staerke, 0x0270, 0x0001);
    while (ReadFpga(0x0271, 0x0001, &retVal.Fehler) == -1);
    WriteFPGA(Tassenanzahl, 0x0272, 0x00FC);
    while(ReadFpga(0x0273, 0x0001, &retVal.Abgeschlossen) == -1);
    return retVal;
}
```

Bild 8: Exemplarische Makros für Funktionen.

Neben den generierten Zugriffsfunktionen auf definierte Adressen, bietet das *Chip Interface* Funktionen an, mit denen frei wählbare Adressbereiche vom FPGA gelesen werden können. Der konkrete Zugriff wird vom *Access Protocol* umgesetzt. Für die Übertragung von größeren Datenmengen, wie bspw. beim Lese-Zugriff auf eine Struktur, werden die Elemente nach der Größe des Datenbusses einzeln übertragen.

4.2. FPGA

Der Middleware-Teil des FPGAs wird als kompatibles Gegenstück zum Mikroprozessor modelliert. Bild 9 zeigt die Architektur der FPGA-Komponenten. Analog zum Mikroprozessor wird die physikalische Schicht mit der *Parametertabelle* verbunden. Für den Zugriff auf die physikalische Schicht wird eine Wrapper-Funktion generiert, die vom Entwickler erweitert werden muss. Für die parallele Datenübertragung und SPI wird der komplette Zugriff implementiert. Das Interface auf die gekapselten Speicherstrukturen ist für Register, RAM- und ROM-Strukturen identisch und beinhaltet die Adressauswahl, den Datenbus, sowie einen Handshake-Mechanismus, der den Zugriff auf die Speicher-strukturen steuert und die Datenübernahme anzeigt.

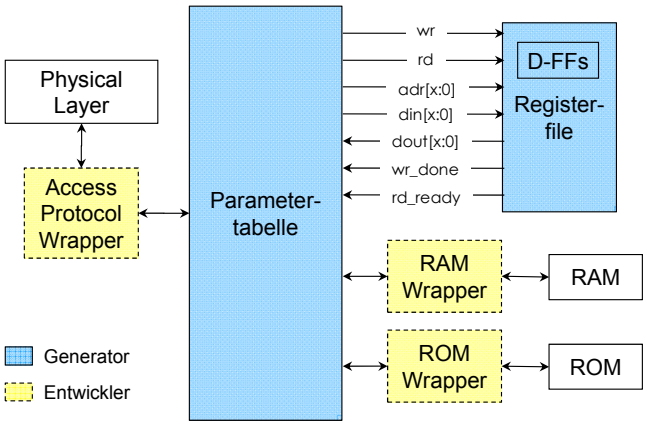


Bild 9: FPGA Komponenten.

Die *Parametertabelle* stellt eine einheitliche Zugriffsschnittstelle auf alle untergeordneten Speicherelemente zur Verfügung. Sie verwaltet die Adressen der Speicherbänke, sowie deren Parameter. Die Adressierung der *Parametertabelle* ist in Speicherbänke und interne Parameter gegliedert. Etwaige Zugriffsanforderungen leitet die *Parametertabelle* als Adress-Dekoder an die entsprechende Speicherbank weiter, wie Bild 10 zeigen soll. Das generische Registerfile besteht aus einer Menge instanziierter D-Flipflops, die sich aus den Dimensionen der Registerbank zusammensetzt.

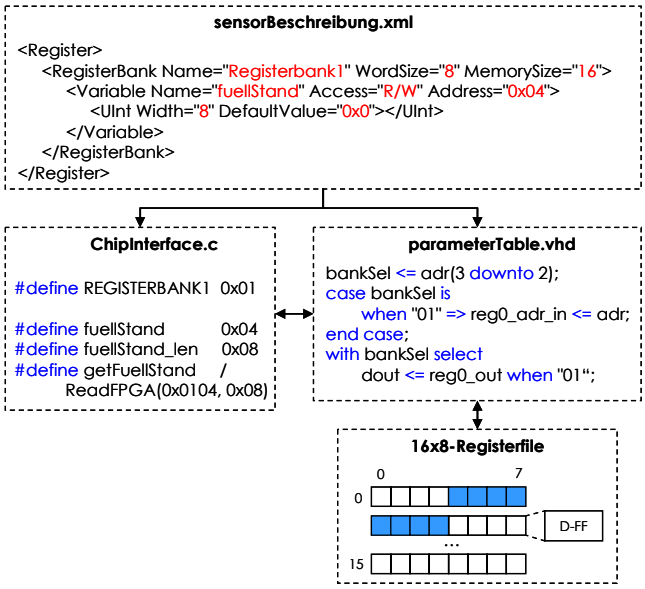


Bild 10: Adressierung der Speicherstrukturen.

Die im *DCID*-Schema beschriebenen Parameter werden in der VHDL-Implementierung primär als Adressen und Bitfolgen in einer Speicherbank behandelt. Eine Zuordnung der im Beschreibungsschema vergebenen Namen zur jeweiligen Adresse erfolgt zunächst nicht. Über die Konfiguration des Code-Generators kann ein Interface erzeugt werden, über das weitere VHDL-Applikationen, mit dem Namen des Parameters, Zugriff auf die Speicherbänke nehmen können.

5. Fazit

5.1. Zusammenfassung

Der vorliegende Lösungsansatz bietet den internen Entwicklungsabteilungen und den Kunden der SICK AG einen großen Mehrwert. Bereits in der Planungsphase einer neuen Entwicklung auf ASIC- oder FPGA-Basis kann die Middleware mit einbezogen werden und erfordert kaum Anpassungen, um zwischen digitalem Baustein, Mikroprozessor und insbesondere SOPAS ET zu kommunizieren.

5.2. Ausblick

Neben der implementierten Funktionalität der Middleware könnten weitere Bussysteme, wie z. B. On-Chip-Busse unterstützt werden. Weiterhin könnten die umgesetzten FPGA-Komponenten genutzt werden, um SOPAS ET direkt, z. B. über IO-Link oder Ethernet, mit einem FPGA-System zu verbinden.

Das Beschreibungsschema könnte so erweitert werden, dass die detaillierte Architektur des FPGA-Systems beschrieben werden und somit bspw. Übergänge von Taktdomänen synchronisiert werden könnten.

6. Literatur

[1] R. Gessler; T. Mahr: *Hardware-Software-Codesign – Entwicklung flexibler Mikroprozessor-FPGA-Hochleistungssysteme*. 1. Auflage, Vieweg-Verlag, Wiesbaden, 2007.

[2] IEC; IEEE: *Behavioural languages - Part 1-1: VHDL Language Reference Manual, IEEE 1076*. 2004.

[3] J. Reichardt; B. Schwarz: *VHDL-Synthese - Entwurf digitaler Schaltungen und Systeme*. Oldenbourg, München, 2007.

[4] SICK AG: *Diverse Quellen: Geschäftsbericht, Internet-Auftritt, Unternehmensportrait, Produktkatalog*. Waldkirch, 2010.

[5] J. Teich; C. Haubelt: *Digitale Hardware/Software-Systeme - Synthese und Optimierung*. 2. Auflage, Springer, Berlin 2007.

[6] W3C: *XML Schema Part 0: Primer - Recommendation*. 2004.

[7] Xilinx Inc.: *MicroBlaze Processor Reference Guide*. v9.0, 2008.

[8] Xilinx Inc.: *Spartan 3A DSP FPGA Family Datasheet*. 2009.

[9] Xilinx Inc.: *LogiCORE IP Fast Simplex Link (FSL) V20 Bus*. 2010.

[10] H. Zimmermann: *Integrated Silicon Optoelectronics*. Bd. Springer Series in Optical Sciences, 2nd Edition, Springer, Berlin, 2009.

Dynamische partielle Rekonfigurierung eines Xilinx Virtex-5 FPGAs

Marco Scheffler, Frank Kesel

Hochschule Pforzheim, 75175 Pforzheim, Tiefenbronner Straße 65

Tel.: 07231/28-6065, e-mail: {marco.scheffler, frank.kesel}@hs-pforzheim.de

Abstract

Als Rekonfigurierung wird der Vorgang einer Funktionsänderung eines programmierbaren Logikbausteins durch Manipulation des Konfigurationsspeichers bezeichnet. Xilinx bietet bei seinen Virtex FPGAs die Möglichkeit, anstatt einer Gesamt- lediglich eine *Teilrekonfigurierung* der Logikressourcen *während der Laufzeit* des Bausteins durchzuführen. Der nichtrekonfigurierte (statische) Teil behält während des Vorgangs seine volle Funktionalität bei. Somit kann die Rekonfigurierung als dynamisch und partiell erfolgend charakterisiert werden. Weiterhin ermöglichen Xilinx Virtex Derivate einen Zugriff auf den SRAM-Konfigurationsspeicher „on-chip"; womit die Logik im statischen Bereich (vorwiegend ein eingebettetes µP-System) einen Rekonfigurationsvorgang selbst initiieren und steuern kann.

Durch die Anwendung der dynamischen partiellen Rekonfigurationsmethode erhalten FPGA basierte Systeme eine neue zeitliche Dimension. Sich wechselseitig ausschließende Logikfunktionen, die nicht zum gleichen Zeitpunkt in der funktionalen Ebene realisiert sein müssen, können nach Bedarf temporal ausgetauscht werden.

Dieser Beitrag beleuchtet Grundlagen, Vorteile und mögliche Zielapplikationen von Systemen, die dynamisch partiell rekonfigurierbar sind. Weiterhin wird die Anwendung dieser Methode in einem signalverarbeitenden System demonstriert, welches mittels eines Hardwarebeschleunigers über eine breitbandige DDR2-SDRAM Anbindung eine quasi optimale Rekonfigurationsperformanz erzielt.

Schlüsselwörter: FPGA Rekonfigurierung, dynamisch, partiell, run-time reconfiguration, partial reconfiguration, reconfigurable computing

1. Einleitung

Field Programmable Gate Arrays (FPGAs) ermöglichen dem Ingenieur die Entwicklung von hochflexiblen digitalen Lösungen. Obwohl FPGAs wegen ihres "field-programmable" Charakters bereits eine hohe inhärente Flexibilität aufweisen, nutzen viele Applikationen nicht den vollen Umfang der Möglichkeiten. Im Allgemeinen werden FPGAs beim Start der Applikation konfiguriert und in diesem spezifischen Konfigurationszustand während der gesamten Dauer der Anwendung belassen.

Der FPGA Hersteller Xilinx bietet in seinen High-End Bausteinen Virtex seit mehreren Produktgenerationen eine Möglichkeit, diese FPGAs während der Laufzeit partiell zu rekonfigurieren. Hierbei wird nur ein Teil der Logikressourcen auf dem Chip rekonfiguriert, der nichtrekonfigurierte, im folgenden statisch genannte Teil behält seine volle Funktionalität und Logikzustände bei. Außerdem bieten Virtex FPGAs neben den dedizierten Bausteinpins einen „on-chip" Zugang zum Konfigurationsspeicher, den Internal Configurations Access Port (ICAP). Somit ist die im statischen Bereich angesiedelte Logik in der Lage, im Vorfeld generierte Konfigurationsdaten (*partial bitstreams*) von einem Datenspeicher zum ICAP zu transferieren, und somit eine dynamische partielle Rekonfigurierung (DPR) durchzuführen.

Durch Einbeziehung dieser Rekonfigurationsmethode ist mit DPR-fähigen FPGAs eine Ein-Chip-Lösung realisierbar, welche dem Entwickler einen zusätzlichen - zeitlichen - Freiheitsgrad zur Verfügung stellt. Die ursprünglich zweidimensionale Entwurfs*fläche*, begrenzt durch die Logikressourcen des Bausteins, wird durch eine temporale Achse quasi zu einem Entwurfs*raum* erweitert. Funktionale Blöcke, welche in ihrem Systemkontext nicht zwingend gleichzeitig auf dem Chip vorhanden sein müssen, können der momentanen Anforderung entsprechend geladen bzw. getauscht werden. Der zusätzliche Freiheitsgrad bedeutet eine immense Erhöhung der Flexibilität des Bausteins gegenüber nicht DPR-fähigen FPGA-Applikationen. Diese können höchstens durch eine extern gesteuerte Gesamtrekonfigurierung neu geladen werden, mit allen bereits erwähnten Nachteilen (zusätzlicher Controllerbaustein, Verlust aller Logikzustände).

Bisherige Arbeiten zu diesem Thema finden sich vor allem unter dem übergeordneten Forschungsparadigma des „Rekonfigurierbaren Rechnens“ (*reconfigurable computing*), welches zum Ziel hat, die Hardwareplattform eines digitalen Systems adaptiv an die momentan zu berechnenden Algorithmen anzupassen. Hierdurch lassen sich gegenüber reinen GPP Architekturen Verbesserungen bezüglich leistungs-, flächen-, und energiespezifischen Parametern erreichen [1–4].

Nahezu alle bisherigen Forschungsarbeiten im Bereich DPR wurden auf Virtex-II oder Virtex-4 Derivaten durchgeführt, z.B. in [5–7]. In diesem Beitrag wird die Methode mit einem Virtex-5 Baustein auf einem Xilinx ML501 Eval-Board implementiert.

2. Techn. Realisierung einer DPR

2.1. Modulbasierter Ansatz

Um ein DPR-fähiges System zu entwickeln, stellt Xilinx einen modulbasierten EDA-Toolflow zur Verfügung. Dieser Toolflow ergänzt die üblichen EDA-Tools wie ngdbuild, par oder bitgen um zusätzliche Optionen, die es dem Entwickler ermöglichen, neben dem initialen Gesamtbitstream zusätzlich partielle Bitstreams zu erstellen, die lediglich einen Teil der funktionalen Ebene rekonfigurieren.

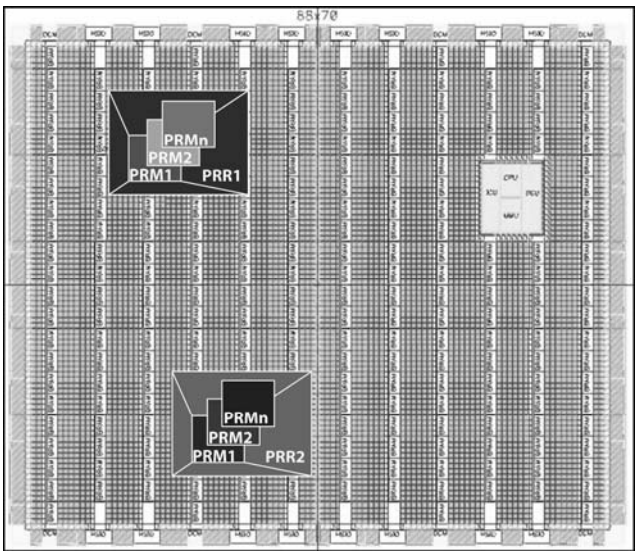


Bild 1: Schematische Darstellung der DPR auf einem FPGA (Xilinx Virtex-4). PRR1 und PRR2 sind partiell rekonfigurierbare Regionen, welche jeweils mit verschiedenen partiell rekonfigurierbaren Modulen (PRM) geladen werden können. Der Bereich außerhalb der PRRs ist der nichtrekonfigurierbare statische Bereich. (Bildquelle: Xilinx)

Rekonfigurierbare funktionale Blöcke werden als Module (PRM für Partial Reconfigurable Module) in rekonfigurierbare Regionen (PRR für Partial Reconfigurable Region) geladen. Größe und Platzierung einer PRR bilden somit die logischen Randbedingungen für die zu implementierenden Module. Die Anzahl an PRMs je PRR ist nicht beschränkt. Mit Hilfe des Tools PlanAhead kann der Entwickler die PRRs auf der Ebene des FPGAs mit den erwähnten Randbedingungen platzieren und die PRMs entsprechend zuweisen.

Der modulbasierte Realisierungsansatz ermöglicht eine ressourcenschonende Implementierung von funktionalen Blöcken, die sich wechselseitig ausschließen (*„mutually exclusive“*). Ist also im Anwendungskontext des Systems aus einer Auswahl an Funktionen zu einem beliebigen Zeitpunkt nur eine aktiv, so können die Funktionen als PRMs in eine PRR abgebildet werden und dynamisch rekonfiguriert werden.

2.2. Floorplan Budgeting

Der modulbasierte Ansatz erfordert die Einhaltung gewisser Randbedingungen bei der Platzierung der PRRs auf der Ebene des Virtex-5 Bausteins (*floorplan budgeting*). Diese Randbedingungen sind aus der Konfigurationstechnologie des Bausteins abgeleitet.

Die Rekonfigurierung eines Virtex-5 FPGAs kann mit dem EDA-Toolflow nicht mit beliebig kleiner Granularität erfolgen. Die kleinste adressier- und somit rekonfigurierbare Einheit ist ein sog. Frame (siehe Bild 2). Ein Frame besteht immer aus 41 32-bit Wörtern und definiert in der funktionalen Ebene die Logik eines Teils einer Spalte (column) in einer Reihe (row). Je nach Logikressource wird eine unterschiedliche Anzahl an Frames benötigt, um eine komplette Spalte zu rekonfigurieren. Für eine CLB-Spalte (d.h. 20 CLBs) sind z.B. 36 Frames nötig, bei Rekonfiguration einer BRAM-Spalte (4 BRAM-Blöcke) 158 Frames (30 zur Verbindungs- und 128 zur Inhaltskonfiguration). Aus Sicht des Anwenders ist die kleinste rekonfigurierbare Einheit quasi ein Block aus solchen Frames. Die bit-genaue Abbildung von Konfigurationsdatenbits auf die entsprechende Logikressource in der funktionalen Ebene ist von Xilinx nicht publiziert.

Die Abmessungen und Platzierung der PRRs sollte den Konfigurationseigenschaften des Bausteins angepasst werden, um eine optimale Nutzung der Fläche und möglichst kleine partielle Bitstreams zu erzielen. Das Tool bitgen ist bei der Erstellung der partiellen Bitstreams in der Lage, eine Datenkompression (*bitstream compression*) durchzuführen, wenn der Inhalt mehrerer

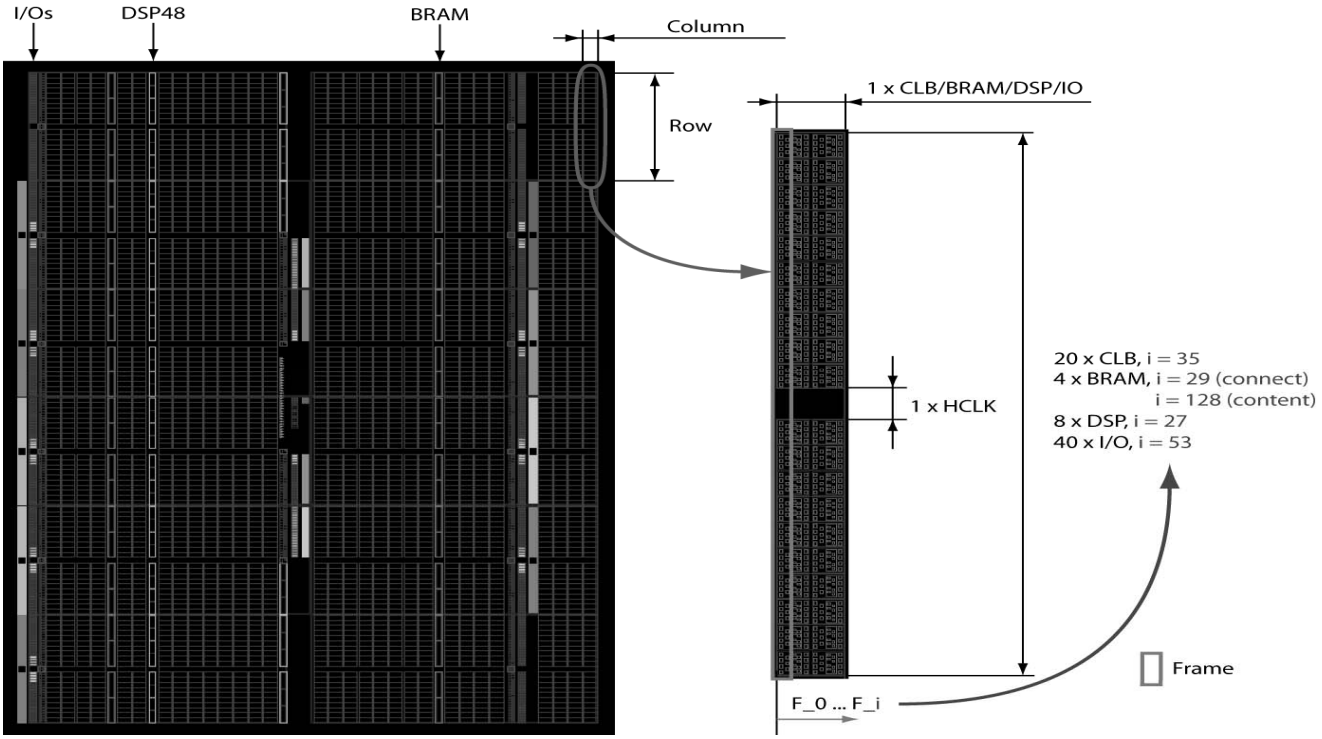


Bild 2: Visuelle Darstellung der funktionalen Ebene eines Virtex-5 LX50 FPGAs (aus PlanAhead). Aus Sicht der Konfigurationsebene ist die funktionale Ebene eine Fläche bestehend aus 6 Reihen (Rows), welche in 37 Spalten (Columns) eingeteilt werden. Je nach Logikressource wird ein solcher Block (hier als Beispiel ein CLB Block) in eine unterschiedliche Anzahl von senkrechten Frames (F_0 usw.) unterteilt. BRAMs werden zweimal indiziert, für die Verbindungs- und die Inhaltskonfiguration.

(nacheinander indizierbarer) Frames identisch ist. Im Allgemeinen sind die Bitstreams nach Fertigstellung also kleiner als vorab mathematisch berechnet. Hier liefert Xilinx ebenfalls wenig Dokumentation, so dass die Einsparungen durch die Datenkompression nicht deterministisch vorhergesagt werden kann.

2.3. Busmacros

Signale, die vom statischen in den dynamischen Bereich gelangen (oder vice versa), müssen über sogenannte Busmacros (bei Virtex-5 im Speziellen: Single Slice Macros) geführt werden. Busmacros sind Hardmacros, welche im .nmc Format vorliegen, vom Entwickler in VHDL instanziiert und im Rahmen des Floorplannings mittels eines LOC-Constraint (Location) in einer Slice innerhalb einer PRR platziert werden müssen. Diese wahlweise Platzierung innerhalb einer PRR ist möglich, da der Virtex-5 Baustein eine Rekonfigurierung „glichless“ garantiert (in statischen Verbindungen innerhalb einer PRR treten bei Rekonfiguration des SRAM-Speichers mit denselben Bitwerten (0->0, 1->1) keine metastabilen Zustände auf, deshalb dürfen statische Verbindungen vom Route Tool in und sogar komplett durch eine PRR geführt werden). Die Slices mit den platzierten Busmacros werden von den Tools als definierte Signalübergänge zwischen statischem und dynamischem Bereich erkannt. Busmacros stehen in mehrere Ausführungen zur Verfügung: synchron/asynchron, mit/ohne enable-Ports. Je Single Slice Macro können vier Signale (da vier Register/Slice) den dynamischen Bereich betreten oder verlassen.

2.4. ICAP

Das interne Konfigurationsinterface ICAP (internal configuration access port) ist vom SelectMAP abgeleitet, weshalb zeitliche und bitspezifische Parameter (z.B. Bitreihenfolge) übernommen werden können.

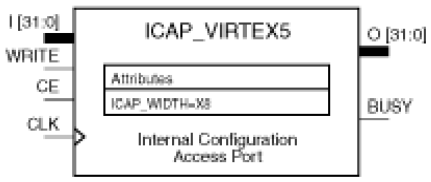


Bild 3: ICAP Interface Primitiv

Der Schreibbus des ICAP Interface kann mit einer Breite von bis zu 32-bit parametrisiert werden. Nach Datenblatt [8] ist eine Taktrate von 100 MHz spezifiziert, was einer max. Kanalkapazität von 400 MB/s[1] entspricht.

3. Mögliche Zielapplikationen

3.1. Charakterisierung

Betrachtet man unterschiedliche Implementierungsvarianten für digitale Prozesse, so unterscheiden sich diese in mehreren grundlegenden Metriken (Bild 4, [1]).

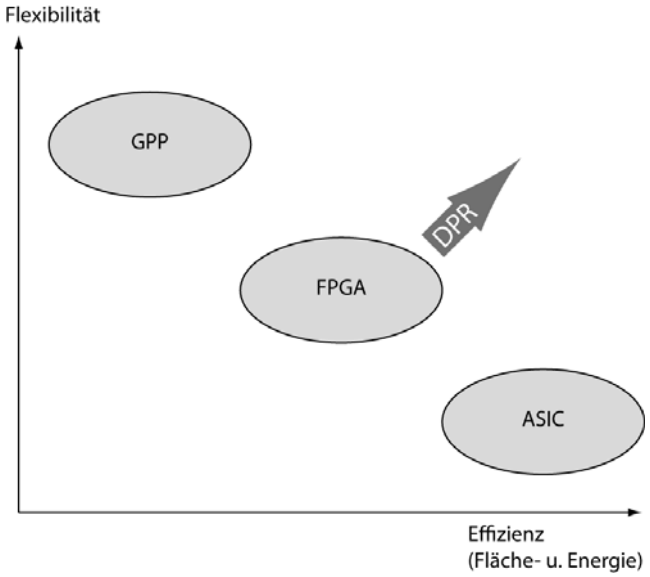


Bild 4: Flexibilität heutiger Implementierungsvarianten für digitale Prozesse in Abhängigkeit ihrer Energie- und Flächeneffizienz.

Universalprozessoren (General-Purpose-Processors, GPP) benötigen wegen ihrer rein sequentiellen Funktionsweise eine hohe Taktung, was mit einem gesteigerten dynamischen Energieverbrauch korreliert. Betrachtet man zusätzlich das Verhältnis zwischen Rechenleistung und dem dafür notwendigen Die-Flächenbedarf, so ist dies beim GPP ebenfalls sehr gering. Im Gegensatz dazu steht ein applikations-spezifischer Baustein (Application Specific Integrated Circuit, ASIC). Dieser benötigt für dieselbe Anzahl von Operationen ein Vielfaches weniger an Energie und Fläche. Selbstverständlich unterscheiden sich GPP und ASIC deutlich in ihrer Flexibilität. Ein GPP ist universell (re)programmierbar und somit prinzipiell bei allen Aufgaben in digitalen Prozessen anwendbar, wenn auch zu den bereits erwähnten Nachteilen. Den negativen Gegenpol bildet der ASIC: Speziell für eine Aufgabe entwickelt, gewährt er nur geringe Flexibilität und kann nicht an sich wechselnde Bedingungen angepasst werden. Dieser Vergleich kann über die funktionale Flexibilität hinaus fortgesetzt werden, indem auch die Entwicklungsflexibilität bewertet wird. So bestehen zwischen den beiden Varianten dieselben Differenzen bei Betrachtung der Realisierungsgeschwindigkeit von Produkten („time-to-market"), Baustein- bzw. Herstellerauswahl, Lieferzeit, Kosten und Entwicklungsaufwand.

In den Bereichen Leistungsaufnahme und Flächennutzung nimmt ein FPGA eine Position zwischen den beiden „Extremen" GPP und ASIC ein [1]. Auf Grund seiner nativen parallelen Arbeitsweise kann es gegenüber dem GPP Effizienzvorteile aufweisen, reicht aber nicht an einen applikationsspezifisch optimierten Baustein heran. Gegenüber diesem besitzt ein FPGA allerdings ein hohes Maß an Flexibilität - in allen oben erwähnten Facetten - und stellt mit diesem Kriterium eine Lösungsvariante dar, die die jeweiligen Vorteile von GPP und ASIC vereint.

Bezieht man nun das in diesem Beitrag vorgestellte Konzept der DPR mit in die Überlegungen ein, so erkennt man, dass der Grad an Flexibilität nochmals deutlich gesteigert werden kann. Zwar ist auch ohne DPR eine Rekonfiguration des FPGAs möglich, allerdings ist hierzu entweder eine spezielle Benutzerbedienung oder der Einsatz eines zusätzlichen Controllers erforderlich, der die Gesamtrekonfiguration (mit Verlust aller logischen Zustände) übernimmt. Nach Meinung des Autors ist auf Grund dieser Einschränkungen in der Vergangenheit das „field-programmable"-Prinzip eines FPGAs oft nicht ausnutzbar gewesen. FPAGs dienten oftmals lediglich als Evaluierungsbausteine, als Hilfsmittel während einer ASIC-Entwicklung oder als deren Ersatzbausteine, wenn der Kostenaufwand für ein ASIC nicht durch entsprechende Stückzahlen gedeckt war. Der Einsatz von DPR auf FPGAs hat mit Blick auf Bild 4 jedoch nicht nur Auswirkungen auf die Flexibilität. Module, die momentan nicht geladen sind, verbrauchen auch keine Energie und schließen damit weiter die Effizienzlücke zu einer Implementierung mit ASIC. Die Methode der DPR ist also in der Lage, Flexibilitätsvorteile des GPPs mit den Flächen- und Energievorteilen der ASIC-Implementierung in nicht geringem Maß in sich zu vereinen.

Mit den folgenden möglichen Zielapplikationen soll gezeigt werden, dass die Flexibilität, die ein FPGA mit Hilfe von DPR erreicht, Herstellern und Kunden Vorteile verschaffen kann, die eventuell mit anderen Technologien nicht derart realisierbar sind.

[1] alle Datenangaben mit SI-Präfix (1 MB = 1000 kB)

3.2. Update von IP-Cores über Netzwerk

Die Fähigkeit, während des Betriebs bestimmte Logikflächen des FPGAs zu rekonfigurieren, kann sowohl aus Kunden- wie auch als Herstellersicht interessant sein.

So bietet sich die Option, Rekonfigurationsdaten über ein Datennetzwerk zur Verfügung zu stellen, auf das ein entsprechend konfiguriertes FPGA mit Netzwerkverbindung zugreifen kann (Bild 5). Im Allgemeinen integriert das FPGA zu diesem Zweck ein µP-System, der entsprechende Kontroll-und Zugriffsroutinen ausführt. So könnte ein im Einsatz befindliches Gerät autonom ohne Zutun des Anwenders in seiner Logikfunktionalität aktualisiert werden, beispielsweise wenn ein IP-Core Lieferant ein Update (Fehlerbeseitigung, Funktionserweiterung) bisher gelieferter Cores anbietet oder wenn auf Kundenseite ein IP-Core Repository für mehrere FPGAs zentral verwaltet werden soll.

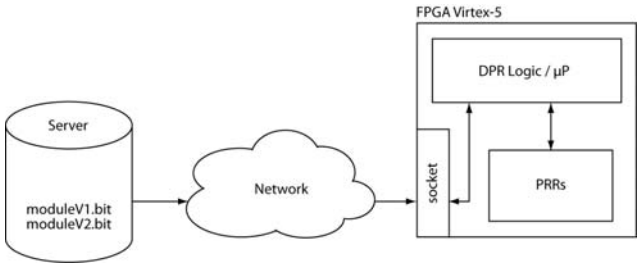


Bild 5: Mögliches Anwendungsszenario von DPR. Über eine Netzwerkverbindung lädt das FPGA Konfigurationsdaten für eine PRR von einem Server. Das FPGA kann sich so selbstständig aktualisieren, evtl. wenn der Hersteller Updates von IP-Cores anbietet oder wenn das FPGA in Einsatzumgebungen ohne möglichen menschlichen Zugriff betrieben wird (Raumfahrt o.ä.).

3.3. Software Defined Radio

Software Defined Radio (SDR) ist ein international vorangetriebenes Entwicklungsprojekt mit dem Ziel, Streitkräfte verschiedener (verbündeter) Länder mit den am jeweiligen Einsatzort nutzbaren Ressourcen in Kommunikation treten zu lassen. Zu diesem Zweck wird versucht, die integrierte digitale Signalverarbeitung der Geräte adaptiv an verschiedene - im Vorfeld definierte - schmal- und breitbandige „Wellenformen“ (Waveforms) anzupassen.

Der Einsatz von FPGAs ist bei SDRs nicht zwingend vorgesehen, die Signalverarbeitung kann auch auf einem GPP oder DSP implementiert werden (deshalb der Name „software defined“). Allerdings bietet sich der Einsatz von FPGAs in diesem Projektumfeld an, weshalb sie – gerade als partiell rekonfigurierbare Bausteine – das Interesse des Entwicklungskonsortiums geweckt haben.

3.4. Unterhaltungselektronik

Der Einsatz rekonfigurierbarer Hardware kann allgemein in der digitalen Signalverarbeitung von Vorteil sein, wenn größere algorithmische Blöcke auf Grund wechselnder Anforderungen oder Benutzereingaben getauscht werden können. Dass hierbei der Einsatz von DPR-basierenden FPGAs sogar im Unterhaltungselektronikbereich („Braune Ware“) sinnvoll sein kann, soll nun vorgestellt werden.

Die Unterhaltungselektronik war in den vergangenen Jahren auf Grund ihrer großen weltweiten Absatzzahlen fest von ASIC-Lösungen dominiert. Sowohl DVD-Player als auch erste Set-Top-Boxen (STB) für den Empfang von digitalen TV-Signalen (DVB-S/C/T) waren mit wenigen ASICs ausgestattet, da sowohl Videodaten auf einer DVD als auch im Transport –Stream der TV-Anbieter überwiegend im MPEG-2 Format (seltener in MPEG-1) kodiert waren, für welches schnell dedizierte ASICs (MPEG-2 Dekodierbausteine) entwickelt wurden. Ebenso waren die Tonspuren in beiden Domänen mit nur wenigen unterschiedlichen Standards kodiert.

Mit der Entwicklung der Blu-Ray- und HDTV-Technologie änderten sich die Bedingungen. Blu-Ray-Player müssen abwärtskompatibel zur DVD sein (MPEG-2) und weiterhin eine Vielzahl zwischenzeitlich neu entwickelter Kodierstandards unterstützen, die weiterhin ihrerseits noch in eine Vielzahl von Profilen und Qualitätslevel unterteilt sind. Ein ähnliches Bild zeichnet sich bei den HDTV-tauglichen STBs ab, die neben dem früheren MPEG-2 Signal die für die HDTV-Übertragung (DVB-S2) benutzet MPEG-4/AVC (H.264) Kodierung verarbeiten müssen.

Es lässt sich feststellen, dass die unterschiedlichen Marktteilnehmer anscheinend an einer erhöhten Flexibilität interessiert sind. Interessant ist, wie die Hersteller auf die Forderung reagieren, nämlich mit hochtaktenden GPP-Architekturen (bzw. Application Specific Instruction Processors, ASIP), die die notwendige Flexibilität durch Softwarebibliotheken zu erreichen versuchen. Die Hersteller Sony und Panasonic setzen momentan für ihre Produkte (z.B. Playstation 3) die eigenentwickelten Prozessorarchitekturen Cell respektive UniPhier ein.

Aus den bereits angesprochenen Gründen könnte ein FPGA mit DPR-Funktionalität durchaus in der Lage sein, die benötigte Flexibilität im Multimediabereich ebenfalls zur Verfügung zu stellen, und dies zu gesteigerten Effizienzwerten – sowohl in energetischer- als auch in rechenleistungsspezifischer Hinsicht. In einer üblichen STB könnten

beispielsweise viele sich ändernde Teilaufgaben der Demodulation, Kanaldecodierung (z.B. Forward Error Correction, FEC), Dechiffrierung (über das Conditional-Access-Modul) und den anschließenden Demultiplex- und Quellendekodierungsprozessen vom FPGA übernommen werden und dort effektiv berechnet werden. Auf Grund der üblicherweise über einen längeren Zeitraum persistenten Systemkonfiguration bei Multimediaanwendungen eignet sich das Prinzip der DPR hierfür sehr gut, da sich die insgesamt je Verarbeitungsschritt bevorrateten Funktionen per se wechselseitig ausschließen.

4. DPR Implementierung

4.1. HW-Architektur

In diesem Kapitel wird eine Implementierung eines dynamisch, partiell rekonfigurierbaren Systems vorgenommen, erläutert und analysiert. Die Gesamtfunktion des Systems ist hierbei nebensächlich, der Hauptaugenmerk liegt auf der Realisierung einer DPR Funktion und deren Auswirkungen auf das Gesamtverhalten.

Bild 6 zeigt ein mögliches Grundgerüst einer DPR Architektur mit zwei rekonfigurierbaren Regionen.

Es wird davon ausgegangen, dass die partiellen Bitstreams (und möglicherweise der Grundkonfigurationsbitstream) vom Entwickler auf einer Compact-Flash (CF) Karte abgespeichert sind, worauf der µProzessor (µP, hier Microblaze Softcore) über den SysACE Controller zugreifen kann. Selbstverständlich könnten alternativ auch andere nichtflüchtige Speicher (z. B. NOR-Flash) verwendet werden. In der vorliegenden Beispielarchitektur kann der Benutzer über den seriellen RS232 Port Steuereingaben tätigen sowie Statusausgaben überwachen.

Der Zugriff auf den ICAP-Port kann über zwei grundlegende Arten erfolgen. Xilinx bietet einen fertigen HWICAP-Core an, der den ICAP instanziiert (gestrichelte Verbindung) und über eine Slave-PLB Schnittstelle abstrahiert. Somit wird eine einfache softwarebasierte DPR möglich, die jedoch keine maximale Performanz bietet, da der PLB nicht mit jedem Taktzyklus einen Wert von der CF-Karte in den HWICAP transferieren kann. In der zweiten - hier implementierten - Variante wird der ICAP manuell über die Primitives-Libraries instanziiert und direkt über eine breitbandige Verbindung mit einem Datenspeicher gekoppelt. Die Übertragung wird von einem Hardwarebeschleuniger gesteuert (npi_if).

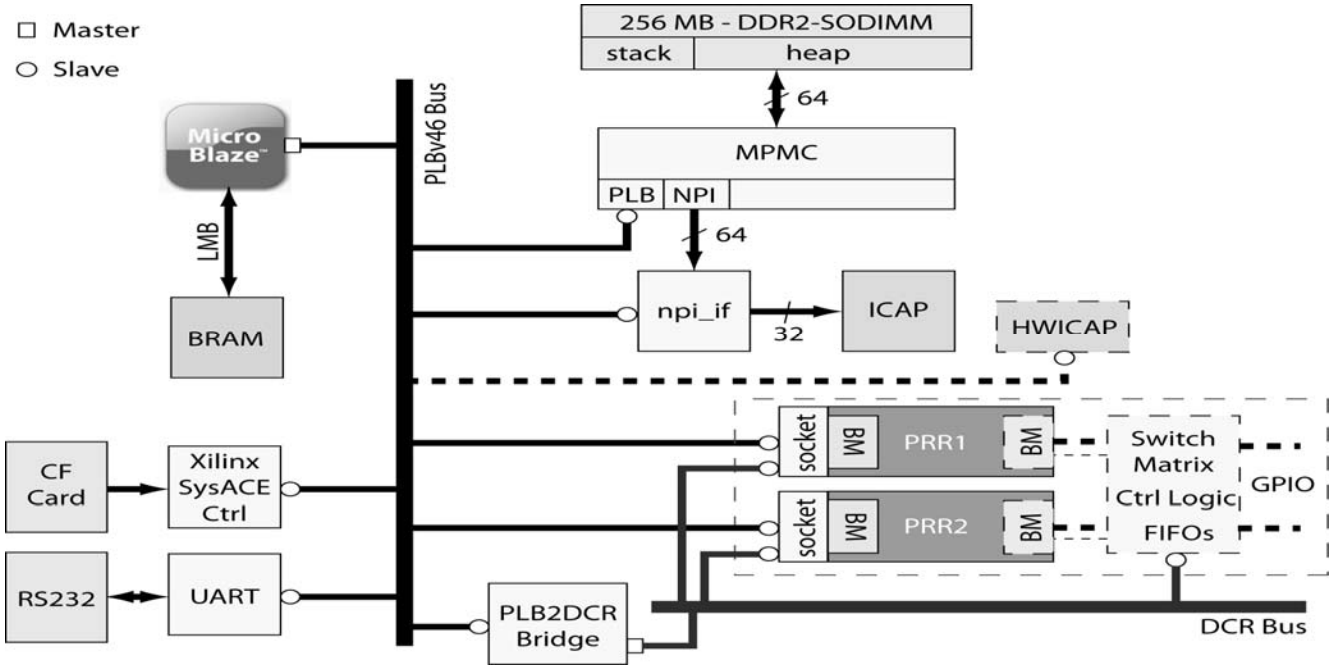


Bild 6: Beispiel einer prozessorbasierten Architektur eines DPR Systems mit zwei PRRs. Neben dem in dieser Thesis implementierten breitbandigen Konfigurationsdatenspeicher-zugriff kann alternativ auch der von Xilinx gelieferte HWICAP-Core verwendet werden. Dieser bietet allerdings nur eingeschränkte Performanz. Rot umrandet ist eine Beispielankopplung der PRRs über den DCR Bus.

Der µP lädt die Bitfiles über konventionelle FAT-Dateisystemfunktionen während des zeitunkritischen Bootvorgangs von der CF-Karte in ein externes DDR2-SDRAM (max. 4,2 GB/s). Der µP hat mit dieser Implementierung jederzeit durch den PLB-Port des Multi Port Memory Controllers (MPMC) wahlfreien Zugriff auf alle Konfigurationsdaten und kann diese somit auch während des Betriebs (wenn nötig) noch byteweise manipulieren. Zum Speicherzugriff nutzt der Beschleuniger einen zweiten Port des MPMC, welcher nach dem NPI-Protokoll (Native Port Interface) konfiguriert ist. Dieses Interface bietet hochperformante 64-word Burstfunktionalität (32bit/word) mit einem 64 Bit Datenbus.

4.2. Hardwarebeschleuniger npi_if

Die Implementierung eines Hardwarebeschleunigers verfolgt zwei Ziele: schnelle Rekonfiguration (geringer Overhead) bei gleichzeitiger Entlastung des µP. Wäre der µP selbst für den Datentransfer verantwortlich, könnte er während der Dauer einer Rekonfiguration keine anderen Aufgaben übernehmen, was in der Zielapplikation eventuell nicht akzeptierbar ist.

Der Hardwarebeschleuniger hingegen wird vom µP lediglich zu Beginn der Rekonfiguration über den PLB parametrisiert und gestartet. Nach Beendigung des Vorgangs meldet der Core über seine PLB Register Erfolg oder Fehlercodes an den µP zurück.

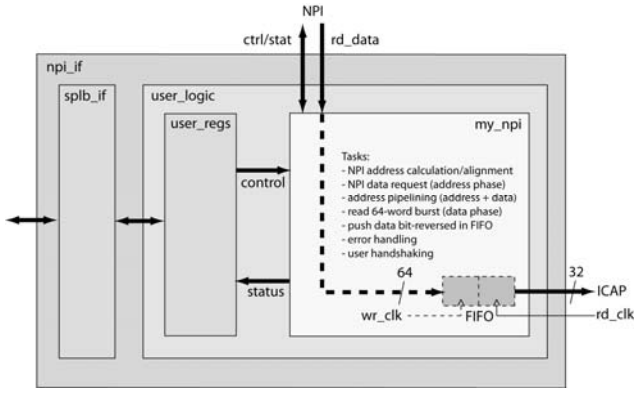


Bild 7: Struktur und Aufgaben des Hardwarebeschleunigers. Die Struktur des Cores entspricht der gängigen Xilinx PLB-peripheral Vorgabe. Den Transfer vom 64-bit NPI Datenbus zum 32-bit ICAP Datenbus übernimmt ein asymmetrisches FIFO, das außerdem die Synchronisation der verschiedenen Taktdomänen übernimmt (125 MHz (wr_clk) <–> ICAP Taktrate (rd_clk)).

Die Logik des Beschleunigers realisiert die DDR2-SDRAM Anbindung über das NPI Protokoll des MPMC und transferiert die gelesenen Daten zum ICAP Interface. Zu diesem Zweck führt die Logik Adressberechnungen entsprechend den Alignment-Bestimmungen des Protokolls aus und koordiniert die Adress- und Datenphasen während der Rekonfigurierung.

Das NPI Protokoll stellt eine native point-to-point 64-bit Datenanbindung mit Burstfunktionalität variabler Länge zur Verfügung. Wird das NPI Interface mit dem 125 MHz Systemtakt betrieben, kann eine maximale Datenrate von 1 GB/s erreicht werden (d.h. Kapazität der Anbindung) was für die garantierte Kapazität des ICAP-Ports (vergleiche Abschnitt 2.4) genügt. Die Logik verwaltet außerdem eine Fehlererkennung und die Benutzerkommunikation (Handshaking).

Zur Steigerung des Wirkungsgrads erlaubt das NPI Protokoll Address-Pipelining. Hierbei darf die Adressphase in die vorhergehende Datenphase gelegt werden, womit die negativen Auswirkungen der Adressphase auf den Wirkungsgrad (nahezu) eliminiert werden. Durch die Implementierung des Address-Pipelining konnte der Wirkungsgrad der RAM-Anbindung von 60,4% (604 MB/s) auf 97% (970 MB/s) gesteigert werden. Somit wäre mit dem Beschleuniger eine hypothetische ICAP-Taktrate von 242 MHz möglich.

4.3. Evaluierung des Beschleunigers

Zur zeitlichen Evaluierung der Leistungsfähigkeit des Hardwarebeschleunigers ist in eine Testarchitektur ein zusätzlicher Zählerbaustein aus der EDK-Bibliothek inkludiert. Der Core besteht aus zwei einzelnen Zählereinheiten, mit dem die Dauer der gesamten Rekonfiguration in zwei unterschiedlichen Kontexten festgestellt werden kann. Während ein Zähler die reine Dauer der DPR misst, d.h. die Zeit zwischen der steigenden Taktflanke des Startsignals und der steigenden Taktflanke der Beschleuniger-signalisierung „DPR beendet“, misst ein softwaregesteuerter Zähler die Dauer des Rekonfigurationsvorgangs aus Sicht des Prozessors, d.h. inklusive der notwendigen Datentransaktionen auf dem PLB sowie unter Berücksichtigung der C-Code Kompilierung in Maschinensprache. Der softwaregesteuerte Zähler wird vor dem DPR-Startbefehl über einen Funktionsaufruf gestartet und nach Abarbeitung einer Polling-Schleife wieder gestoppt.

Beide Zählerwerte variieren für eine identische DPR leicht. Im reinen Hardwareprozess verhindern variierende Arbitrierungs- und Aquirierungsvorgänge des MPMC beim Speicherzugriff einen aus Sicht des Beschleunigers streng deterministischen Prozess. Bei zusätzlicher Betrachtung der Softwarefunktionsaufrufe macht sich vor allem ein zeitlicher Overhead bei der Abarbeitung des Codes nach dem Kompilieren bemerkbar. Die Abfrage der Ende-Bedingung in der Polling-Schleife findet nicht jeden Takt statt, sondern

in einer zwischen 20 und 50 Zyklen andauernden Periode, je nach Komplexität der Schleifenabbruchbedingung sowie der PLB Verfügbarkeit. Im Allgemeinen wird daher eine DPR-Dauer gemessen, die länger ist als die wahre Rekonfigurationszeit des Beschleunigers, hierbei allerdings den Softwareoverhead dieser Realisierung mit abbildet.

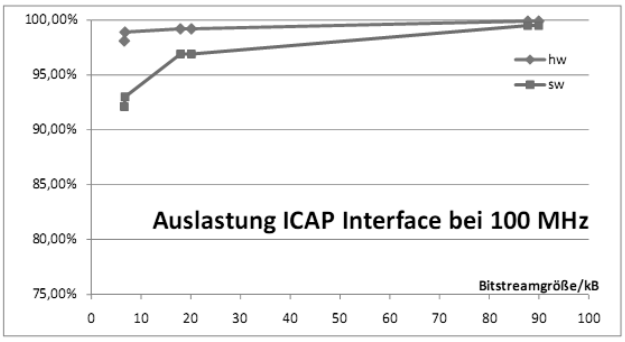


Bild 8: Graphisches Ergebnis der Beschleunigerevaluierung bei 100 MHz ICAP-Takt.

Bei der Messung wurde bei Streamgrößen von ca. 90 kB die maximal mögliche Auslastung und somit die minimale Rekonfigurationsdauer bis auf ca. ein Promille des Optimalwertes (bei Betrachtung der reinen Rekonfigurationsdauer) erreicht. Ein Bitstream der Größe 89,9 kB (Bitstreamgröße einer 480 CLB großen Test-PRR) wird in 224,9 µs (226 µs inkl. Software) rekonfiguriert , was einem Durchsatz von 399,85 MB/s (398 MB/s) entspricht.

4.4. Demonstrationsprojekt

Im Rahmen der Erstellung dieses Beitrags wurde ein kleines audiosignalverarbeitendes Demonstrations-projekt erstellt, in dem DPR zur Anwendung kommt. Ziel dieses Projekts ist eine einfache Demonstration, wie ein an den PLB angekoppelter Peripheriecore mittels DPR ausgetauscht wird.

Für das Projekt wurden zwei digitale FIR-Filter erstellt, eines mit Tief- und eines mit Hochpassverhalten, jeweils 12. Ordnung. Die Filter bilden zwei PRMs, die in eine entsprechend ausgelegte PRR abgebildet werden. Die Anbindung der Filterbausteine (bzw. der PRR) an den PLB geschieht über einen Sockelbaustein (*socket*). Der Socket hat generell die Aufgabe, alle zwischen Filtermodul und PLB verlaufenden Signale über Busmacros zu führen, um diesen einen geometrisch definierten Übergang zwischen statischem Bereich und der PRR zu garantieren. Eingangssignale in die PRR brauchen nicht schaltbar ausgelegt werden, Ausgangssignale zum PLB aber auf jeden Fall, um die während der DPR möglichen Signalinkosistenzen nicht auf den Bus zu führen. Die Steuerung des sockets geschieht über ein DCR-Businterface.

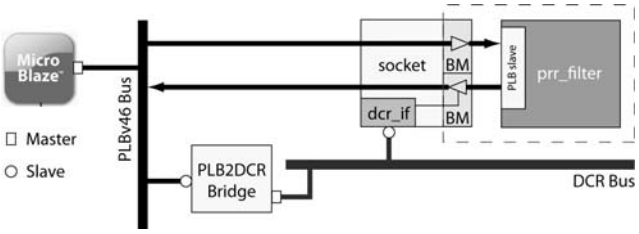


Bild 9: Rekonfigurationskonzept des Filters. Weitere Peripheriekomponenten wie z.B. A/D-Wandler sind nicht gezeichnet.

Bild 10 zeigt die Auslegung der PRR. Die Filtercores benötigen neben einem BRAM sieben DSP48 Blöcke. Die weitere Logik benötigt ca. 75% der innerhalb der PRR zur Verfügung stehenden Slices.

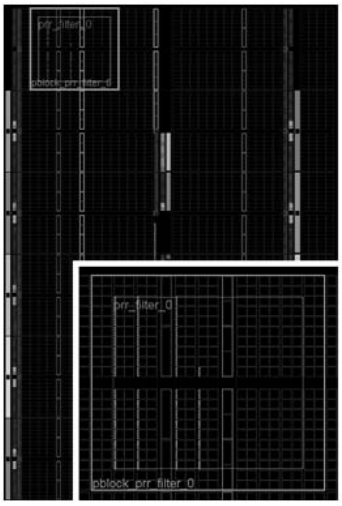


Bild 10: Floorplan des DPR-Demonstrationsprojektes. Die Gesamtansicht zeigt Größe und Position der Filter-PRR. Unten rechts in der vergrößerten Ansicht sind die platzierten Busmacros zu sehen (48 Stück).

Die beiden partiellen Bitstreams für Tief- und Hochpass haben jeweils eine Größe von 68104 B. Eine Rekonfigurierung der Filterfunktionalität geschieht in 170,5 µs, was einem Durchsatz im ICAP-Interface von 399 MB/s (von max. 400 MB/s) entspricht. In der audiosignalverarbeitenden Anwendung ist die Pause von 170,5 µs für das menschliche Ohr nicht mehr wahrnehmbar.

5. Zusammenfassung

In diesem Beitrag wurde versucht, dem Leser einen Einblick in die dynamische, partielle Rekonfigurierung eines FPGAs zu geben. Nach einer Beschreibung der technischen Grundlagen dieser Methode wurde mögliche Zielapplikationen vorgestellt. Weiters wurde die Implementierung eines DPR-fähigen Systems vorgestellt und evaluiert sowie Architektur-möglichkeiten präsentiert. Die Realisierung eines Hardwarebeschleunigers führte zu einem nahezu optimalen Rekonfigurationsprozess. Zum Abschluss wurden die erarbeiteten Grundlagen und Vorschläge in einem funktionsfähigen Demonstrationsprojekt verknüpft, das eine digitale Filterfunktionalität dynamisch partiell rekonfigurieren kann.

Es zeigt sich, dass das Prinzip der dynamisch, partiell rekonfigurierbaren FPGAs in vielen Facetten der digitalen Hardwareentwicklung neue, bisher unbekannte Wege, ebnet. War ein FPGA schon Kraft seiner „field-programmable“-Definition ein Logikbaustein mit einer inhärenten Flexibilität, erhöht die DPR-Methode diese nochmals um einen weiteren großen Schritt. Ressourcenspezifische statische Beschränkungen können durch dieses neue temporale Designparadigma umgangen werden, selbst energetische Aspekte können positiv verändert werden. Somit werden die Vorteile der bisher vorherrschenden Designlösungen GPP auf der einen und ASIC auf der anderen Seite im FPGA vereinigt, da dieses eine effiziente parallele Verarbeitung mit hoher Flexibilität verknüpfen kann.

Literatur

[1] Platzner M, Teich J and Wehn N (Hg.), 2010, Dynamically Reconfigurable Systems - Architectures, Design Methods and Applications, Dordrecht: Springer.

[2] Hauck S.und Dehon A., "Reconfigurable Computing: The Theory and Practice of FPGA-based Computation", Burlington: Morgan Kaufmann, 2008.

[3] DFG, "Rekonfigurierbare Rechnersysteme, Schwerpunktprogramm 1148", verfügbar unter: http://www12.informatik.uni-erlangen.de/spprr/ (Zuletzt geprüft am: 20. April 2010).

[4] DFG, "Rekonfigurierbare Rechnersysteme, Schwerpunktprogramm 1148 - Jahresbericht 2008", verfügbar unter: http://www.dfg.de/jahresbericht/detail_9_4_ELE_1148.htm (Zuletzt geprüft am: 20. April 2010).

[5] Donato A., Ferrandi F., Santambrogio M.und Scuito D., "Operating System Support for Dynamically Reconfigurable SoC Architectures" in Proceedings IEEE International SOC Conference, pp 235–238, 2005.

[6] Athalyse A.und Hong S., "Mapping of Partial Reconfigurable Data Flows to Xilinx FPGAs" in Proceedings IEEE International SOC Conference, pp 111–112, 2005.

[7] McDonald E. J., "Runtime FPGA Partial Reconfiguration" in IEEE Aerospace Conference, 2008.

[8] Xilinx Inc., "Virtex-5 FPGA Data Sheet: DC and Switching Characteristics, DS202 (v4.10)", verfügbar unter: http://www.xilinx.com/support/documentation/data_sheets/ds202.pdf (Zuletzt geprüft am: 29.06.2010).

[9] Xilinx Inc., "Virtex-5 FPGA Configuration User Guide UG191 (v3.8)", verfügbar unter: http://www.xilinx.com/support/documentation/user_guides/ug191.pdf (Zuletzt geprüft am: 05.05.2010).

[10] www.xilinx.com

Design for Testability (DFT) Strukturen für ASIC-Design und ihre Emulation auf FPGA

Benjamin Dusch, B.Eng.

Hochschule Offenburg, Badstraße 24

0781 / 205 267, benjamin.dusch@fh-offenburg.de

Mit dem Übergang zu immer komplexeren Designs an der Hochschule Offenburg werden DFT-Strukturen wie „Boundary Scan“ und „Scan“ in ASIC-Designs notwendig. Die DFT-Struktur Scan wird hierbei zukünftig bei Implementierung eines speziellen Scan Chain der Core Logic des ASIC-Designs verwendet und danach in der Boundary Scan Architektur integriert.
Zunächst werden die Strukturen im recht einfachen ASIC-Design „Rolling Dice“, entwickelt am IAF der Hochschule Offenburg, implementiert.
Nach Verifizierung der Funktionalität der Strukturen durch Emulation erfolgt die Einführung in komplexere ASIC-Design wie Front-End ASIC DQPSK sowie Prozessor-ASIC PDA V.2 (beide ebenfalls entwickelt am IAF der Hochschule Offenburg).
Eine Verifizierung der mit DFT-Strukturen ausgestatteten komplexeren ASIC-Design erfolgt im Rahmen dieser Ausarbeitung nicht, Bezug genommen wird hauptsächlich auf die Einführung der DFT-Strukturen in das ASIC-Design des „Rolling Dice“.
Ein Vergleich von Aufwand gegenüber Nutzen bei Implementierung von DFT-Strukturen in „kleine“ gegenüber „große“ ASIC-Design bildet ein wichtiges Fazit.

1. Einleitung

1.1 Motivation

In einem Zeitalter, in welchem Integrierte Schaltungen immer mehr an Komplexität zulegen, mikroelektronische Schaltungen aus immer mehr Bausteinen bestehen, ist es, um der Komplexität gerecht zu werden, zwingend nötig, eine mikroelektronische Schaltung eines ASICs (Application Specific Integrated Circuit) um zwei im Folgenden erläuterte Funktionen zu erweitern.
Die erste Funktionserweiterung ist die automatische Testbarkeit des ASICs nach der Herstellung. Die bisher am IAF nur ausgeführten manuellen funktionalen Tests sollen durch „Automatische Tests“ ersetzen werden. Das heißt, dass nach Herstellung des ASICs man diesen über eine bestimmte Testlogik, welche der funktionalen ursprünglichen mikroelektronischen Schaltung hinzugefügt wurde, ohne großen Aufwand testen können muss, um so die Verifikation der Funktion des ASICs nach der Herstellung zu vereinfachen.
Die zweite Funktionserweiterung ist die Debugging-Funktion eines ASICs, welche die Entwicklung eines ASICs (Application Specific Integrated Circuit) vereinfachen und damit die Entwicklungszeit verkürzen soll. Sinnvoll ist es von extern sämtliche Registerinhalte (Zustände der Flip Flops) während des Betriebs im Einzeltaktbetrieb überwachen und frei manipulieren zu können.
Im Rahmen dieser Arbeit werden beide Funktionserweiterungen eines ASIC-Designs erzielt.

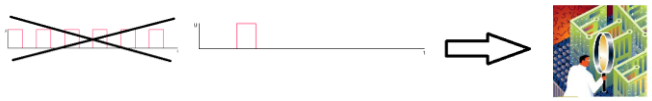


Abbildung 1: Verdeutlichung des Einzeltaktbetriebs zwecks Debugging eines ASICs

1.2. Design for Testability Strukturen

Bei DFT handelt es sich um Designtechniken in der Mikroelektronik, bei deren Anwendung mikroelektronische Schaltungen mit weiterer Testlogik erweitert werden, mit dem Ziel adäquate Testabdeckung bei gleichzeitiger Vereinfachung von Testmechanismen zu erzielen.
Verwendet werden in dieser Ausarbeitung als Designtechniken die DFT-Struktur Scan zur Implementierung eines Scan Chain in die Core Logic des ASIC-Designs (als Core Logic wird die gesamte funktionale Logik eines ASIC-Designs bezeichnet) sowie übergeordnet die DFT-Struktur Boundary Scan.
Die genannte Testlogik soll eine Boundary Scan Architektur sein, wobei es sich bei der Boundary Scan Architektur eher um eine Teststeuerungslogik handelt, durch welche andere integrierte Testlogiken kontrol-

liert werden. Es sollen durch die Boundary Scan Teststeuerungslogik innerhalb der Boundary Scan Architektur integrierte Scan Chain implementiert und gesteuert werden.

Für die Core Logic des ASIC-Designs wird zunächst ein Scan Chain implementiert.
Dadurch werden alle Flip Flops der Schaltung des ASICs auslesbar, steuerbar und manipulierbar gemacht.
Es werden hierfür bei der „Scan Insertion“ alle Flip Flops der Core Logic durch spezielle Scan Flip Flops mit integrierter Testlogik ersetzt, welche in einen Scan-Testmodus versetzt werden können.
Um diesen Scan-Testmodus nutzen zu können wird die Boundary Scan Architektur implementiert.

Im Normalmodus des ASICs nehmen die Scan Flip Flops ihre ursprüngliche Funktion innerhalb des Designs ein und die Scan-Funktionalität hat keinen Einfluss auf deren normale Funktion.
Der Testmodus hingegen kann aktiviert werden mittels Aktivieren einer speziellen Boundary Scan Instruction für den Scan Chain. Wird dieser Testmodus aktiviert, werden alle Scan Flip Flops seriell zu einem Schieberegister verschaltet, bilden einen Scan Chain von TDI nach TDO, welcher über Schiebevorgänge ausgelesen oder beliebig auf Werte aus 0 oder 1 gesetzt werden kann.

Dieses serielle Schieberegister wird als Scan Chain der Core Logic (siehe Abbildung unten) bezeichnet und ist also in einer die Steuerung des Scan Chain übernehmenden Teststeuerungslogik, der Boundary Scan Architektur, als Test Data Register (TDR) eingebunden.

Anstatt aufwendiger manueller Tests eines ASICs können so mittels der DFT-Struktur Boundary Scan und integrierter DFT-Struktur Scan „Automatische Tests“ auf einem ASIC-Design ausgeführt und desweiteren der ASIC im Debugging-Modus betrieben werden.

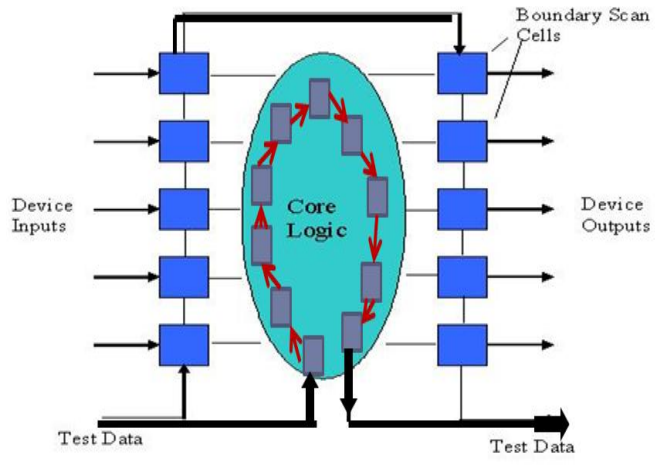


Abbildung 2: ASIC mit Boundary Scan und Scan (-Chain)

2. Allgemeine Herangehensweise

Der Einfachheit wegen werden die Funktionserweiterungen Debugging und Automatische Tests zunächst in einem einfachen ASIC-Design eingeführt, im vom IAF der Hochschule Offenburg entwickelten Design des „Rolling Dice“. Es handelt sich hierbei um ein ASIC Design, bei welchem bei Aktivierung des ASICs eine Zahl „erwürfelt“ wird. Dies geschieht, indem der Anwender eine frei gewählte Zeit lang eine Taste betätigt, ein Signal auf den ASIC gibt, damit die Logik aktiviert, welche eine Zahl von 1 bis 6 errechnet und über LED-Ausgabe anzeigt.
An diesem Design kann leicht nach Einführen der zusätzlichen DFT-Strukturen die korrekte Funktion des funktionalen Designs mittels manueller funktionaler Tests sowie auch der Testlogik von Boundary Scan und Scan nachgewiesen werden.
Der Nachweis der Funktionalität von Boundary Scan und Scan stellt zugleich den Nachweis der Funktionserweiterung „Automatische Tests“ eines ASIC-Designs dar. Automatische Tests werden im Umfang dieser Arbeit nur auf Basis funktionaler Boundary Scan Design (BSD)-Test-Pattern durchgeführt, wobei das Maß der Testabdeckung ermittelt werden wird.
Eine größere Testabdeckung ergibt sich bei Erzeugung und Verwendung scan-basierender ATPG (Automatic Test Pattern Generation)-Test-Pattern, die für die Automatischen Tests in Kombinationen mit den funktionalen BSD-Test-Pattern verwendet werden. Diese Art der Test-Pattern werden im Umfang dieser Arbeit nicht erzeugt.
Nach Bestätigung der Funktion der Automatischen Tests des ASIC-Designs über die Boundary Scan Architektur wird am ASIC-Design des „Rolling Dice“ eine Lösung gesucht, um hardware- wie softwareseitig mit dem fertigen emulierten ASIC-Design kommunizieren und das Design debuggen, im Einzeltaktbetrieb betreiben zu können.
Können die gewünschten Funktionserweiterungen des ASIC-Designs des „Rolling Dice“ bestätigt werden, erfolgt die Einführung der DFT-Strukturen in das komplexe ASIC-Design des „PDA V.2“.

An dieser Stelle kann dann ein kurzer Vergleich vorgenommen werden, welcher Aufwand gegenüber Nutzen bei Implementierung von DFT-Strukturen in „kleine“ gegenüber „große“ ASIC-Design thematisiert.

3. Scan/Boundary Scan Insertion in ASIC-Design

3.1 Einleitung

Die Einführung von Scan in die Core Logic von ASIC-Design und Boundary Scan in ASIC-Design erfolgt mittels des professionellen Programmes Design Vision von Synopsys durch die Skriptsprache Tickle (TCL), über welche Befehle an das verwendete Programm gegeben werden. Jeweils zusammengehörende Befehle werden zusammengefasst zu einem auszuführenden Skript, welches per Aufruf über die Kommandozeile ausgeführt wird.

Es werden folgende Skripte ausgeführt:

3.2 Scan Insertion

Skript 1: Definition der zu verwendenden Bibliotheken
Zunächst werden die bei der Synthese zu verwendenden Technologien als Bibliotheken in einem Skript definiert.

Skript 2: Definition von nicht zu verwendenden Standardzellen
Es werden anschließend in einem weiteren Skript die Standardzellen definiert, welche in der Synthese nicht verwendet werden sollen. Hier werden nun im Gegensatz zur normalen Synthese eines Designs Standard Flip Flops verboten, d.h. nur Scan Flip Flops zugelassen.

Skript 3: Einlesen, Verlinken und Prüfen des Designs
In diesem Skript werden die Quellcodes des ASIC-Designs eingelesen und das Design verlinkt sowie auf Korrektheit/Konsistenz geprüft.

Skript 4: Design Rule Checking des Designs
Bevor die Synthese erfolgen kann, wird mittels RTL Test Design Rule Checking (DRC) das Design auf mögliche Verletzungen von Design Rules gecheckt.

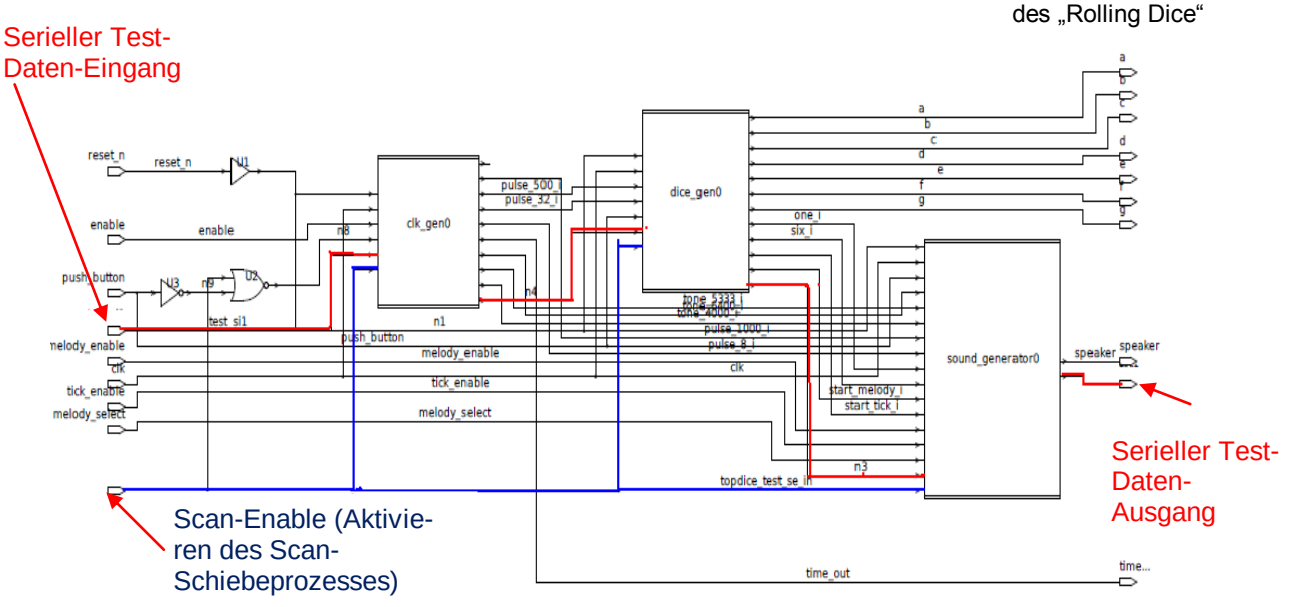
Skript 5: Synthese: test-ready-compile des Designs
Das Skript führt die eigentliche Synthese auf Gatter-Ebene durch und optimiert das Design hinsichtlich der zu erfüllenden constraints, also Auflagen, die das Design erfüllen muss und prüft letztlich nach dem Test-Ready Compilation nochmals die Einhaltung der Design Rules.

Skript 6: Definition und Implementierung des Scan Chains
Nun können die Definitionen für das Implementieren des Scan Chain erfolgen und der Scan Chain eingeführt werden.

Skript 7: Design Rule Checking des Scan-Designs
Nach Einführen des Scan Chain muss erneut das Design auf Verletzung Design Rules geprüft werden.

Das Ergebnis der Scan Insertion (der serielle Pfad durch die gesamte Core Logic) ist folgend am Design des „Rolling Dice“ dargestellt:

Abbildung 3: Scan Chain der Core Logic des „Rolling Dice“



3.3 Boundary Scan Insertion

Implementieren des Test Access Ports (TAP)
Bevor Boundary Scan per Skripte implementiert wird, erfolgt vorab die grafische Erweiterung des Designs um den TAP im professionellen Programm HDL Designer von Mentor.

Skript 1: Definition und Implementierung des Boundary Scan
Das Skript führt sämtliche Befehle aus, die zum Einführen der Boundary Scan Architektur samt TAP nötig sind.

Skript 2: Verifizieren des Boundary Scan Designs gemäß IEEE 1149.1
Die Einführung von Boundary Scan muss anschließend verifiziert werden und hinsichtlich der Einhaltung des Standards IEEE 1149.1 geprüft werden. Dafür muss das Design vorbereitet, ein spezieller Flow durchlaufen werden.

Skript 3: Erstellen von Test-Pattern sowie der BSDL-Datei
Zum Testen der Boundary Scan Logik werden in einer Verilog Test-Bench integrierte funktionale Boundary Scan Test Pattern herausgeschrieben, die später auf Basis der Netzliste des fertigen Designs simuliert werden. Auch wird ein BSDL-File (Boundary Scan Description Language File) erstellt, welches in IEEE 1149.1 definiert ist und der Beschreibung der Testfähigkeiten des Designs dient.

Skript 4: Herausschreiben einer Netzliste für Emulation
Um das Design auf dem FPGA-Emulations-TestBoard der Hochschule Offenburg nachbilden und testen zu können, muss vom fertigen Scan-Design eine spezielle Netzliste geschrieben werden.

Das fertige Boundary Scan Design des „PDA V.2“ ist in der unteren Abbildung dargestellt.
Das implementierte Boundary Scan Register, der endliche Zustands-Automat „TAP-Controller“ mit Boundary Scan Architektur neben der Core Logic mit Peripherie und PLL des ASIC-Designs, sind zu sehen.

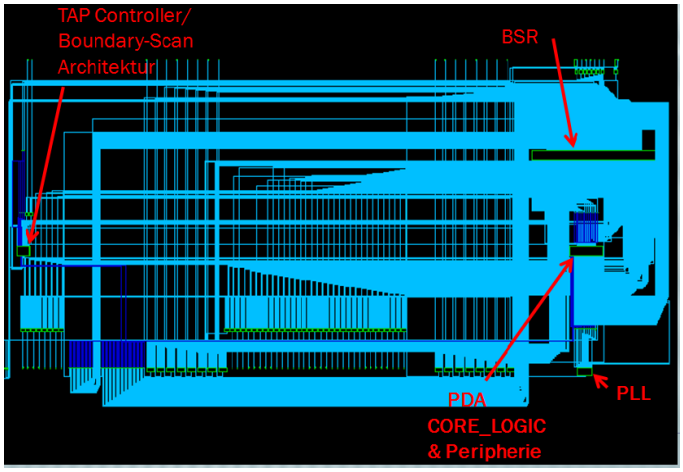


Abbildung 4: Boundary Scan Design des „PDA V.2“

4. Automatische Tests/Automatische Verifikation der Funktion des ASIC-Designs

Zum Erreichen der Fähigkeit Automatischer Tests wird die implementierte Boundary Scan Architektur mit integriertem Scan Chain der Core Logic verwendet.
Durch automatische Steuerung der Testlogik am Test Mode Select (TMS) und Einfüttern von Bitfolgen (Test-Pattern) am Test Data Input (TDI) bei gleichzeitiger spezieller Taktung des Systems über die Test Clock (TCK) und Überwachen der am Test Data Output (TDO) ankommenden Bitfolgen, kann das ASIC-Design durch Verwendung von Test-Pattern verifiziert und automatisch getestet werden.

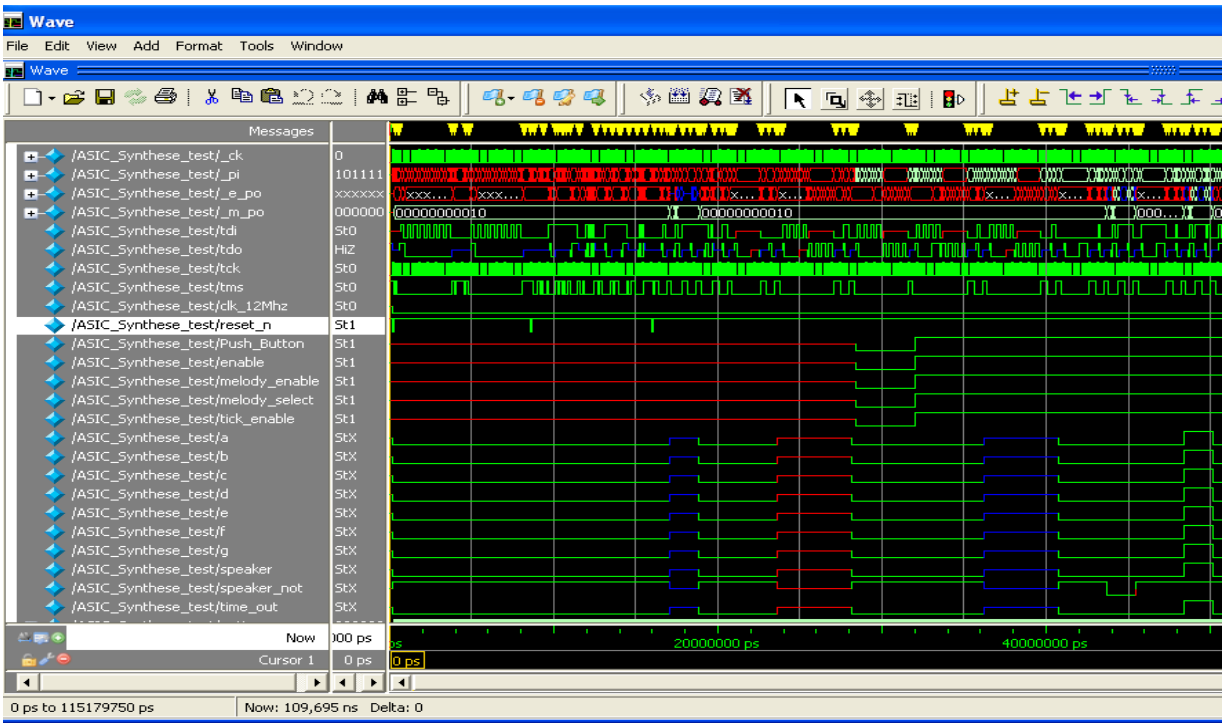
Automatische Tests, basierend auf den herausgeschriebenen funktionalen Test-Pattern, geben über zwei Aspekte Auskunft.
Werden diese Test-Pattern in einer Simulation am fertigen ASIC-Design verwendet, so sagt uns das Testergebnis, ob die implementierte Boundary Scan Architektur funktioniert und korrekt implementiert wurde.
Verwendet man diese Test-Pattern hingegen am gefertigten ASIC-Design, die Test-Pattern werden also in den ASIC tatsächlich eingefüttert und die Testergebnisse überwacht und mit Erwartungswerten abgeglichen, dann gibt uns dies Auskunft, ob Boundary Scan am gefertigten ASIC funktioniert und ob das ASIC-Design mit einer bestimmten Wahrscheinlichkeit korrekt gefertigt wurde, seiner ursprünglichen Funktion entspricht. Diese Wahrscheinlichkeit wird

bestimmt durch das Maß der Testabdeckung, die sich durch Verwendung der Test-Pattern ergibt. Test-Pattern können bei Verwendung nie eine Testabdeckung von 100 % erzielen, da es nicht möglich ist alle möglichen Fertigungsfehler festzustellen. Dem Maß der Testabdeckung entsprechend, verhält sich auch der Wert der Wahrscheinlichkeit, dass der ASIC korrekt gefertigt wurde. Die Vorgehensweise um das Maß der Testabdeckung zu erfahren, wird im darauffolgenden Kapitel behandelt.

An dieser Stelle wird nur der erste Typ von Automatischen Tests in einer Simulation ausgeführt.
In ModelSIM wird das Boundary Scan Design des ASICs mit entsprechenden Bibliotheken, die erstellte Verilog Test-Bench, welche die Test-Pattern beinhaltet, geladen, das Projekt kompiliert, ein Makro (Stimuli-Do-File) ausgeführt, um im Wesentlichen das System samt Boundary Scan vor der Simulation zu reseten sowie um die Simulationszeit festzulegen, und dann die Simulation der Test-Pattern gestartet.

Nach Beendigung der Simulation erscheint eine Ansicht aller Signalverläufe.
Das Wave-Fenster (siehe nächste Abbildung) zeigt Signale der Eingangs-/Ausgangsports des Designs bei der Simulation als Simulationsergebnis an. Wurden alle Test-Pattern erfolgreich simuliert und ergaben keine Fehler, so endet die Simulation mit der Bildschirmausgabe „TEST COMPLETED WITH NO ERRORS“.
Ist dieser Stand erreicht, ist sichergestellt, dass die implementierte Boundary Scan Architektur korrekt implementiert wurde und funktioniert.

Abbildung 5:
Wave-Fenster nach erfolgter Simulation der funktionalen Boundary Scan Design Test-Pattern in ModelSIM



5. Ermitteln der Testabdeckung durch die funktionalen BSD Test-Pattern

Die Testabdeckung gegeben durch die funktionalen BSD Test-Pattern, kann durch Simulation der Test-Pattern auf Basis eines Fehlermodells mittels des Programmes Synopsys TetraMAX ATPG errechnet werden.

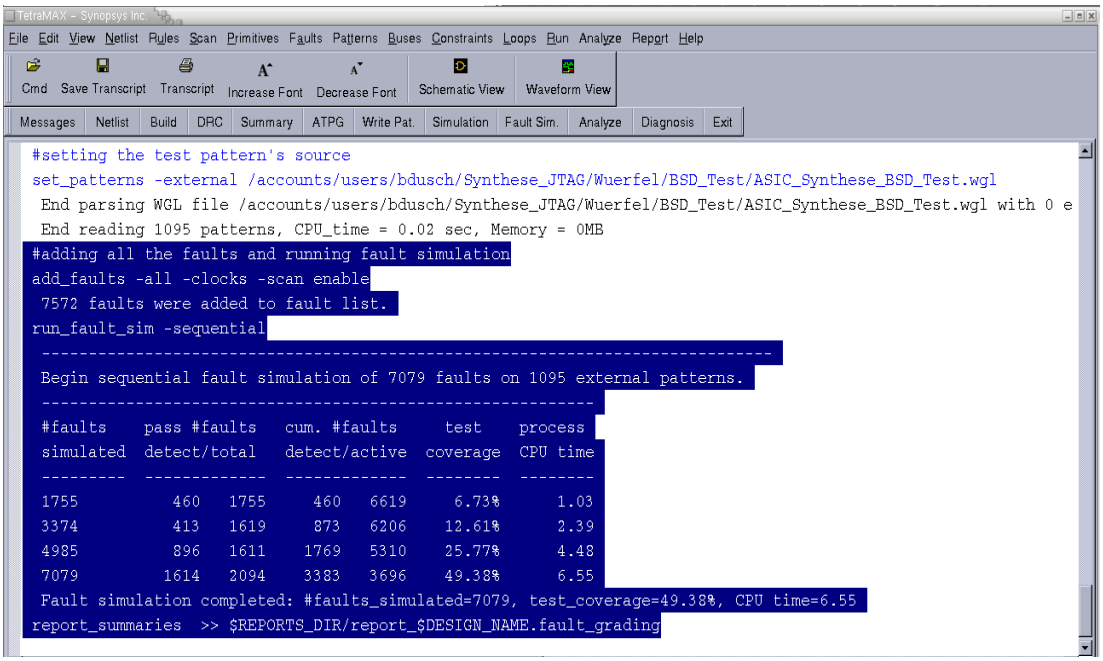
Diese Fehlersimulation wird wiederum skriptgesteuert durchgeführt.
Es werden Hardware-Fehler, die bei Fertigung des ASICs entstehen können, systematisch vom Programm automatisch zu einer Fehlerliste, zu einem

Fehlermodell, zusammengefügt, und durch Simulation von Test-Pattern ermittelt, welche der möglichen Fertigungsfehler gefunden werden können bzw. abgedeckt werden.

Die nächste Abbildung zeigt das Ergebnis der Testabdeckung. Die Testabdeckung von 49,38 % ist nicht ausreichend groß, da nur 3383 von 7079 Fehlern abgedeckt werden. In Anbetracht der Tatsache aber, dass nur funktionale BSD Test-Pattern verwendet wurden, ist das Ergebnis signifikant.

Dieses Programm kann, wie bereits erwähnt, im weiteren Sinne für das Herausschreiben von ATPG-Test-Pattern für ein Design verwendet werden.
Eine Optimierung der Testabdeckung wird im Rahmen der Erzeugung scan-basierender ATPG-Test-Pattern im Laufe meiner weiteren Arbeit am IAF erzielt werden.

Abbildung 6:
Ergebnis der Fehler-Simulation der funktionalen Boundary Scan Design Test-Pattern in TetraMAX



6. Emulation des ASIC-Designs zwecks Test der Debugging-Funktion

Die Funktionserweiterungen werden durch Emulation des fertigen Boundary Scan Designs auf einem FPGA-Emulations-Testboard nachgewiesen.

Auf dem Board befindet sich ein FPGA, Programmier-Schnittstellen sowie den konfigurierbaren I/Os des FPGAs zugeordnete Steckerleisten zur Kontaktierung der FPGA-I/Os. Über die Steckerleisten ist es somit möglich, I/Os des FPGAs, welche im Rahmen der Emulation Ports des ASICs nachbilden sollen, über die Steckkontakte zu erreichen, mit Signalen zu beaufschlagen oder per Messung zu überwachen.

Das FPGA-Emulations-Testboard verfügt desweiteren über I/Os mit angeschlossenen LEDs.

Zum Emulieren ist nun die spezielle Netzliste des Designs, herausgeschrieben durch Skript 4 der Boundary Scan Insertion, zu verwenden. Das ASIC-Design wird im FPGA nachgebildet, wobei Ports des FPGAs Ports des ASICs darstellen, nachbilden sollen. Im Programm Altera Quartus II werden hierfür vorher Pins des FPGAs den Ports des ASIC-Designs zugewiesen.
Jedem Eingangs-Port und jedem Ausgangs-Port des ASICs werden Pins des FPGAs zugewiesen. Die I/Os des FPGAs, welche über angeschlossene LEDs verfügen, werden als Ausgänge des ASICs verwendet, die LED zeigen damit Würfel-Ergebnisse an, wenn der ASIC arbeitet. Dies ist ideal für die Demonstrationszwecke der Debugging-Funktion am ASIC-Design.

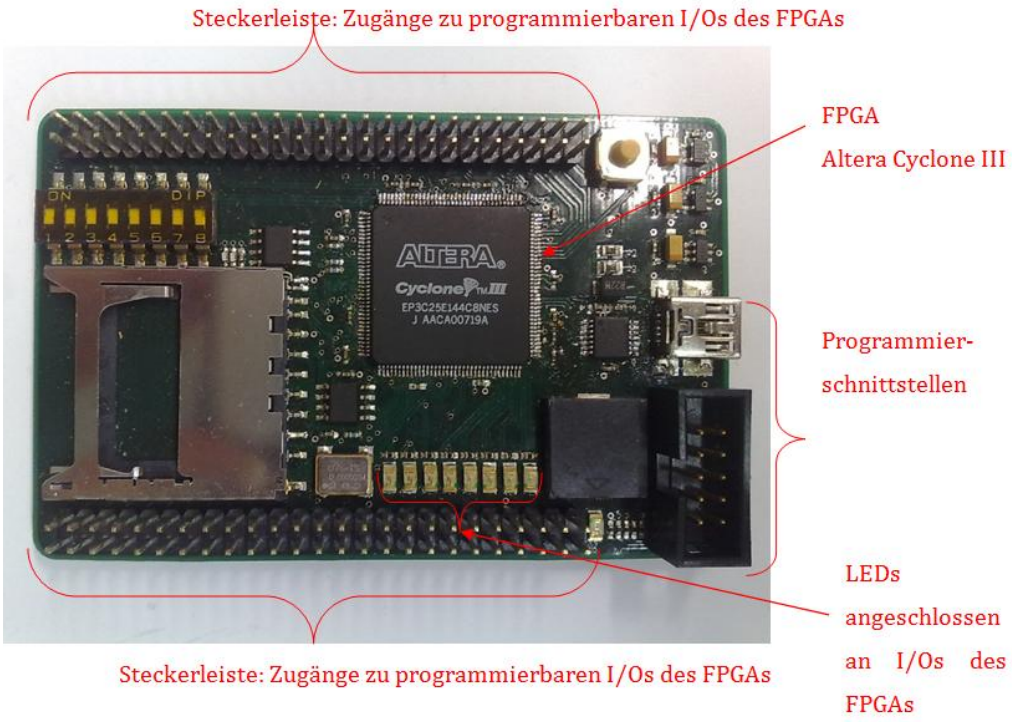


Abbildung 7: FPGA-Emulations-Testboard der Hochschule Offenburg

7. Debugging des ASIC-Designs: Einzeltaktbetrieb

7.1 Verwendete Hardware/Software

Im Rahmen von Recherchen hinsichtlich möglicher zu verwendender Software und Hardware, um ein Boundary Scan ASIC-Design anzusprechen, steuern und debuggen zu können, fiel die Wahl auf das XJTAG Development System.
Dieses beinhaltet diverse Programme zur Verwendung bei Boundary Scan Design sowie eine USB zu JTAG Schnittstelle.

Zur Demonstration der Debugging Funktion wird das Programm XJDeveloper verwendet und das auf einem FPGA-Emulations-Testboard emulierte ASIC-Design des „Rolling Dice“ über das Interface mit dem PC bzw. der Software verbunden.

7.2 Konfiguration des Programmes XJDeveloper / Einbindung des emulierten ASICs / Inbetriebnahme des ASICs

Für die Erkennung des ASIC-Designs ist eine Konfiguration des Programmes auszuführen, die spezifisch auf das ASIC-Design ausgerichtet ist.

Der genaue Ablauf der Konfigurierung soll in dieser Dokumentation nicht behandelt werden, es erfolgt lediglich ein grobes Erklären wichtiger Aspekte.

7.2.1 Laden der Board-Netzliste und der Boundary Scan Description Language (BSDL)-Datei

Das Programm XJDeveloper ist so aufgebaut, dass es einen ASIC nur dann hardware-mäßig erkennt, wenn dieser auf einem Board platziert bzw. aufgelötet ist. Deshalb wird eine Board-Netzliste für den ASIC geschrieben und der ASIC weiterhin bei Verwendung des Programmes XJDeveloper als Bestandteil eines Boards betrachtet.
Es ist eine spezielle Board-Netzliste im RINF-Format für das ASIC-Design und das im Kapitel der Boundary Scan Insertion generierte BSDL-File zu laden, damit das Programm XJDeveloper den ASIC grundsätzlich als „Board“ erkennt und Definitionen über dessen Testfähigkeiten erhält. Die manuell geschriebene Board-Netzliste und automatisch generierte BSDL-Datei werden per Programm-Menü eingelesen.

7.2.2 Konfigurieren der JTAG Chain

Eine JTAG Chain ist zu definieren, d.h. der Scan Chain der Core Logic vom Test-Daten-Eingang TDI zum Test-Daten-Ausgang TDO des ASICs zu beschreiben.

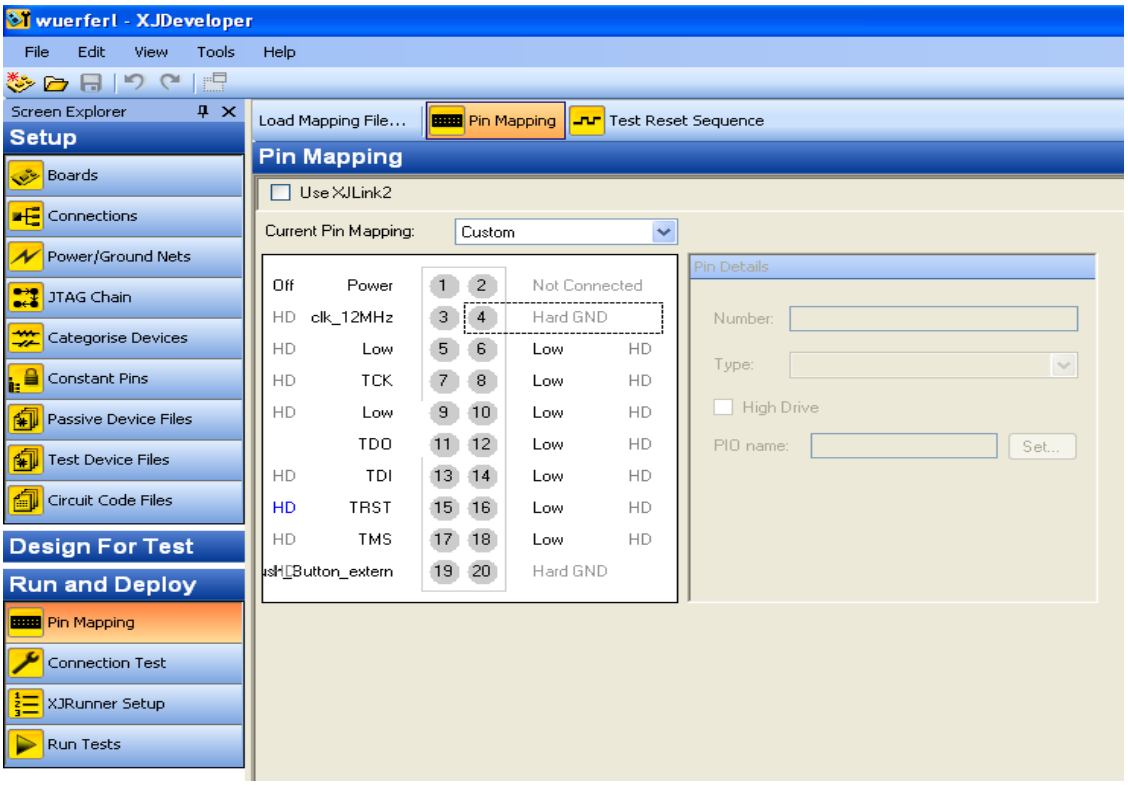
7.2.3 Pin Mapping der JTAG zu USB-Schnittstelle

Es ist ein „Pin Mapping“ der JTAG zu USB-Schnittstelle von XJTAG zu definieren, d.h. hauptsächlich festzulegen, welchen der 16 möglichen frei konfigurierbaren Anschluss-Pins der Schnittstelle die einzelnen Ports TDI, TDO, TCK und TRST des Test Access Ports (TAP) zuzuweisen sind.

Das Programm ermöglicht im weiteren Sinne über Befehle *SET PIO.XXX := 0* (oder *1*); Eingänge XXX des ASICs anzusteuern, wobei per XJTAG-Schnittstelle der jeweilige ASIC-Eingang mit dem PC, dem Programm XJDeveloper, verbunden werden muss. Diese PIO-Verbindungen, „Programmable I/Os“, müssen an dieser Stelle auch konfiguriert werden.

In diesem Fall werden für die Demonstration der Debugging-Funktion des ASIC-Designs der Reset *TRST* zum Ausführen des System-Resets und der Eingang *Push_Button_extern* (Push_Button-Port) des ASICs zum Aktivieren der Funktion „Würfeln“ des ASICs „Rolling Dice“ als ansteuerbare Eingänge eingerichtet.

Abbildung 8:
Pin Mapping der JTAG zu USB-Schnittstelle



7.3. Schreiben eines Test-Programmes zur Demonstration der Debugging-Funktion am emulierten ASIC-Design

Das Programm ist nun grundsätzlich konfiguriert und das emulierte ASIC-Design angeschlossen.
Es kann nun im Programm XJDeveloper in einer XJTAG-eigenen Programmiersprache programmiert, das Programm kompiliert und ausgeführt werden.
Im Rahmen dieser Arbeit wurde ein umfangreiches Programm geschrieben, welches die Funktionen des Rolling Dice im Einzeltaktbetrieb bedient, gleichzeitig Registerinhalte aller Flip Flops des Scan Chains der Core Logic des ASIC–Designs des „Rolling Dice“ ausliest und anzeigt.

In einem Programmierfenster (Test Editor) im Programm XJDeveloper zum Schreiben, Kompilieren und Ausführen eines Debugging-Programmes über Boundary Scan / JTAG wird ein Testprogramm (Test

Device File) geschrieben, das Ausführen des Testprogrammes erfolgt per einfaches Anwählen der Funktion „Run Tests“ des Programmes XJDeveloper.

Visualisiert wird das ausgeführte selbst geschriebene Programm je nach Programmierung in einem Textausgabefenster, die Debugging-Funktion über Boundary Scan wird damit verdeutlicht.

7.4 Ausführen der Debugging-Funktion

Das Debugging-Testprogramm detailliert zu beschreiben würde hier den Rahmen sprengen, weshalb in Auszügen kurz auf Ergebnisse eingegangen wird.

Die folgende Abbildung zeigt eine ausgegebene Bestätigung eines durchgeführten Resets zu Beginn des Testprogramms als Textausgabe an, ebenso werden alle Registerwerte der Scan Flip Flop des TDR auf dem Bildschirm binär angezeigt.
Da Boundary Scan Register (BSR) und Scan Chain der Core Logic in einem TDR zusammengefasst sind (STT, „Scan Through TAP"), können sowohl Zustände von Ein-/Ausgängen des ASICs als auch von Flip Flops des ASIC ausgelesen und angezeigt werden.

Dank einer erstellten Beschreibung des Scan Chains der Core Logic am Ende der Scan Insertion Synthese, können nun alle Bits des TDR beim Debuggen eindeutig den jeweiligen Flip Flops des ASIC-Designs zugeordnet werden.

Die untere Abbildung zeigt alle Zustände der Ein-/Ausgänge des konfigurierten emulierten ASICs mit beschalteten Eingängen und alle Zustände der Flip Flops der Core Logic des ASICs „Rolling Dice“ nach Reset

Der Zähler „Counter“ des ASIC-Designs des „Rolling Dice“ ist nach dem Reset auf 0.
Wird der ASIC durch das Debugging-Testprogramm über Boundary Scan gesteuert und im Einzeltakt betrieben, der ASIC in seiner Funktion gestartet, so zählt der Counter ca. alle 1000 Takte (f = 1 kHz) um eins hoch. Nach Einzeltakt 1017 bei Takt 1018 zählt der Counter um eins hoch, so wie dies von der Logik erwartet wird (siehe darauffolgende Darstellung).

Die Zustände aller Flip Flops der Core Logic des ASICs sowie der Ein-/Ausgänge des ASICs konnten bis zu diesem Takt Nummer 1018 beobachtet und analysiert werden.

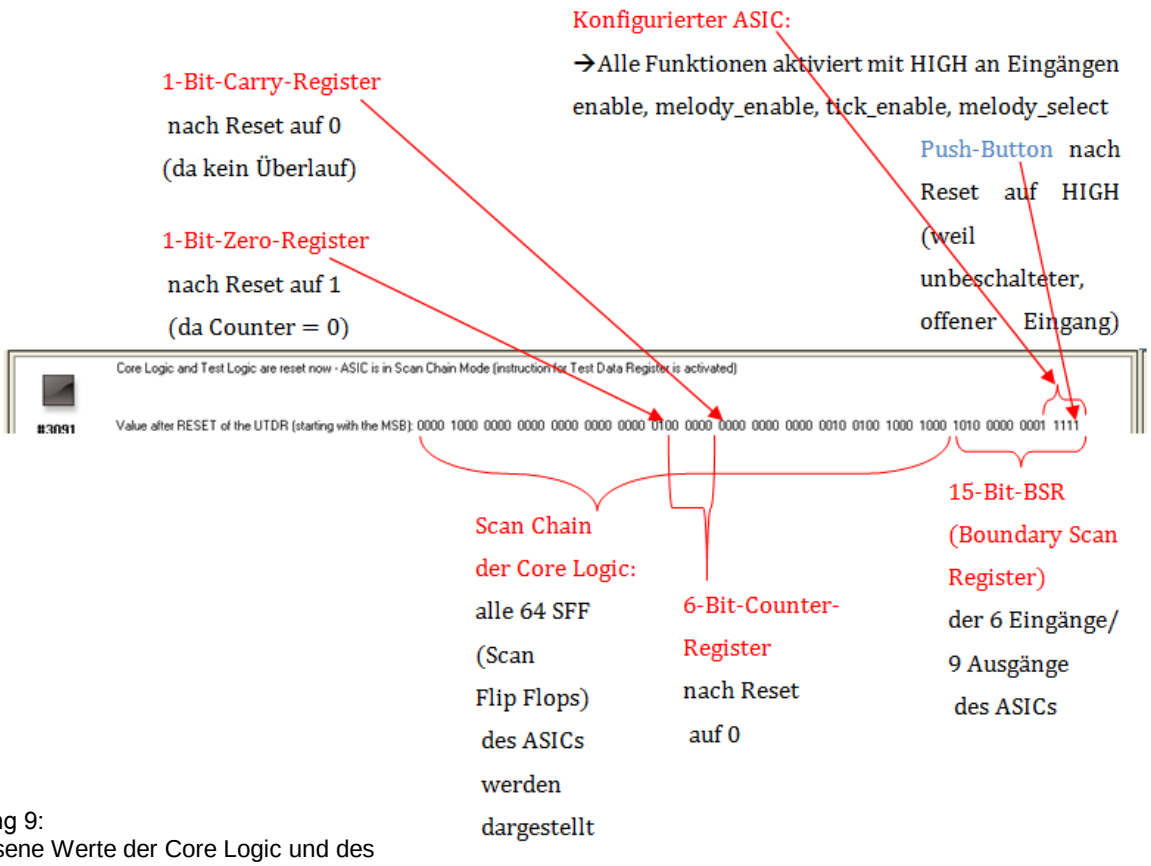


Abbildung 9:
ausgelesene Werte der Core Logic und des Boundary Scan Registers (BSR) des „Rolling Dice“ nach Reset der Testlogik

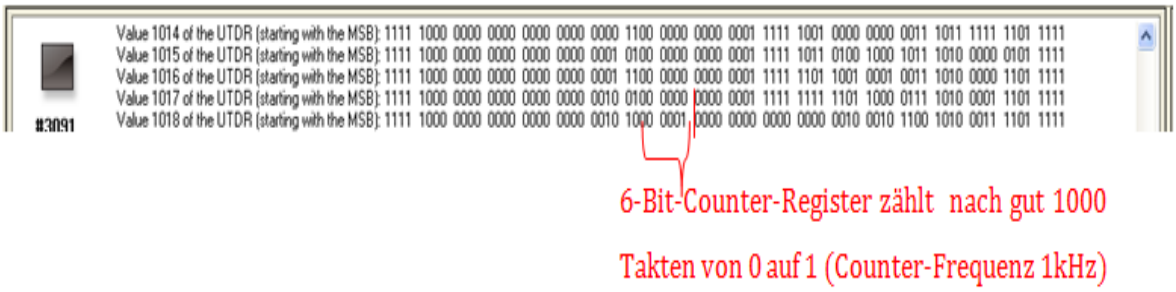


Abbildung 10: ausgelesene Werte der Core Logic und des Boundary Scan Registers (BSR) des „Rolling Dice“

8. Einführung von DFT-Strukturen: Abwägung von Nutzen gegenüber Aufwand

Anzahl Standardzellen des ASIC-Designs	mit DFT	ohne DFT
„Rolling Dice“	1170	400
„PDA V.2“	17000	16000

Tabelle 1: benötigte Standardzellen mit/ohne DFT je nach Größe eines ASIC-Designs

Die Einführung der DFT-Strukturen in das „kleine ASIC-Design des „Rolling Dice“ zeigt einen verhältnismäßig großen zusätzlichen schaltungstechnischen Aufwand, der bei der Umsetzung der Schaltung auf Ebene der Standardzellen zu erbringen ist. Das Maß der benötigten Standardzellen erhöht sich beinahe auf 300%. Bei Implementierung in das relativ „große“ ASIC-Design des „PDA V.2“ verringert sich der Mehraufwand relativ betrachtet, das Maß der benötigten Standardzellen erhöht sich auf weniger als 110%.

Sich nun festzulegen und zu sagen, dass Einführung nur in „große“ ASIC-Design lohnt, will ich mir nicht anmuten. Ob der je nach ASIC-Design nötige Mehraufwand sich lohnt einzugehen, muss im Einzelfall abgewogen werden. Hierbei sind die enormen Vorteile positiv ins Kalkül zu ziehen, die sich durch die Funktion „Automatische Tests“ und „Debugging“ des ASICs, ergeben.

9. Fazit und Ausblick

Die DFT-Struktur Boundary Scan mit integrierter DFT-Struktur Scan (zur Implementierung des Scan Chain der Core Logic) wurde erfolgreich in das einfache ASIC-Design des „Rolling Dice“, in das Frontend-ASIC-Design des „DQPSK“ und in das Prozessor-ASIC-Design des „PDA V.2“ eingeführt.
Am Design des „Rolling Dice“ konnten die durch die Einführung dieser Struktur zu erzielenden Funktionserweiterungen „Debugging-Funktion“ und „Automatische Tests“ vollständig nachgewiesen, grundlegende Vorgehensweisen erarbeitet und verifiziert sowie die Grundsteine gelegt werden für vertiefende Entwicklungsarbeiten bezüglich dieser neuen Funktionen von ASIC-Designs.
Das Design des PDA V.2 konnte im zeitlichen Rahmen bisher nicht überprüft werden, hier sind zukünftige Aufgaben noch auszuführen, zu denen das Verifizieren des ASIC-Designs mittels Manueller und Automatischer Tests gehört.

Allgemeine Entwicklungsarbeiten, die hinsichtlich der eingeführten DFT-Struktur Boundary Scan noch angegangen werden müssen, umfassen beispielsweise aus dem Bereich der Automatischen Tests die weitere Erforschung der Möglichkeiten, die Testabdeckung des Designs mittels zu kreierender scan-basierender ATPG-Test-Pattern zu erhöhen, dann das Entwickeln einer hard-/softwareseitigen Architektur, um gefertigte ASIC-Designs über den implementierten Boundary Scan mittels der Test-Pattern Automatischen Tests unterziehen zu können.
Im Bereich des Debugging ist spezifische Software zu erschaffen, um bei der Entwicklungsarbeit ASIC-Design über den eingeführten Boundary Scan möglicherweise über eine Benutzeroberfläche geeignet im Einzeltaktbetrieb betreiben zu können.

Am Ende dieser Dokumentation bleibt zu sagen, dass sich durch DFT-Strukturen vielseitige Möglichkeiten ergeben, weitere Entwicklungsarbeit zu betreiben, da dieses weite Feld noch längst nicht ausgereizt wurde.

10. Literaturverzeichnis

IEEE Standard Test Access Port and Boundary Scan Architecture 1149.1-2001. (27. März 2008).
Synopsys BSD Compiler User Guide. (Version D-2010.03-SP2, June 2010).
Synopsys BSD Compiler Reference Manual (Version D-2010.03-SP2, June 2010).
Synopsys DFT Compiler Scan User Guide. (Version C-2009.06, September 2009).
Synopsys TetraMAX® ATPG User Guide. (Version C-2009.06, June 2009).
Synopsys Library Compiler™ Methodology and Modeling Functionality in Technology Libraries User Guide . (Version C-2009.06, September 2009).
Synopsys Synthesis Commands. (Version D-2010.03, March 2010).
Synopsys DFT Overview. (Version A-2007.12, December 2007).

Übertragung und Echtzeitverarbeitung eines Datenstroms zwischen Virtex-4 FPGA und Multi-Core PC mit USB

Jan Philipp Kießling, Marcel Beuler

FH Gießen Friedberg, Wiesenstraße 14, Gießen

Tel.: 0641 309 -1943, JP-Kiessling@gmx.de, Marcel.Beuler@web.de

In der Biologie und Pharmazie werden neuronale Netze berechnet, um etwa psychische Erkrankungen oder Medikamente zu erforschen. Da eine Simulation mit aktuellen PC-Systemen sehr lang dauert, wurde das Projekt Nepteron gegründet. Sein Anspruch ist es, neuronale Netze mittlerer Größe in Echtzeit zu berechnen und darzustellen. Um dieses Ziel zu erreichen, müssen die Daten eben so schnell an einen PC gesendet und dort verarbeitet werden, wie sie der FPGA produziert. Dieses Paper beschreibt die dazu verwendete Hardware und ein Konzept für die PC Software.

1. Vorstellung des Nepteron Projekts

Das Ziel dieser Projektgruppe ist es, neuronale Netze mittlerer Größe in Echtzeit zu berechnen. Unter Echtzeit wird hier verstanden, dass z.B. die Simulation von 20 Sekunden Zellverhalten auch 20 Sekunden dauert. Mittlere Größe heißt, dass Netze in der Größenordnung 500.000 Zellen angestrebt werden. Dies soll durch das Vernetzen mehrerer ASIC-Chips realisiert werden, die jeweils mehrere Neuronenkerne beinhalten. Die Zellen werden nach dem Huber-Braun Modell berechnet, was einen sehr guten Kompromiss zwischen Rechenaufwand und Realitätsnähe darstellt [1].

Es sei an dieser Stelle darauf hingewiesen, dass es nicht darum geht, Berechnungen von dem Netz ausführen zu lassen. In der Informatik werden neuronale Netze beispielsweise zur Gesichtserkennung eingesetzt. Dabei werden die Netze so angelernt, dass sie auf bestimmte Eingabedaten bestimmte Ausgabedaten erzeugen. Dies ist hier wie gesagt nicht der Fall. Vielmehr wird das Verhalten von menschlichen Zellen simuliert. Die Zustände der einzelnen Zellen sind von Interesse, um zum Beispiel die Wirkung von Medikamenten zu untersuchen oder die Gründe für Schlafstörung zu erforschen. Es geht im Grunde um die Frage: Was passiert mit einer Gruppe Zellen, die sich im Zustand A befinden, wenn man X tut? Wobei A und X vom Benutzer per PC frei wählbar sind.

Um die digitale Architektur zu entwickeln und zu testen wird ein Virtex-4 FPGA eingesetzt. Auf diesem ist zur Zeit ein Neuronenkern implementiert, der 400 Neuronen in Echtzeit berechnen kann. Außer dem eigentlichen Neuronenrechner [2] wird eine USB 2.0 Verbindung zu einem PC benötigt und ein Programm, dass die ankommenden Daten entgegen nimmt, verarbeitet und in einer Graphischen Benutzer-oberfläche (GUI) darstellt.

2. Vorstellung der Strecke

Abbildung 1 zeigt das Blockdiagramm der gesamten Strecke von der Datenquelle (FPGA) zur Senke (Benutzer). Die Datenrichtung am USB wird allerdings immer aus Sicht des PCs definiert. Das heißt wenn der PC Daten anfordert und bekommt ist dies eine Eingabe. Ausgabe bedeutet, dass der PC Daten zum FPGA sendet. Orange Blöcke gehören zur Hardware des Neuronenrechners und blaue Blöcke zum PC.

Wenn eine C++ Funktion Daten senden will, dann übergibt sie diese an eine Funktion der CyAPI [3]. Diese Bibliothek wird von Cypress zur Verfügung gestellt, um dem Programmierer die Arbeit mit WinAPI und Treibern abzunehmen. Nachdem die Daten auf diesem Weg das Betriebssystem passiert haben gelangen sie zum USB Hostcontroller auf dem Mainboard. Dieser überträgt sie über differentielle Datenleitungen zum Cypress USB-Controller[1]. Er wird in Abb.1 durch die drei Blöcke „GPIF“, „USB Slave“ und „8051“ dargestellt. Der „USB Slave“ gibt sie chipintern an das sogenannte General-Programmable-Interface (GPIF) weiter. Das GPIF ist ein programmierbarer Zustandsautomat, über den externe Peripherie – in diesem Falle der Virtex-4-FPGA – mit dem USB-Controller verbunden werden kann. Hierfür stehen neben einer Takt- und 8 bzw. 16 bidirektionalen Datenleitungen noch diverse unidirektionale Output- und Handshake-Leitungen zur Verfügung. Insgesamt lassen sich bis zu vier unterschiedliche Datenübertragungen zwischen FPGA und GPIF programmieren, von denen jeweils nur eine ausgeführt werden kann. Für eine ausführliche Betrachtung der GPIF-Programmierung siehe [3]. Welche Übertragungsart wann eingesetzt wird

1 Exakter Name: CY7C68013A-128AX

entscheidet die Firmware des integrierten 8051. Sie kann nach Belieben geändert werden und kontrolliert neben dem GPIF noch einige nach außen geführte Pins.

Auf der FPGA Seite übernimmt die sogenannte FIFO-Management-Unit (FMU) die Bedienung des GPIF. Sie ist in VHDL geschrieben und setzt sich im Wesentlichen aus zwei Dualport-Speicherblöcken für beide Kommunikationsrichtungen sowie zwei Zustandsautomaten für die Steuerung der beiden Übertragungsrichtungen zusammen. Die Speicherblöcke sind mit FIFO_IN und FIFO_OUT bezeichnet und beinhalten jeweils eine frei skalierbare Anzahl an Pufferspeichern der Organisation 512*8 Bit zur Aufnahme eines USB-Pakets. Die Systemumgebung auf dem FPGA stellt ein Top-Level-Modul dar und bietet zu Testzwecken drei Betriebsarten: Copying, Generating und Verifying. In der Betriebsart “Copying“ werden die vom PC gesendeten USB-Pakete direkt wieder zurück zum PC geschickt. Im Modus “Generating“ generiert die Systemumgebung fortlaufend pro Millisekunde eine frei einstellbare Anzahl von Dummy-Paketen und überträgt diese in den FIFO_IN zur weiteren Übertragung an den PC, der sie auf Korrektheit prüft. In der letzten Betriebsart “Verifying“ hingegen generiert der PC fortlaufend Dummy-Pakete und sendet sie in Richtung FMU. Parallel zum Auslesen aus dem FIFO_OUT werden die Pakete vom Top-Level-Modul in Echtzeit auf Korrektheit geprüft.

Die Daten passieren naturgemäß jede dieser sechs Stationen und daher bestimmt das langsamste Glied der Kette die maximale Datenrate.

3. Anforderungen an die Strecke

Der zur Zeit implementierte Neuronenkern berechnet 400 Neuronen in Echtzeit und wirft dabei 3*512 Byte Daten pro Neuronenzyklus aus. Da ein solcher Zyklus 100 µs dauert ist eine Datenrate von mindestens 15,36 MB/s nötig. Wenn sie für mehr als 100 µs unterschritten wird, beginnen die Puffer im FPGA voll zu laufen. Diese können den Datenstrom für 1 ms vorhalten, ohne das der Neuronenkern bei seinen Berechnungen gestört wird. An sich ergibt das eine harte Echtzeitanforderung, weil – ob mit oder ohne Puffer – die Datenrate auf Dauer nicht unter die genannten 15,36 MB/s fallen darf. Tut sie das doch, gehen entweder Daten verloren oder der Neuronenkern muss angehalten werden. Falsche oder fehlende Daten können die Benutzer des Systems verwirren und zu falschen Forschungsergebnissen führen. Ein Stoppen der Berechnungen verlängert zwar die Simulationsdauer, wenn diese Unterbrechungen aber so kurz sind, dass der Benutzer keine Verzögerung erkennen kann, können sie in Kauf genommen werden. Es bestehen also zwei Anforderungen an die Datenstrecke:

- Die Datenrate soll nicht unter 15,36 MB/s fallen.
- Die Datenübertragung muss sehr sicher sein.

Wie kurz „kurz genug“ und wie sicher „sicher genug“ ist soll an dieser Stelle offen bleiben. Die folgenden Tests ermitteln wie schnell und wie genau das vorliegende System tatsächlich ist. Anhand dieser Ergebnisse wird entschieden, ob es gut genug ist.

4. Untersuchung der Strecke

Ein USB-Paket besteht physikalisch aus 512 Bytes. Um die Übertragungsrate zu messen, müssen mehrere hundert Pakete verschickt werden; zum Messen der Übertragungssicherheit mehrere Milliarden. Selbst die kleine Menge von fünf Paketen kann mit den verfügbaren Logikanalysatoren nicht mehr verarbeitet werden. Daher finden die Messungen innerhalb eigens entwickelter C++ Programme statt. Außerdem wird die 8051 Firmware modifiziert, um bei bestimmten Ereignissen einen Pin auf High oder Low zu setzen.

4.1 Maximale Datenrate ohne FPGA

Um ein Richtmaß für folgende Messungen zu haben, wurde mit Hilfe des Verfahrens aus [5] die maximale Datenrate ermittelt. Dabei fordert eine PC-Software Daten an, der im Chip integrierte 8051 Kern stellt sie an den Endpoints zur Verfügung und misst die Zeit,

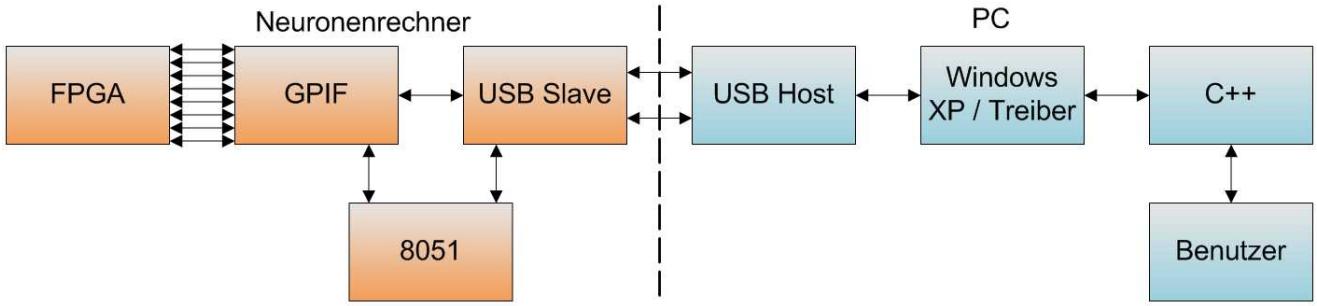


Abbildung 1: Die Datenstrecke

die zum Abholen benötigt wird. Die Firmware wurde von Cypress in Assembler geschrieben, um den Einfluss des 8051 so gering und bekannt wie möglich zu halten. Da keinerlei Datenverarbeitung oder -erzeugung stattfindet, die Datenmenge sehr klein ist und die Zeitmessung im 8051 stattfindet, wird hier die maximale Datenrate gemessen.

Das theoretische Maximum der USB 2.0 brutto Datenrate beträgt 60 MB/s. Mehrere Messungen ergaben eine durchschnittliche Datenrate von 55 MB/s, Dieses Ergebnis übertrifft die Anforderungen um ein Vielfaches und zeigt, dass die Strecke zwischen Windows XP und USB-Controller für das Nepteron Projekt mehr als geeignet ist. Die Streckenteile rechts und links davon (C++ und FPGA) werden nachfolgend gesondert untersucht.

4.2 Daten- und Fehlerrate mit FPGA

Nachdem die grundlegende Eignung des USB-Controllers festgestellt wurde, soll nun die gesamte Strecke untersucht werden. Dazu wurde ein C++ Programm geschrieben, dass Dummy-Daten vom FPGA liest und überprüft. Das Programm heißt SpeedTest und baut auf Knopfdruck eine Verbindung zum USB-Controller auf und liest eine größere Menge Daten. Anschließend berechnet es die Datenrate und überprüft, ob es Übertragungsfehler gab. Hierbei wird Bulktransfer genutzt, da er auf USB Hardware Ebene eine sichere Übertragung garantiert [6]. Um trotzdem auf Übertragungsfehler zu prüfen ist jedes der 512 Byte Pakete folgendermaßen aufgebaut:

- Byte 0 und 1: Fortlaufende Paketnummer
- Byte 2 bis 511: Byte 2 des ersten Pakets ist Null, alle folgenden werden um jeweils drei erhöht.

Dabei werden Paketgrenzen und Überläufe ignoriert. Das heißt Byte 2 von Paket 1 hat einen anderen Wert als Byte 2 von Paket 2 etc.. Nachdem der Startknopf gedrückt wurde nimmt SpeedTest Einstellungen am USB Gerät vor und zeigt in einer Konsole wichtige Daten wie zum Beispiel Art und Anzahl der Endpoints oder den Namen des Geräts an. Anschließend startet die Initialisierung der Testroutinen. Dabei werden unter anderem zwei Datenpuffer angelegt, die groß genug sind um alle Daten der folgenden Übertragung aufzunehmen. Einer dieser Puffer nimmt alle ankommenden Daten auf und der Andere wird mit Vergleichswerten initialisiert. Dies kann je nach gewünschter Datenmenge einige Minuten in Anspruch nehmen. Danach startet der eigentliche Lesevorgang, der möglichst knapp gehalten wurde. Schleifen, Abfragen und Ausgaben werden auf ein Minimum

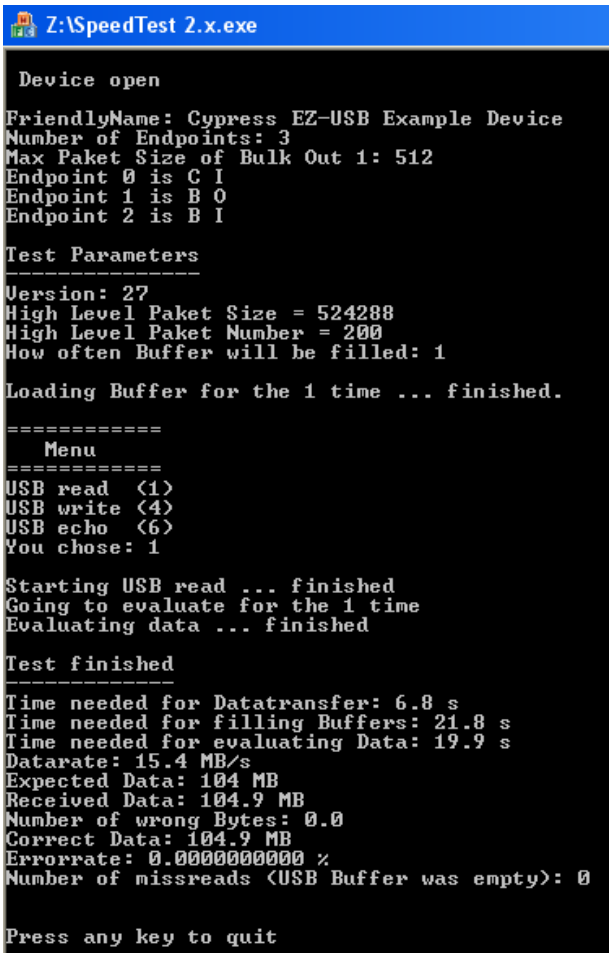


Abbildung 2: Messprotokoll SpeedTest2.7

begrenzt. Außerdem wurde darauf geachtet an dieser Stelle nur den C Teil von C++ zu verwenden.

Dabei laufen drei Messungen, die folgende Zeiten protokollieren:

- Initialisierung der Puffer
- Lesen der Daten vom USB
- Auswertung der Daten

Abb. 2 zeigt die Ausgabe einer solchen Messung. Dieser Test wurde mehrfach auch mit größeren Datenmengen wiederholt und unter zig Gigabyte fand sich nicht ein einziges falsches Byte. Damit gilt die USB-interne Datensicherung als ausreichend und es wird auf eine Fehlerbehandlung im Anwendungsprogramm verzichtet.

Um die geringe Datenrate zu erklären wurden verschiedene Messungen vorgenommen. Die Lösung brachte eine Untersuchung der GPIF Verbindung zwischen FPGA und USB-Controller. Abbildung 3 zeigt das Ergebnis einer Messung, bei der der FPGA vier Pakete überträgt und den Zustand „GPIF aktiv“ auf

einen Pin legt. Das GPIF wurde mit acht Datenleitungen betrieben. Die Puffer im USB-Controller waren dabei leer, um eine maximale Datenrate zu garantieren.

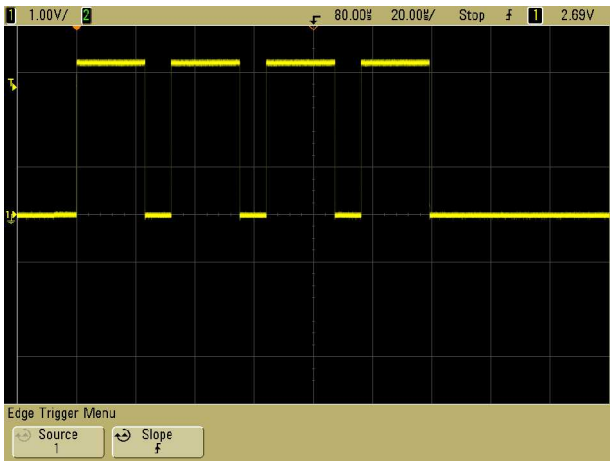


Abbildung 3: GPIF-Aktivität

Zu beachten ist vor Allem die lange Totzeit zwischen den Übertragungen. Vermutlich schalten hier die Zustandsautomaten von einem Sende-FIFO auf das andere um. Um die volle Datenrate zu erhalten muss der FPGA alle sechzehn Leitungen bedienen.

Bei Experimenten mit dem FX2LP-Board[2], einer Firmware, die laufend Dummy-Daten generiert bzw. annimmt wurden Datenraten von 30 MB/s beim Schreiben und 40 MB/s beim Lesen erreicht. Dabei kam weder GPIF noch ein FPGA zum Einsatz und daher werden diese Datenraten als praktisches Maximum des USB-Controllers angesehen.

Alles in Allem fällt der Test sehr gut aus. GPIF, der Cypress USB-Controller und die Cypress Bibliothek sind für das Vorhaben bestens geeignet.

4.3 Verarbeitungsdauer im PC

Die letzte Unbekannte in dem System ist der PC. Da Windows XP verwendet wird, steht die Echtzeitfähigkeit unter einem schlechten Stern. Die Daten müssen nicht nur regelmäßig abgeholt, sondern

```
Test finished
-------------
Time needed for Datatransfer: 49.0 s
Time needed for filling Buffers: 205.9 s
Time needed for evaluating Data: 194.8 s
Datarate: 21.9 MB/s
Expected Data: 1073 MB
Received Data: 1073.7 MB
```

Abbildung 4: SpeedTest mit Arrays

2 Exakter Name: CY3684 EZ-USB FX2LP Development Kit

auch so schnell verarbeitet werden, wie sie ankommen. Die Messungen aus Abb.2 zeigen, dass der einfache Vergleich der empfangenen Daten mit den Vergleichsdaten ca. drei mal so lang dauert wie das Empfangen. Um die Verarbeitungszeit zu minimieren wurde SpeedTest so modifiziert, dass als Puffer einfache Arrays verwendet werden. Wie in Abb.4 zu sehen ist, ändert sich wider Erwarten an der Verarbeitungsdauer nichts. Abb.5 zeigt einen Ausschnitt des typischen CPU-Lastverlaufs während eines „Lade Puffer – Lese USB – Vergleiche Puffer“ Zyklus'. Hier lässt sich leicht erkennen, dass der verwendete PC („Intel Core2Duo @ 2,66 GHz“, 3 GB RAM) während den Array-Operationen am Leistungsmaximum agiert. Die geringe Auslastung während der Lesephase erklärt sich dadurch, dass die Funktion XferData() erst zurückkehrt, nachdem die geforderte Datenmenge empfangen wurde. Während der Wartephasen ist dieses 1-Thread Programm also tatenlos.

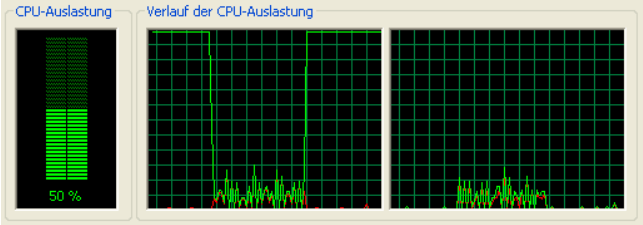


Abbildung 5: CPU Last SpeedTest

4.4 Ergebnis der Untersuchung

Der USB-Controller erfüllt in jeder Hinsicht die Anforderungen dieses Projekts. Der Test aus Kap.4.1 demonstriert seine Leistungsfähigkeit für die USB-Verbindung, die auch durch ein C++ Programm und Windows XP (s. Kap. 4.2) kaum geschmälert wird. Die GPIF-Verbindung zum FPGA arbeitet ebenfalls zufriedenstellend und erfüllt die Anforderungen.

Fraglich bleibt, ob eine Echtzeitverarbeitung mit einem PC möglich sein wird. Für die Datenverarbeitung ist ein Rechner nötig, der regelmäßig die USB-Daten abholt. Grundsätzlich ist Windows XP nicht dazu in der Lage. Allerdings hat Kap. 4.2 hat gezeigt, dass der PC die Daten wesentlich schneller abholen kann als sie von GPIF angeliefert werden. Dadurch kann er vollgelaufene Puffer bei Bedarf sehr schnell wieder leeren. Des Weiteren gibt es Mittel und Wege um, sich Echtzeitverhalten anzunähern. Da ein Versagen des Systems keine Leben gefährdet und da es sich um eine weiche Echtzeitanforderung handelt, ist ein Versuch Windows XP einzusetzen vertretbar.

5. Vorstellung der Lösung

Dieses Kapitel behandelt die Konzepte zur Lösung der Probleme aus Kapitel 4. Es geht darauf ein, warum echt-parallele Verarbeitung nötig ist und wie sie realisiert werden soll. Außerdem wird die MVC-D Architektur, der Datenfluss während der Simulation sowie die Klassenhierarchie beschrieben und zwei Möglichkeiten aufgezeigt, Windows XP echtzeitfähig zu machen.

5.1 Modularität durch MVC-D Architektur

Damit die Software leicht zu schreiben und zukunftssicher, d.h. änderbar ist wird sie modular aufgebaut.

Es handelt sich hier um eine Datenverarbeitung mit Benutzeroberfläche und deshalb wird das Modell-View-Controller (MVC) Konzept eingesetzt. Dabei wird ein Programm streng in die genannten drei Teile getrennt. Das Modell enthält alle Daten des Systems, bearbeitet sie aber nicht. View ist für die Darstellung des Modells verantwortlich. Das heißt es zeigt z.B. die Neuronendaten als xy-Graphen am Bildschirm an und wickelt die üblichen Windows-Operationen wie Speichern/Laden, Abbrechen etc. ab. Der Controller ist die einzige Instanz die die Daten des Modells manipuliert. Er führt also die eigentliche Arbeit aus. In diesem Programm nimmt er die Daten von der USB-Schnittstelle entgegen, dekodiert die Header und füllt damit das Modell. Außerdem sendet er vor dem Start der Simulation Einstellungs- und Initialisierungsdaten an den FPGA. MVC wird hier um eine Datenquelle/-senke erweitert, die dank fest definierter Schnittstellen ebenfalls leicht ausgetauscht werden kann. Abb. 6 zeigt eine Übersicht der Architektur. Die Rechtecke stellen Schnittstellen dar. Die roten Pfeile zeigen den Fluss der Nutzdaten von der Datenquelle (D) zur Darstellung (V) während einer laufenden Simulation.

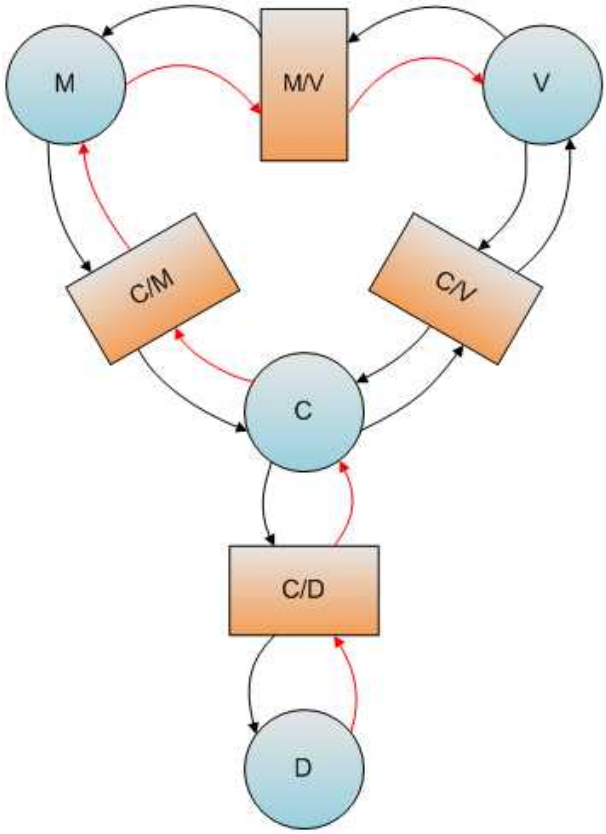


Abbildung 6: MVC-D Konzept mit Schnittstellen

Um die nötige Geschwindigkeit zu erreichen wird an der Schnittstelle Controller/Modell die strikte Trennung der vier Teile aufgehoben. Die Hauptarbeit des Programms ist es, Daten in das Modell einzusortieren. Da dies vom Controller erledigt wird muss er den schnellst möglichen, noch vertretbaren Zugriff auf das Modell haben. Der Controller schreibt deshalb direkt auf die Originaldaten. Die Modularität ist trotzdem gewährleistet, weil der Controller keine Kenntnis der internen Modellstruktur benötigt. Er bekommt das nötige Wissen zur Laufzeit von einer Interface-Funktion.

5.2 Echt-parallele Verarbeitung

Ein Anspruch des Nepteron Projekts ist es, dass die gesamte Simulation in Echtzeit abläuft. Dazu muss der PC die Daten genauso schnell verarbeiten, wie der FPGA sie produziert. Kapitel 4.3 hat gezeigt, dass ein einfacher Vergleich der Daten ca. drei mal so lang dauert wie der Empfang. Hierzu könnte ein PC mit QuadCore Prozessor eingesetzt werden. Ein Kern würde sich um den Datenempfang kümmern, während die anderen drei die Daten überprüfen. Dadurch wäre das Testprogramm zwar schnell genug, aber die eigentliche Datenverarbeitung ist wesentlich komplizierter und daher reicht ein QuadCore nicht aus.

Die finale Software wird auf einem Server mit zwei QuadCore-Prozessoren mit je 4 GB RAM getestet. Damit das Programm alle acht Kerne optimal nutzt, wird Intels ThreadingBuildingBlocks (TBB) eingesetzt. Diese Bibliothek erweitert C++ um Kontrollstrukturen wie „parallel_for“ und Container wie „concurrent_vector“ [7]. Dadurch wird es möglich Parallelität in C++ auszudrücken. Außerdem arbeitet der Anwendungsprogrammierer nicht mit Prozessen oder Threads, sondern mit Tasks. Der TBB-Scheduler übernimmt das Verteilen der Tasks auf die Prozessoren. Das ist nicht nur effektiver als ein Zuweisen per Hand, sondern auch selbst-skalierend. Das heißt, wenn das Programm auf einem 16 Kern

Computer ausgeführt wird nutzt es automatisch alle 16 Kerne. (Vorausgesetzt das Programm kann in 16 oder mehr nebenläufige Teile geteilt werden) Der Programmierer kümmert sich also nicht mehr um die Verwaltung seiner Hardware sondern um das eigentliche Problem.

Abb. 7 zeigt den Weg der Daten während einer Simulation. Dieses Diagramm ist eine Detaildarstellung des Controllers und zeigt, an welchen Stellen die Verarbeitung parallelisiert werden kann. Die orangefarbenen Rechtecke stellen spezielle TBB-Datencontainer dar. Sie sind für den gleichzeitigen Zugriff von mehreren Lesern/Schreibern geeignet und wachsen oder schrumpfen je nach Bedarf [7]. Die blauen Rechtecke stellen aktive Teile des Programms dar. Die transparenten Rechtecke deuten an, dass dieser Programmteil mehrfach ausgeführt werden kann. Wie stark die Parallelisierung an diesen Stellen ist, wird vom TBB-Scheduler entschieden. Seine Entscheidungen hängen davon ab, wie viele Daten noch auf ihre Verarbeitung warten, wie viele CPU-Kerne zur Verfügung stehen und wie der Programmierer den Code implementiert hat.

Die tatsächlichen Daten werden so wenig wie möglich bewegt. Wenn die Quelle etwas produziert, legt sie es im Rohdaten-Container ab. Dazu erstellt sie einen Array, den sie mit einem intelligenten Zeiger vom Typ TR1::shared_array [8][9] verwaltet. Diese Zeiger werden in den Rohdaten-Container eingehangen, wo sie darauf warten, von einer Vorsortierung abgeholt zu werden. Diese Funktionen nehmen sich einen oder mehrere Zeiger aus dem Container und dekodieren die Header. Wenn mit den Daten alles in Ordnung ist, müssen sie in in die RTP, SSP und FSP Container gelegt werden. Leider ist das Weiterreichen der Zeiger nicht möglich, da die ursprünglichen, recht großen Pakete eine Menge kleinerer Pakete enthalten, die nach den eben genannten drei Datentypen RTP, SSP und FSP sortiert werden müssen. Deshalb kopiert die Vorsortierung die Daten in die entsprechenden Container und löscht das Original. Das heißt, die Datenquelle füllt zwar den Rohdaten-Container, das Leeren übernimmt aber die Vorsortierung.

Nachdem die Daten nach Typ sortiert wurden, werden sie von den zuständigen Funktionen in das Modell geschrieben, womit der Controller seine Aufgabe erledigt hat. Das Modell entscheidet übrigens selbstständig, wann es Daten auf die Festplatte (HDD) auslagern muss. Dies wird nicht vom Controller übernommen, da die Festplatte als eine Erweiterung des Arbeitsspeichers verwendet wird. Für die Außenwelt ist es dadurch egal, wo die Daten liegen. Wenn jemand Daten lesen will, übergibt er an das Modell die nötigen Informationen wie Zeitpunkt und Neuronen-Adresse und bekommt anschließend das Gewünschte geliefert.

5.3 Klassenhierarchie

Um die MVC-D Architektur zu realisieren, wurden verschiedene ausführende Klassen entwickelt, die

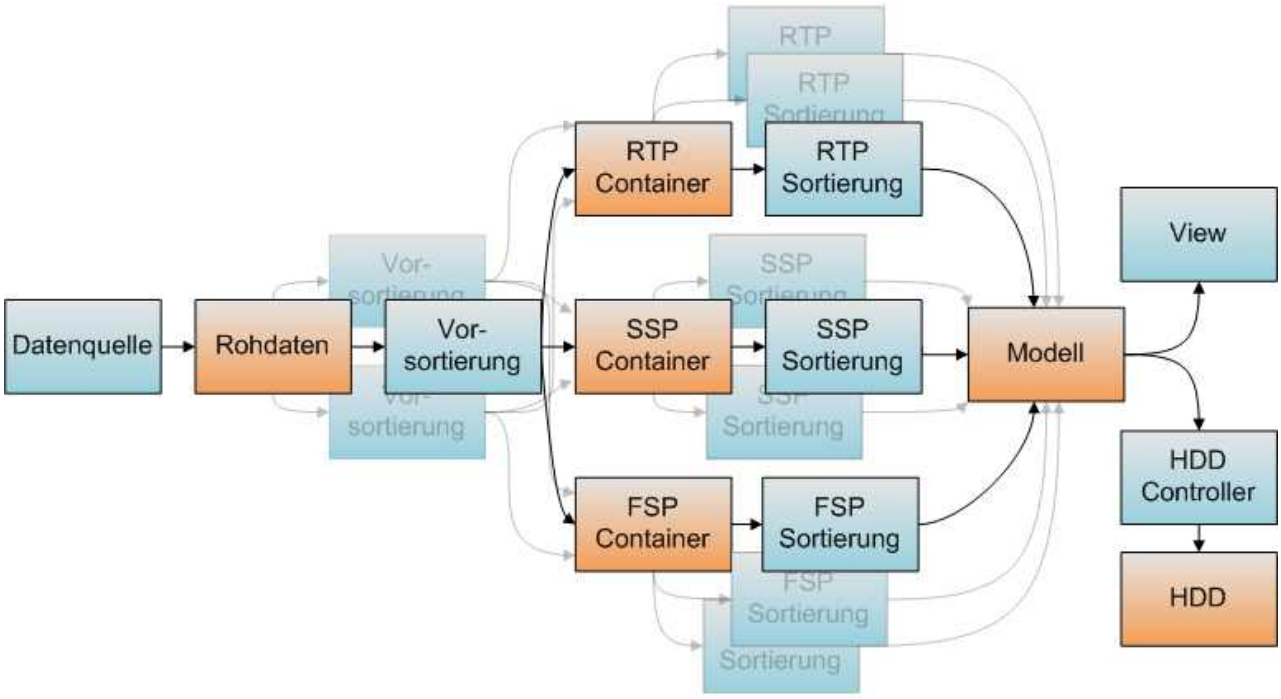


Abbildung 7: Parallele Datenverarbeitung

über Interface-Klassen miteinander kommunizieren. Im Rahmen dieses Papers kann nicht auf jede Klasse einzeln eingegangen werden. Es haben sich aber verschiedene Kategorien herauskristallisiert über die ein kurzer Überblick mit Beispielen gegeben wird.

5.3.1 Master

Das Datenmodell besteht im Kern aus den zwei Klassen Neuron und Cluster. Ein Objekt vom Typ Neuron enthält alle Informationen, die nötig sind, um ein Neuron im Sinne des Huber-Braun-Modells für einen Zeitpunkt zu beschreiben. Ein Cluster enthält so viele Neuronen wie nötig, um das Netz eines Neuronenkerns abzubilden, aktuell 400. Es existiert also für jeden Simulationszeitpunkt ein Cluster, das alle Daten enthält. Beim Zugriff auf das Modell kann wie durch Karteikarten geblättert werden, bis der gewünschte Zeitpunkt erreicht ist. Dieses Blättern übernimmt ein Objekt vom Typ ClusterMaster. Er erzeugt und löscht Cluster und führt eine Liste aller existierenden Cluster. Auf Anfrage nimmt er einen Zeitpunkt entgegen und liefert einen intelligenten Zeiger auf den passenden Cluster zurück. Er ist die in Kap. 5.1 erwähnte Schnittstelle zwischen Modell und Controller. Weitere Master sind: TaskHDDMaster und VideoDataMaster.

Allgemein ist in diesem Projekt ein Master eine Instanz, die Ressourcen verwaltet und Dienste zur Verfügung stellt. Master können als eigenständige Tasks laufen oder untätig im Arbeitsspeicher liegen, bis sie aufgefordert werden etwas zu tun.

5.3.2 Tasks

Im Sinne von TBB sind Tasks einzelne, unabhängige Aufgabenpakete [10]. Der TBB-Scheduler verteilt sie automatisch auf die freien CPU-Kerne.. TBB kennt zwei Möglichkeiten neue Tasks zu erzeugen, einmal automatisch durch den TBB-Scheduler. Wenn etwa eine parallel_for Schleife aufgerollt wird, dann laufen die aufgerollten Teile parallel als einzelne Tasks. Ist ein Teil beendet wird das Ergebnis zurückgegeben und der Task automatisch geschlossen. Die zweite Möglichkeit ist, dass der Programmierer selbständig Tasks anlegt. Diese laufen wann immer der Scheduler ihnen Rechenzeit zugesteht und enden, sobald ihre persönliche Abbruchbedingung zutrifft. Sie sind vor Allem deshalb interessant, weil damit Code geschaffen werden kann, der ständig läuft. Grundsätzlich bekommen alle vier Teile des MVC-D einen eigenen Task. Diese können je nach Bedarf Nachkommen erzeugen, was der Controller wie in Kapitel 5.2 zu sehen ist auch rege tut. Unter Anderem sind folgende Tasks zum Betrieb der Software nötig: TaskDecodePipeline, TaskHDDMaster, TaskUSBReader und TaskTXTReader. Sie werden je nach Betriebszustand gestartet und beendet.

5.3.3 Interfaces

Die Objekte dieses Klassentyps bestehen hauptsächlich aus einem großen Datencontainer und führen noch einige Methoden zu dessen Verwaltung mit. Teile von View müssen beispielsweise auf Daten aus dem Modell zugreifen, um etwa einen Spannungsverlauf auf den Bildschirm zu zeichnen. Diese Daten jedes Mal aus dem Modell herauszusuchen, wäre sehr aufwendig. Deshalb wird die Klasse RealTimeVideoData eingeführt. Objekte dieses Typs beinhalten alle nötigen Daten, um einen kompletten Verlauf auf den Bildschirm zu zeichnen. View liest diese Objekte regelmäßig aus und aktualisiert damit die Graphen. Wenn ein anderer Ausschnitt gezeichnet werden soll, müssen nur die Daten im entsprechenden Interface-Objekt geändert werden, und View zeichnet automatisch ein anderes Bild. Pro Graph, der gezeigt werden soll, existiert also eine Instanz dieser Klasse. Weitere Interface-Klassen sind CoreSetup und CoreState. Sie beinhalten die Daten zum Einstellen des Neuronenkerns bzw. seinen aktuellen Zustand. Dadurch wird das Interface Controller/View realisiert. Außerdem sind noch folgende Interface-Klassen geplant: RawDataStorage, SnapshotVideoData und FireStatusVideoData.

5.4 Echtzeitfähigkeit von Windows XP

Grundsätzlich gibt es keine Garantie, wann Windows einem Programm Rechenzeit zugesteht. Dadurch gilt es als unmöglich, zeitkritische Anwendungen auf einem PC mit Windows-Oberfläche zu betreiben. Während der Recherche zu diesem Projekt entwickelte der Autor allerdings zwei Wege, die das Ziel - nämlich ein ständiges, kontrolliertes Abarbeiten von beliebigen C++ Code - erreichen könnten.

Eine kurze Einführung: Um mehrere Aufgaben scheinbar gleichzeitig auf einem Prozessor auszuführen, wurden Prozesse, Threads, der Windows-Scheduler u.v.m. eingeführt. Jedes aktive Programm läuft unter Windows als Prozess. Der Windows-Scheduler entscheidet wann welcher Prozess Rechenzeit erhält. Jeder Prozess bekommt seinen eigenen Speicherbereich mit virtuellem (mapped) Adressraum zugewiesen. Deshalb kann zwar jeder Prozess bis zu 4 GB Arbeitsspeicher adressieren ohne einen anderen zu stören aber er kann auch nicht auf die Daten eines anderen Prozesses zugreifen; und der alte BTX-Hack, bei dem ein Anwendungsprogramm den Return-Stack des Kernels überschreibt, funktioniert nicht mehr. Nur der Kernel arbeitet mit realen RAM-Adressen. Jeder

Prozess hat seinen eigenen Return-Stack, den er bei jedem Kontextwechsel lädt und speichert. Ein Prozesswechsel dauert recht lang und Inter-Prozess-Kommunikation ist ebenfalls aufwändig. Deshalb wurden Threads eingeführt. Sie sind voneinander unabhängige Ausführungspfade eines Prozesses. Ein Prozess kann beliebig viele Threads beinhalten und steuert selbst wann welcher aktiv ist. Dadurch ist es möglich, dass ein einziger Prozess z.B. „gleichzeitig" Musik spielt, einen Film zeigt und auf Tastatureingaben reagiert. Threads teilen sich sowohl die CPU-Zeit als auch den Speicher des Vaterprozesses. Daher können sie untereinander sehr schnell kommunizieren, und sie müssen keinen vollständigen Kontextwechsel ausführen, wenn von einem Thread auf einen anderen umgeschaltet wird. Auf einem System, mit zwei oder mehr Prozessorkernen können Threads echt parallel bearbeitet werden.

5.4.1 Prozessprioritäten stehlen

Um besser entscheiden zu können, wann welcher Prozess CPU Zeit bekommt, verteilt Windows Prioritäten. Diese sind in acht Stufen unterteilt, wobei REALTIME_PRIORITY_CLASS die höchste und interessanteste ist, sie hat sogar Vorrang vor Betriebssystemprozessen. Mittels SetPriorityClass kann man diese Prioritätsstufen ändern [11]. Ein Programm mit der höchsten Priorität kann auf einem 1-Prozessor-PC verhindern, dass die Festplattencaches geleert werden oder dass auf die Tastatur reagiert wird. Vermutlich nutzen Diskettenlaufwerke diese Priorität, wenn sie Daten lesen oder nach einer Diskette suchen. Bei Multi-Prozessor Systemen ist ein einziges Programm dieser Priorität problemlos einsetzbar, da für Betriebssystemaufgaben immer noch andere Prozessoren zur Verfügung stehen.

Um sicher zu gehen, dass das eigene Programm nicht von fremden verdrängt wird, muss es das einzige sein, das diese hohe Priorität besitzt. Um das zu erreichen kann mit der WinAPI-Funktion CreateToolhelp32Snapshot eine Liste aller Prozesse erstellt werden [12]. Anschließend wird ein Handle mit dem SeDebug Privileg für jeden Prozess benötigt. Unter [13] wird erklärt wie solche Handels erzeugt werden und [14] gibt nähere Information zum Thema „Prozesssicherheit und -zugriff". Durch das SeDebug Privileg ist es u.A. möglich, die Prioritäten fremder Prozesse nach Belieben zu ändern. Dadurch kann man das eigene Programm zum einzigen mit der Priorität „REALTIME_PRIORITY_CLASS" ernennen. Da der Windows Scheduler immer zuerst die Prozesse mit der höchsten Priorität fragt, ob sie CPU Zeit benötigen muss man so niemals auf benötigte Rechenzeit verzichten

An dieser Stelle sei erwähnt, dass auch auf diese Weise kein Programm völlig unterbrechungsfrei laufen wird. Der Windows-Scheduler stellt zwar immer fest, dass unsere Anwendung die höchste Priorität hat, aber dazu benötigt er eben CPU Zeit.

Es ist ratsam, eine Kopie der Snapshot Liste zu erstellen, bevor man etwas ändert. Dann kann vor Beenden des Programms oder während einer Fehlerbehandlung der alte Systemzustand wieder hergestellt werden.

5.4.2 CPU Affinität manipulieren

Für dieses Verfahren benötigt man ebenfalls eine Liste aller Prozesse und SeDebug Privilegien. Außerdem muss das System mindestens zwei Prozessoren vorweisen können.

Mit Hilfe der WinAPI Funktion SetProcessAffinityMask aus [15] kann einem Prozess der Zugriff auf einzelne Prozessoren untersagt werden. Angenommen das vorliegende System habe vier Prozessoren. Dann könnte man z.B. allen fremden Prozessen den Zugriff auf Prozessor Vier verbieten. Dem eigenen Programm wiederum versagt man die Prozessoren Eins, Zwei und Drei. Dadurch kann es nur noch auf Vier laufen, der allen anderen ja verboten ist und muss diesen Prozessor mit niemandem teilen.

Des Weiteren ist es möglich SetThreadAffinityMask [16] zu verwenden. Dabei könnte man alle fremden Prozesse auf Prozessor Eins verbannen und die restlichen drei Prozessoren an seine eigenen Threads verteilen.

Für die Nepteron Software soll exakt dies gemacht werden. Allerdings mit Hilfe von TBB. Das heißt mittels CPU Affinitäten werden alle fremden Prozesse auf den Kern Eins der CPU1 verbannt. Anschließend kann der TBB-Scheduler die restlichen 7 Kerne nach Belieben für sich verwenden.

6. Fazit

Zusammenfassend lässt sich sagen, dass die Strecke den Anforderungen genügt und für das Nepteron Projekt eingesetzt werden kann. Der USB-Controller überrascht mit Datenraten, die äußerst nah am theoretischen Maximum von USB 2.0 liegen. Die mitgelieferte Bibliothek CyAPI ist gut strukturiert und dank der professionellen Dokumentation [4] kann man sich gut einarbeiten und innerhalb weniger Stunden erste C++/USB Verbindungen aufbauen. Der Chip wird zur Zeit zwar durch den GPIF Betrieb mit 8 statt 16 Datenleitungen ausgebremst, bringt aber trotzdem

die nötigen 15 MB/s. Ob der aktuelle PC die benötigte Rechenleistung bringt, bleibt abzuwarten. Falls nicht, könnte er beispielsweise mit zwei Corei7 CPUs aufgerüstet werden, die dann 16 Kerne zur Verfügung stellen.

Der modulare Aufbau der Software wird dafür sorgen, dass das Programm leicht erweitert werden kann, Teile können geändert oder ersetzt werden, ohne andere Teile ändern zu müssen und schließlich lässt sich jeder Teil einzeln testen. Das langfristige Ziel des Nepteron Projekts ist es mehrere Neuronenkerne auf einem ASIC zu realisieren, die mittels einer oder mehrerer USB 3.0 Leitungen mit einem PC verbunden werden. Für dieses Ziel ist der vorliegende Entwurf gut vorbereitet. Einerseits kann die aktuelle Datenquelle problemlos durch eine oder mehrere neue ersetzt werden - sie müssen lediglich den Rohdatencontainer wie gehabt befüllen. Andererseits steigt mit der Datenmenge auch der Anspruch an die Rechenleistung des PCs. Der gesamte Controller (siehe Abb.7) kann dazu mehrfach ausgeführt werden. Dies ist leicht möglich, weil er als TBB Pipeline [17] implementiert wird. Um sie zu duplizieren, muss lediglich eine neue Instanz von ihr erzeugt und verwaltet werden. Da alle Container als „Multiple Writer / Multiple Reader“ Speicher realisiert werden, brauchen sich zukünftige Programmierer um den Speicherzugriff keine Sorgen mehr machen, auch wenn bei zwölf Stunden Simulation über 600 GB Daten verarbeitet werden.

Alles in allem sollte dieser Plan die Anforderungen an Wartbarkeit und Leistungsfähigkeit erfüllen, wodurch seiner Implementierung nichts mehr im Wege steht.

7. Quellen

Alle Internetadressen: Juli 2010, passende Suchbegriffe werden mit angegeben

[1] Huber, M.T.; Braun, H.A.: „Stimulus – response curves of a neuronal model for noisy subthreshold oscillations and related spike generation“; University of Marburg, 2005.

[2] Beuler, M.; Bonath, W.: „Echtzeitfähige Berechnung der Modellgleichungen von Huber-Braun-Neuronen auf einem Virtex-4-FPGA“ http://dok.bib.fh-giessen.de/opus/volltexte/2009/4150/

[3] Cypress: Introduction to the EZ-USB FX2™ GPIF Engine – AN1115:http://www.cypress.com/?rID=12934

[4] USBSuite: http://www.cypress.com/?rID=34870

[5] Cypress: „Measuring Delivered USB 2.0 Bandwidth with an EZ-USB FX2™ Development Board“ http://www.cypress.com/?docID=4383

[6] Kelm, „USB 2.0 Studienausgabe“, Franzis Verlag, S.61

[7] Intel(R) Threading Building Blocks Reference Manual S.102ff. http://www.threadingbuildingblocks.org/uploads/81/91/Latest%20Open%20Source%20Documentation/Reference.pdf

[8] Meyers, „Effektiv C++ programmieren“, ADDISON-WESLEY, S. 81 ff.

[9] Boost.org, smart_ptr: http://www.boost.org/doc/libs/1_43_0/libs/smart_ptr/smart_ptr.htm

[10] Intel® Threading Building Blocks Tutorial S.63 ff. http://www.threadingbuildingblocks.org/uploads/81/91/Latest%20Open%20Source%20Documentation/Tutorial.pdf

[11] Prozessprioritäten ändern, SetPriorityClass http://msdn.microsoft.com/en-us/library/ms686219%28VS.85%29.aspx

[12] Liste aller Prozesse erstellen, CreateToolhelp32Snapshot http://msdn.microsoft.com/en-us/library/ms682489%28VS.85%29.aspx

[13] „How to obtain a handle to any process with SeDebugPrivilege“ http://support.microsoft.com/kb/131065/en-us?fr=1

[14] „Process Security and Access Rights“ http://msdn.microsoft.com/en-us/library/ms684880%28VS.85%29.aspx

[15] WinAPI Funktion, SetProcessAffinty http://msdn.microsoft.com/en-us/library/ms686223%28VS.85%29.aspx

[16] WinAPI Funktion, SetThreadAffinityMask http://msdn.microsoft.com/en-us/library/ms686247%28VS.85%29.aspx

[17] wie [7], jedoch S. 46 ff.

A Compact 12-bit Current-Steering D/A Converter for HDRC® Image Sensors

Tarek Zaki, Tarek Hussein, Cor Scherjon

Institut für Mikroelektronik Stuttgart, Allmandring 30a, 70569 Stuttgart

Telefon 0 711 / 21 855 -10, Fax -100, E-Mail zaki@ims-chips.de

A compact 12-bit current-steering D/A converter in 0.18 µm CMOS technology is presented. The D/A converter is designed to be integrated as a two-step ramp generator in a new HDRC® image sensor with global shutter functionality. A high segmentation level (8-4) has been utilized to guarantee monotonicity and achieve better static and dynamic performances. The simulated integral and differential linearity errors are less than ±1LSB and ±0.5LSB respectively at an update rate of 25 MHz. A triple-centroid switching scheme has been exploited to circumvent the linear and symmetrical gradient errors. The D/A converter employs a newly developed 1-dimensional fully custom 8-bit thermometer decoder based on an iterative architecture. With a careful "design for layout" approach, an active current-source array area of 0.2 mm² has been accomplished.

1. Introduction

Digital-to-analog converters are fundamental devices used in data processing systems. They serve as a conversion interface to construct analog signals from the digital input codes. The basic principle of a D/A converter circuit is to generate a multiple of a certain reference quantity for each digital input code. The D/A converter's transfer function can be expressed as:

$$Y_{out}(D) = \sum_{m=0}^{N-1} 2^{m} D_m Y_{ref} \qquad (1)$$

Where D is the digital binary input code, Y_{ref} is the reference quantity and N is the D/A converter's resolution. The D/A converters are classified into three main classes namely; the resistor, the capacitor and the current-steering D/A converters. This classification is recognized from the reference quantity used. The corresponding D/A converter's output signal can either be a voltage, a charge or a current, respectively.

For each D/A converter class, various implementations ranging from high accuracy to high speed architectures are introduced in literature. These numerous alternative designs introduce challenges for selecting the perfect architecture for a given application. However, there will be always a tradeoff between the different possible architectures.

In general, process tolerances, speed, silicon area, accuracy and glitch considerations rule out the resistive and the capacitive based architectures for resolution above 10-bit. This leaves the current-steering architecture to be the most suitable choice for our 12-bit and compact D/A converter.

This paper is organized as follows. The HDRC® image sensor application for the D/A converter is presented in Section 2. The design and the floorplan of the segmented current-steering D/A converter architecture are provided in Section 3. The switching scheme used in our D/A converter is discussed in Section 4. In Section 5, the newly developed 1-dimensional fully custom 8-bit thermometer decoder based on an iterative architecture is presented. The layout of the current-source array is introduced in Section 6. Simulation results are presented in Section 7. Finally, conclusions are given in Section 8.

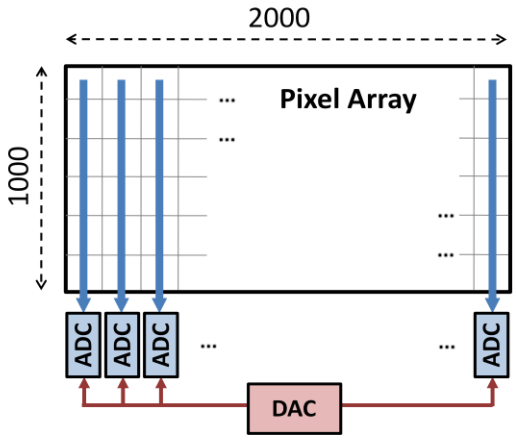


Fig. 1: Simplified diagram for the HDRC® image sensor.

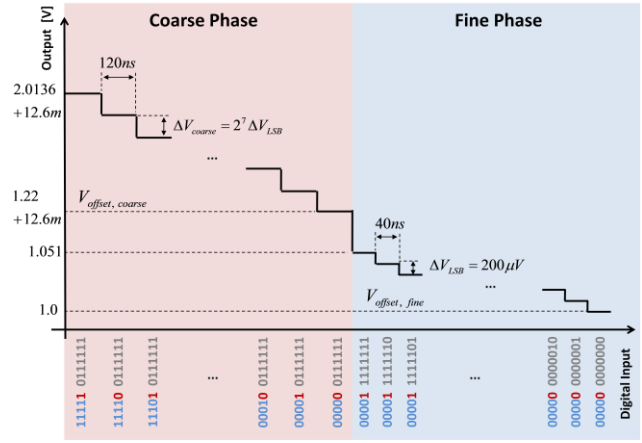


Fig. 2: Desired two-step ramp signal.

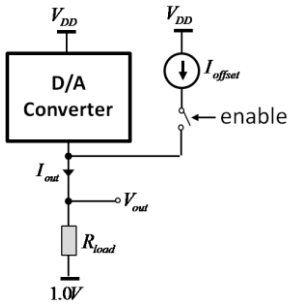


Fig. 3: D/A converter and the offset circuits.

2. HDRC® Image Sensor Application

The D/A converter is designed to be integrated in a new HDRC® image sensor with global shutter functionality (snapshot). More specifically, it is designed to serve as a two-step ramp generator for the A/D converters which are used to convert the analog voltage of the pixel array. For this reason, as depicted in Fig. 1, multi-parallel A/D converters are used to convert the value of each column simultaneously. The idea of parallelism in this case reduces the conversion time significantly. However, a single D/A converter is used to drive all the parallelized A/D converters, this is to minimize mismatch problems.

The operation of the A/D converters is divided into two phases; a coarse and a fine phase, where each phase requires a different ramp signal. Fig. 2 shows the two-step ramp signal desired by the A/D converters and the corresponding digital input code sequence. The D/A converter should be designed to be capable of providing successfully the required two-step ramp signal. As depicted in Fig. 2, the ramp signal has two different offsets; *1.22V+12.6mV* and *1V* in the coarse phase and the fine phase, respectively.

The idea developed to provide these two interchangeable offsets for the output ramp signal of the D/A converter is introduced in Fig. 3. The circuit employs a PMOS based current-steering D/A converter and a resistive load. The resistive load acts as a linear current-to-voltage converter. The main idea is to allocate the lower offset value of *1V* to the supply line to have the necessary output in the fine phase. Note that the supply line should be large enough to avoid fluctuations and to allow a large output current up to *20mA* to flow. While the additional switched offset current I_{offset} is enabled only in the coarse phase to provide additional *0.22V* offset value at the output. The additional offset current can be expressed as:

$$\mathrm{I_{offset}} = \frac{0.22\mathrm{V}}{\mathrm{R_{load}}} \qquad (2)$$

Thus, the overall offset voltage in the coarse phase would be 1.22V as desired. Moreover, it was shown earlier in Fig. 2 that during the coarse phase, the input 6 LSBs are always enabled. This is to shift the coarse ramp signal by 12.6mV as anticipated.

3. D/A Converter Architecture

3.1. D/A Converter Design

Fig. 4 shows the segmented current-steering architecture that is used in our D/A converter with the eight most significant bits (MSBs) converted using unary current sources, and the four least significant bits (LSBs) converted using binary-weighted current sources. The segmentation level has been chosen according to an area approach presented in [1], a high segmentation level (8-4) has been utilized to guarantee monotonicity and achieve better static and dynamic performances. For a resolution of 12 bits, the 8 MSBs are converted to 255-bit thermometer code and driven to the unary current sources, while the remaining 4 LSBs are directly driven to the binary-weighted current sources. The implementation of the 8-bit thermometer decoder is going to be presented later in Section 5.

The schematic diagram of the PMOS based unary current cell is depicted in Fig. 5. A common biasing current source (M0) is used to mirror the current to all the current source transistors (M1, M2, .., M16). As shown in the figure, the current source $16I_{LSB}$ is implemented using 16 identical parallel transistors

with value I_{LSB}. Same applies for all other current sources; this is to reduce the mismatching effects. Moreover, the switches should have the same multiplicity as their corresponding current sources.

The sizing of the current source transistor depends on the required INL-yield and the chosen overdrive voltage $(V_{GS} - V_{TH})$. For an INL-yield of 99.7% on an INL specification of ±0.5LSB according to the equation derived in [1], the relative standard deviation of the LSB unit current source is:

$$\frac{\delta(\mathrm{I_{LSB}})}{\mathrm{I_{LSB}}} = 0.25\% \qquad (3)$$

This leads to the following area requirement for the LSB current source transistor according to the mismatch model derived in [2]:

$$\mathrm{WL_{min}} = \frac{1}{2(\delta(\mathrm{I_{LSB}})/\mathrm{I_{LSB}})^2}\left(A_\beta^2 + \frac{4A_{VT}^2}{(V_{GS} - V_{TH})^2}\right) \qquad (4)$$

where A_β and A_{VT} are the mismatch parameters introduced by the foundry.

It is proposed to design the switch transistors (M17, M18) to be operating always in the saturation mode to provide a shielding effect to the current source transistors (cascode structure). This shielding effect is defined to be; the isolation of the drain nodes of the current source transistors from the varying output node. This adds additional constraints to the design of the transistor's dimensions to ensure a proper operating mode during the whole output voltage range.

3.2. The Floorplan

Strict requirements arrive on the floor planning and the layout due to the use of a thermometer decoder and large number of current sources that require complex routing. Therefore, several approaches have been presented in literature to arrange the current sources for example to a matrix like layout. Moreover, switching schemes are needed to compensate the systematic errors introduced by linear and/or symmetrical gradients on the chip to guarantee a good static performance. The matrix distribution and the switching scheme control the number and the order of which the current sources are switched to the output. The switching scheme will be discussed in more detail in the next section.

There are two possible approaches for the floorplan of the D/A converter, either the switch array (SA) is integrated with the current-source array (CSA) or they

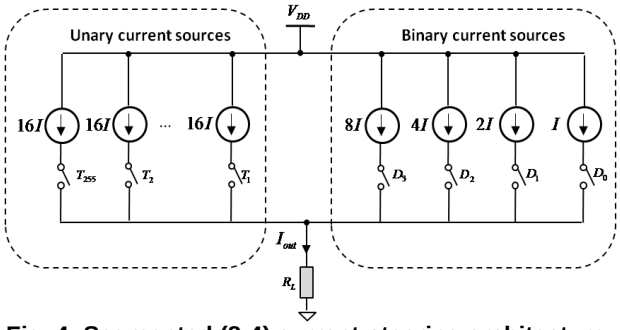


Fig. 4: Segmented (8-4) current-steering architecture.

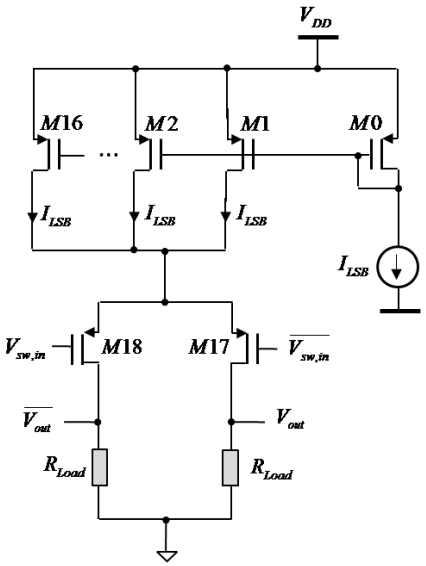


Fig. 5: Schematic diagram of the unary current cell.

are separated. The later choice is preferred to have a clear separation of the digital and the analog part of the D/A converter. Furthermore, the separated approach would make the CSA more compact, leading to a design that is more uniform and less susceptible to gradient errors.

As discussed in the previous subsection, for the (8-4) segmentation level, each unary current source consists of 16 parallel transistors that are switched simultaneously to the output providing the necessary $16I_{LSB}$. For this reason, the CSA is designed to have a 16-square matrix distribution (Fig. 6), where the 16 identical transistors are uniformly distributed among the 16 identical squares. This distribution guarantees a better compensation for the systematical linear and symmetrical gradient errors as depicted in Fig. 6.

The block diagram of the whole D/A converter is shown in Fig. 7. The CSA is surrounded by couple of dummy rows and columns of current source transistors at the edges of the active array to achieve an identical environment for all the active transistors.

This also contributes to the compensation of systematical errors. Depending on the offset current needed in the coarse phase (I_{offset}), some of these dummy current source transistors are utilized to provide the necessary offset current.

4. Switching Scheme

Switching schemes are used to compensate errors that are introduced by linear and/or symmetrical gradient systematical errors that could arise due to process-, temperature- and electrical variations. These errors have to be taken into consideration because they contribute to the static performance of the D/A converter. By reducing these systematical errors using the switching scheme, the random errors will dominate. However, the random errors are kept within the desired boundary ($INL < 0.5LSB$) by adjusting the active current-source transistor area (WL_{min}).

There are various switching schemes that are available in literature. However, by focusing on the 16-square CSA implementation, the random walk [3], double centroid [4] and triple centroid [1] switching schemes are the valid schemes. Since the random switching scheme introduces hard routing problems which may require extra silicon area, therefore, it is excluded from the comparison. Remain the double and the triple centroid switching schemes that are shown in Fig. 8. Referring to [4] and [1], the triple centroid achieves a better integral non-linearity value of ±0.3LSB compared to ±0.5LSB for the double centroid. Moreover, the D/A converters in [4] and [1] have segmentation levels of (8-4) and (5-7), respectively, where lower segmentation levels should have a degraded INL value. However, the D/A converter that employs the triple centroid switching scheme attains a better INL value although a lower segmentation level has been utilized. This proves that triple centroid has a preferable performance. Consequently, the final decision for the most suitable switching scheme for our D/A converter is the triple centroid.

5. Iterative Thermometer Decoder

The decoder logic can be implemented either by using the standard cell libraries or by a fully custom designed logic. In both cases, special care has to be taken for optimizing the timing constraint. This is to improve the operating speed of the decoder. Our D/A converter requires an 8-bit thermometer decoder for the unary current-sources.

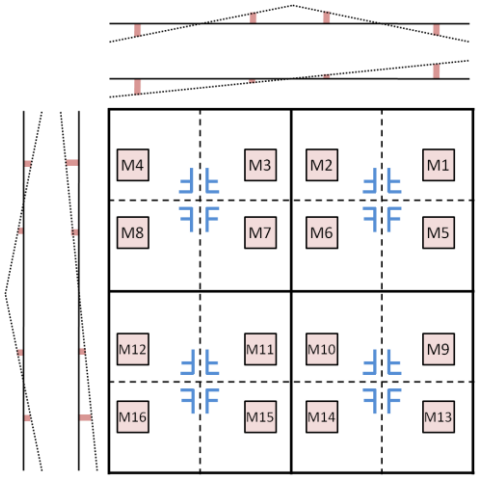


Fig. 6: CSA with 16-square distribution and clarification for the linear and symmetrical gradient compensation.

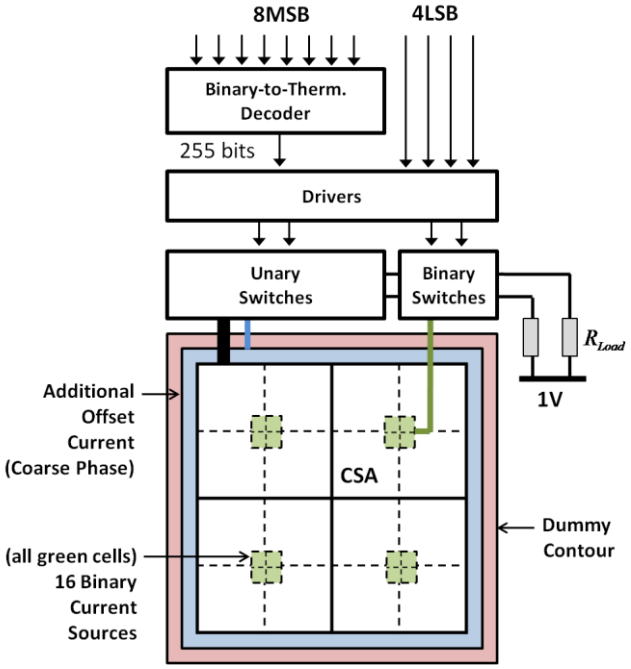


Fig. 7: Block diagram of the whole D/A converter.

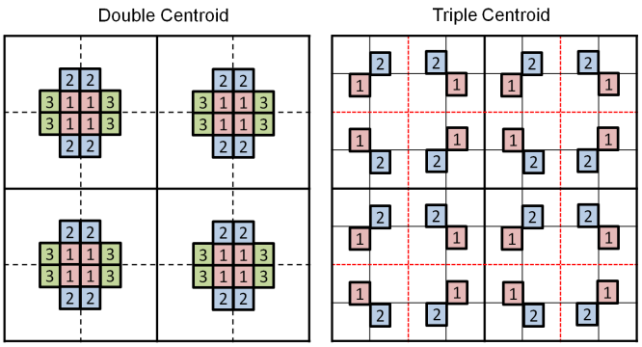


Fig. 8: Double and triple centroid switching scheme.

There are three main categories for the thermometer decoders namely; one-dimensional decoder, two-dimensional row/column decoder and multi-dimensional matrix decoder. Fig. 9 shows the block diagram of the three categories for an 8-bit binary coded input (used in 8-4 segmented D/A converters).

First, the 1D decoder requires no additional logic inside the array. This is because each of the 256 switch cells inside the array is driven by a single bit from the decoder. This makes this implementation smaller in area for large number of binary coded input bits. However, it is more complex than the other two categories, due to the large number of inputs. Second, the idea of the 2D decoder is to split the large 1D decoder into two smaller decoders to have a reduced complexity and delay. However, additional logic for each cell inside the array is used to decide upon the activity of the cell depending on the row and column signals. As a result, the array area is increased and the matching properties are degraded. Third, the multi-dimensional matrix decoder is just an extension of the 2D decoder. The main target of this scheme is to achieve a very high speed decoder at very high resolutions at the cost of larger area. However, it can be easily deduced that this decoding scheme is only useful for very high resolutions (>12bits). Finally, this makes the 1D decoder scheme to be the most suitable for our application due to the smaller array area and better matching properties.

The 8-bit 1D thermometer decoder can be implemented easily by the conventional ROM based design. The design simply consists of a 1-of-n decoder plus additional OR gates at the output for generating the desired thermometer output code. However, a non-redundant logic design that is shown in Fig. 10 proved to have a smaller area and delay. The design has an iterative phenomenon that can be easily deduced from the Boolean equations, where the 2-bit thermometer decoder is common in all equations. Each additional input bit requires only an additional row of NAND/NOR gates at the output. Fig. 10 shows the generalized n-bit iterative thermometer decoder design. This scheme simplifies the effort for designers to develop higher order thermometer decoders in a simple systematic approach while attaining the minimum possible silicon area and delay.

6. Layout

The Layout is designed using the Cosmos tool from Synopsys, and the layout design rule check (DRC) is done using Synopsys Hercules. Transistor models from the United Microelectronics Corporation (UMC) are used.

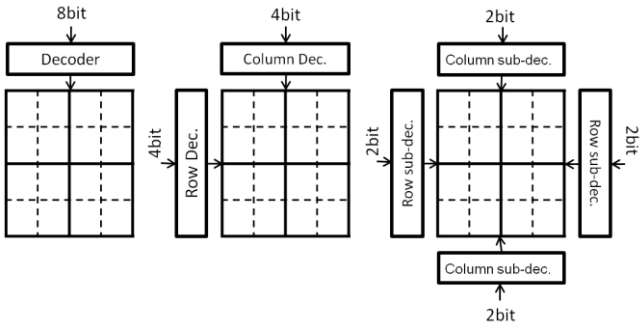


Fig. 9: Thermometer decoder categories: (left) 1D decoder, (middle) 2D row/column decoder, (right) multi-dimensional matrix decoder.

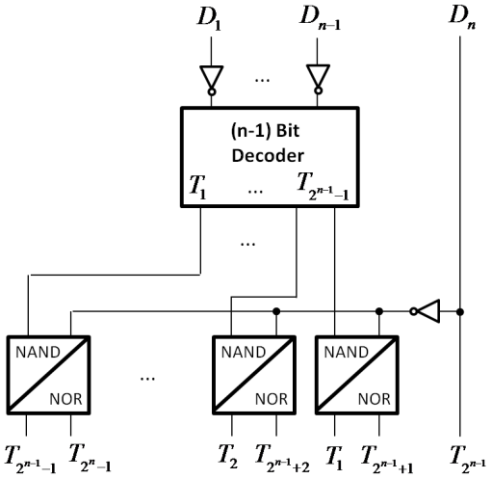


Fig. 10: n-bit iterative thermometer decoder.

Since the CSA consumes the major area of the whole D/A converter circuit, special care has been involved to minimize the area. Reducing the area of the CSA makes the current-source transistors more uniform and potentially less susceptible to mismatch effects.

The layout is implemented in a way to allow complete flexibility in the arrangement of the 255 unary and 4 binary inputs. With this approach, the area of the D/A converter is dramatically reduced due to the elimination of the routing matrix between the SA and the CSA. An additional advantage of this implementation is that it gives a complete flexibility for designing a compact thermometer decoder layout without constraints on the output bit arrangement.

Fig. 11 shows the layout diagram of the complete CSA. The total CSA area is found to be around *0.2 mm*^{2}. Noting that the minimum complete 12-bit D/A converter area noted in literature is *0.44 mm*^{2} [3]. Since the CSA consumes the largest area of the whole D/A converter, it is expected that our 12-bit D/A converter will consume a smaller silicon area.

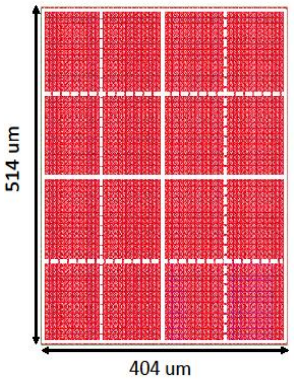


Fig. 11: CSA Layout.

Resolution	12bits
Digital Supply	1.8 V
Analog Supply	3.3 V
Update rate	25 MHz
Output Swing	1 V
INL [LSB]	±0.0026
DNL [LSB]	±0.877
CSA Area	0.2 mm^2

Table 1: D/A converter characteristics.

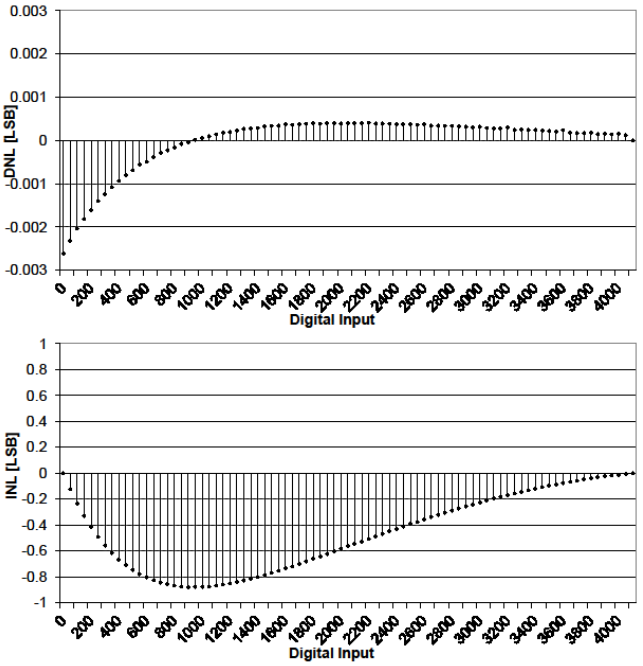


Fig. 12: Simulated DNL and INL.

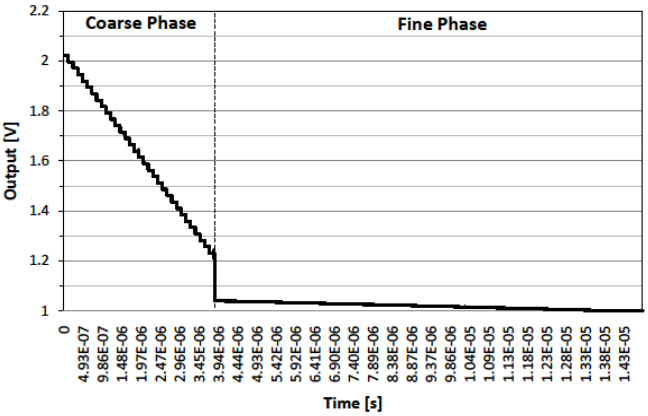


Fig. 13: Simulated two-step ramp signal.

7. Simulation Results

The transistors were designed to withstand the large offset voltage of approximately *1.22V* without changing their mode of operation from the desired saturation mode. Simulation results are prominent and the DC operating points are checked to be as wanted.

The static linearity errors of the D/A converter are simulated at an update rate of *25 MHz*. Fig. 12 shows the simulated differential non-linearity (DNL) and the integral non-linearity (INL), respectively. The maximum DNL is found to be -0.0026LSB, while the maximum INL is found to be -0.877LSB. Moreover, the unit current source area is taken $1.5WL_{min}$ to reduce the expected random mismatch effects after fabrication. Fig. 13 shows the simulated two-step ramp signal, the graph conforms to the required two-step ramp signal shown earlier in Fig. 2. Finally, Table 1 summarizes the D/A converter characteristics

8. Conclusion

In this paper, a segmented (8-4) current-steering D/A converter in *0.18 µm* CMOS technology is presented. The converter serves as a two-step ramp generator in a new HDRC® image sensor. The systematical gradient errors have been dramatically reduced by the utilization of the triple centroid switching scheme and the use of the 16-square matrix CSA distribution. This converter employs a newly proposed iterative thermometer decoder, which offers advantages in complexity, area and delay. The simulated maximum DNL and INL are within ±0.877LSB and ±0.0026LSB, respectively. A CSA area of *0.2 mm^2* has been achieved, which should be less than the minimum recently published 12-bit D/A converter area.

Literature

[1] M. Steyaert et al., **Static and Dynamic Performance Limitations for High Speed D/A Converters**. Kluwer Academic, 2004.

[2] A. Duinmaijer et al., **Matching properties of MOS transistors**. In IEEE Journal of Solid-State Circuits, vol. 24, no. 5, 1989.

[3] M. Hennessy et al., **A 12-bit 320MS/s current-steering CMOS D/A converter in 0.44 mm^2.** In IEEE Journal of Solid-State Circuits, vol. 39, no. 7, 2004.

[4] M. Borremans et al., **A 12-bit 200 MHz low glitch CMOS D/A converter.** In Proceedings of Custom Integrated Circuits Conference, 1998.

An Experimental Test Chip for TDC-based Digital Sensors

Andreas Brönner, Daniel Fuchs, Ulrich Brunsmann

Laboratory for Electronic Devices
University of Applied Sciences Aschaffenburg, Würzburger Str. 45, 63743 Aschaffenburg

{andreas.broenner, daniel.fuchs, ulrich.brunsmann}@h-ab.de

An experimental test chip for time-to-digital-converter-(TDC)-based digital sensors fabricated in 0.35 um CMOS-technology is presented. It contains voltage controlled differential delay lines, a 10-bit cyclic pulse shrinking TDC, a temperature sensitive differential ring oscillator and a 12-bit ripple counter. The components are deliberately designed to operate in the 0.5 MHz to 50 MHz range thus allowing easy access to all signals and wiring outside the chip. We present the basics of TDC-based digital sensors, the design and layout of the chip and an initial evaluation of performance data. A smart temperature sensor is proposed as an example of use.

1. Introduction

During the past decades, CMOS chip design has become dominant due to low power consumption, high level of integration, ease of design and low cost of fabrication. CMOS-MEMS-technology has been introduced in order to monolithically integrate sensors and electronics. CMOS technologies are typically optimized for the needs of digital circuitry, asking for smart sensors, which could easily communicate using a digital interface. This has spurred the development of integrated sensors that combine an analog sensor and a converter on the same chip. Commonly, most conventional smart sensors convert the measured signals into frequency, voltage or current signals. Whereas digital frequency read-out is straight forward, traditional analog-to-digital converters (ADCs) are utilized for converting voltage/current signals into subsequent digital output coding. As with emerging CMOS technologies the feature size dimensions are being reduced to increase the packing density and to reach higher levels of integration, the reduction in transistor gate-oxide thickness forces the system voltage to decrease, making analog design difficult to do [Roberts].

With the evolution of high-resolution CMOS-TDCs, conversion of the sensor signal to time or frequency and TDC or FDC (frequency-to-time conversion)-read-out has been proposed as an alternative to traditional ADC circuitry. This concept allows for integrating sensors and electronics using pure digital circuitry, what we call a digital sensor. The main advantages that have been addressed beyond overcoming analog impairments in nanometer-scale CMOS technologies are reduced active chip area and power consumption. Additionally, we would like to point out that digital sensors are promising candidates for technologies, where high precision analog circuitry is not available, such as thin film transistor technology including printed electronics.

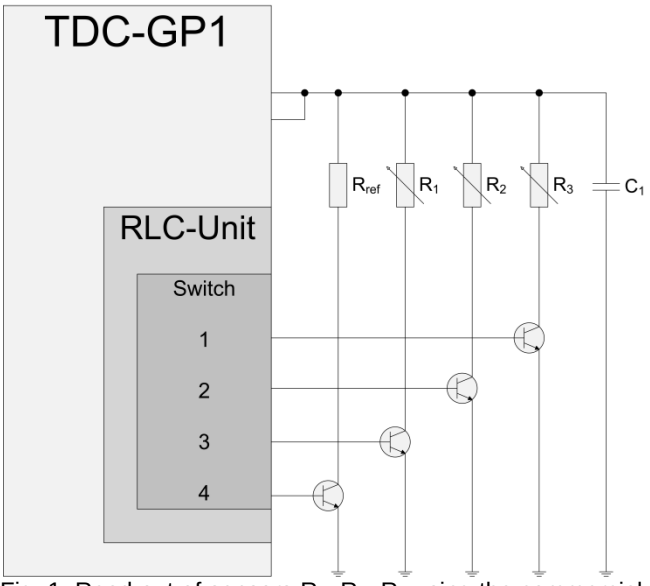


Fig. 1: Read-out of sensors R_1, R_2, R_3 using the commercial TDC-GP1 [Bruns]. First, the RC-circuit is charged to V_{DD} by the RLC-unit (switches 1-4 open). The capacitor C_1 is then discharged via a selected switch and the time measurement via the TDC is started. The measurement stops, if the voltage of C_1 is lower than a given threshold. Determination of the sensors resistance is by evaluating the ratio with respect to the reference resistance R_{ref}.

For resistive or capacitive sensors, a hybrid solution is commercially available. The measurement of the sensors resistance or capacitance is transformed to a TDC-based time measurement by measuring the discharging time of a RC-circuit outside of the TDC-chip (Fig. 1). The result is compared to that of a reference device. Patented methods that are used for reducing noise enable an accuracy that is comparable

to high-end 24-Bit A/D-converters or even better at high measurement rates [acam]. We have shown that semiconductor gas sensors can be operated at outmost sensitivity using this method [Bruns]. Monolithically integrated digital sensors that have been fabricated, e.g. [ChenP], within the last decade mostly use a delay element to convert the sensor signal into a time interval. Fig. 2 illustrates the operating principle.

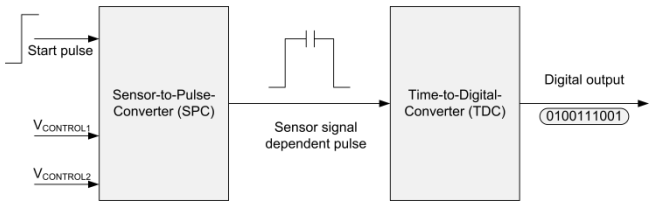


Fig. 2: Operating principle of TDC-based monolithically integrated digital sensors. The SPC converts the analog signal of a sensor into a pulse with a width proportional to the magnitude of the signal. The TDC converts this time interval to the digital output.

2. Related Work

Several CMOS-compatible methods for measurement of time intervals have been proposed that can be broadly classified into two groups: those whose time resolution is based on the minimum gate delay available in the process technology, and those achieving sub-gate delay resolution, including time stretching followed by counter methods [Chao], double conversion (time-to-amplitude followed by standard analog to digital A/D conversion) [Becker], the Vernier method [Rama, Dudek], successive approximation interpolation method [Mänti], sample and hold based TDC [Napo] and the cyclic pulse shrinking TDC [ChenC]. TDC's based on the minimum gate delay are classically utilizing the tapped delay line, sub-gate delay resolution is based on the evaluation of delay differences. Typically TDCs are deployed for experiments in nuclear physics [Junn], laser range-finding [Niss], and all-digital frequency synthesis [Stasz]. For a review of TDC architectures, the theoretical basis and an overview of applications see [Henzler]. By converting a sensor analogue signal into a pulse with a width proportional to the magnitude of the signal, every TDC can also be used as an alternative to the traditional ADC. Digital sensors, however, require that the main building blocks are based on pure digital concepts. The fabrication up to now concentrated on temperature sensors using CMOS technology.

For traditional temperature sensors with on-chip ADC more than ten output bits is usually required in order to obtain the necessary resolution [Bakker]. The monolithically temperature sensor usually adopts the parasitic substrate or lateral bipolar transistor for temperature sensing using the bandgap reference. In order to reduce measurement errors, additional calibration circuits at the expense of chip area and power consumption proved to be necessary. As an alternative, a time-domain smart temperature sensor was introduced to address these issues [ChenP]. First of all, the test temperature was converted into a pulse with a width proportional to the test temperature. This temperature-to-pulse-generator was accomplished by XORing the outputs of two delay lines. A thermal compensation circuit was used in only one of these. Thus, the pulse-width generated by the XOR-gate was sensitive to temperature. It was fed into a TDC for output coding.

Currently, besides this TDC-based concept there are further approaches in digital sensor designs. Recently starved ring oscillators are implemented as temperature sensitive elements [Kim, Park]. The bottleneck of this circuit is the accurate biasing of reference current generator circuit, a truly temperature independent reference can hardly be obtained [Dalal]. Effects of static supply voltage variations on the temperature measurement are reduced by using a pair of differently-biased, current-starved ring oscillators [Spencer]. Today low power applications implement ring oscillators working in subthreshold region [Park, Woo]. Lately a new method of delay generation was reported with MOSFETS working in linear region [Law], suitable for low power implementation.

In this paper an experimental test chip for time-to-digital converter-(TDC)-based digital sensors fabricated in 0.35 um CMOS technology is presented. The chip was designed for advanced laboratory courses within our research projects on smart sensors. Therefore, the components were deliberately designed to operate in the 0.5 MHz to 50 MHz range thus allowing easy access to all signals and wiring outside the chip. Furthermore, the elementary cells have not been compensated with respect to variations of temperature and supply voltage in order to clearly demonstrate the resulting effects. The chip essentially contains two voltage controlled differential delay lines, a 10-bit cyclic pulse shrinking TDC and a temperature sensitive differential ring oscillator. The delay lines can be used to test signal-to-pulse converters within the complete operation range from weak to strong inversion as well as to test ring oscillator- and TDC-structures by external wiring. Unlike previous realizations we use a voltage controlled linearization circuitry which emphasizes sensitivity to temperature or allows for compensation of temperature sensitivity if operated in the region of weak or strong inversion, respectively. The TDC can be used for time interval

measurements, e.g. for LIDAR-experiments, and as a temperature sensor. The ring oscillator which has been integrated as a test structure for frequency-to-time-converter-based sensors will not be considered here. As an example of use we present a smart temperature sensor which in contrast to previous TDC-based approaches does not use a delay element but the TDC itself as temperature sensitive component.

The paper is organized as follows: in Section 3 we use Spice level 1 equations to describe the operation of the delay elements and the TDC, in Section 4 we present the design and the layout of our test chip, in Section 5 we show initial performance data and in Section 6 we present the smart temperature sensor and its performance, before we summarize the main conclusions and discuss open issues in Section 7.

3. Basics of TDC-based digital sensors

We use level 1 SPICE equations and the modelling given in [ChenP, Tisa] in order to derive models for the design of our current starved delay line, of the cyclic pulse shrinking TDC, and of digital sensors. Channel length modulation is neglected. The underlying MOSFET models are described in textbooks, e.g. [Baker, Weste].

3.1. Propagation delay and digital sensors

Fig. 3 shows the basic delay cell which we use in the delay line of the TDC. With respect to the use in laboratory courses, we use long channel FETs (length L = 3.5 µm) for reasonable agreement of measured results with hand calculations. Additionally to the intrinsic delay of the inverters we add a NMOS-capacitor to the capacitive load in order to further increase the propagation delay.

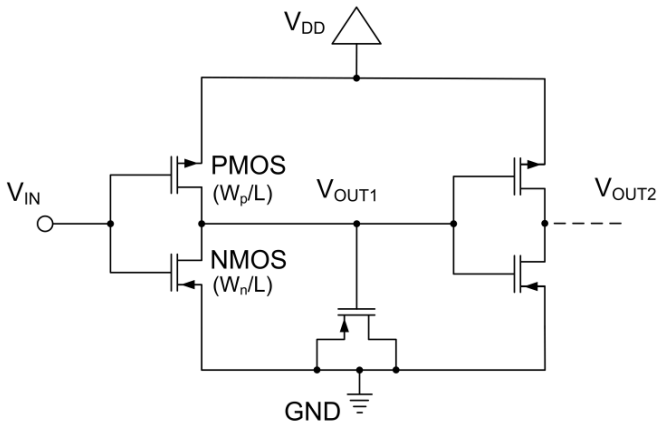


Fig. 3: Basic delay cell. W and L denote the width and the length of the gate, respectively.

Fig. 4 shows a simple RC-equivalent circuit, where the transistors have been replaced by switches. Cs denotes a capacitance additive to the intrinsic MOSFET capacitance. Within a digital sensor Cs might be replaced by a capacitive-type sensor element, within our test chip it denotes the NMOS-capacitor.

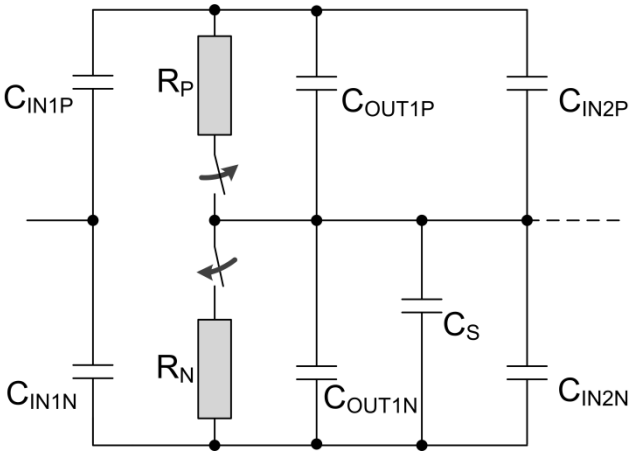


Fig. 4: Equivalent circuit of the basic delay cell. The arrows indicate switching the output from high to low (input high).

We summarize the load capacitance of the switches to C_{load}. The equations which describe the propagation delay for a switching point at 50% of the amplitude, t_{pLH} (transition from low to high, and t_{pHL} (transition from high to low),

$$t_{pHL} = 0{,}7 \cdot R_N \cdot C_{load} \tag{3.1}$$

$$t_{pLH} = 0{,}7 \cdot R_P \cdot C_{load} \tag{3.2}$$

illustrate that a propagation-delay-based sensor can be realized using digital circuitry incorporating a capacitive-type sensor in C_{load} or a resistive-type sensor in R_P or R_N. The latter means, that the current which charges or discharges the capacitive load is sensitive to a physical property.

If silicon is used as base material, the mobility of the charge carriers is a function of temperature [Sze] and the current through the switches varies accordingly. Operating the MOSFETS above the threshold voltage V_T, which also varies with temperature, and above cryogenic temperatures, well known empirical formula [Baker, Weste] apply for silicon:

$$\mu(T) = \mu(300K)\left(\frac{T}{300K}\right)^{-a} \qquad a = 1{,}5 \ldots 2{,}5 \tag{3.3}$$

$$\frac{\Delta V_T}{\Delta T} = -1\frac{mV}{°C} \ldots -3\frac{mV}{°C} \tag{3.4}$$

These effects have been used to build a digital sensor and as they inversely modify the switching current, also for thermal compensation, e.g. by [ChenP].

If the MOSFET is operated in the subthreshold region temperature enabled carrier diffusion dominates the carrier transport in the channel, thus a strong increase of the drain current is observed with increasing temperature. This effect recently has been used to build a digital temperature sensor based on a PTAT (proportional to absolute temperature) element by [Ueno].

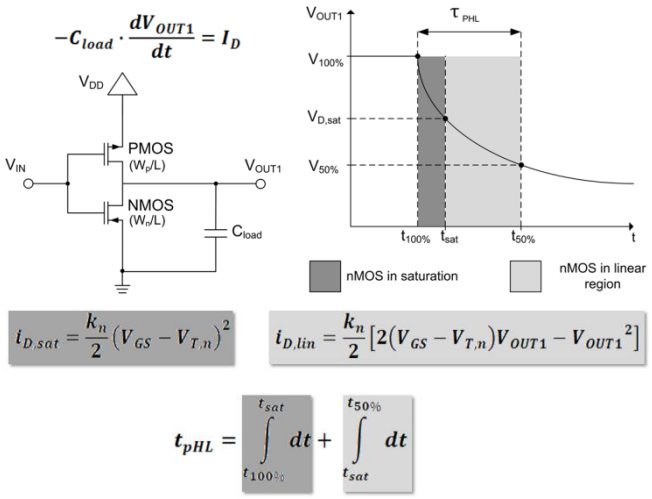


Fig. 5: Modelling the propagation delay t_{pHL}: basic differential equation (upper left side) describing the discharging and illustration of the discharging curve (right side); equivalent circuit (left side) and current-voltage-characteristics. A switching point at 50% of the amplitude is assumed.

In order to get useful equations for a hand-calculation-based design, the basic differential equation describing the charging/discharging process is integrated. Thereby it has to be considered that the drain-source-voltage surpasses the saturation voltage $V_{DS,sat}$

$$V_{DS,sat} = V_{GS} - V_T \qquad (3.5)$$

during charging/discharging, where V_{GS} denotes the gate-source-voltage and V_T the threshold voltage for either the NMOS (V_{Tn}) or the PMOS (V_{Tp}). Therefore, the integral over time is split into a sum according to the drain-source current in the saturation ($I_{D,sat}$) and in the triode region ($I_{D,lin}$). Fig. 5 illustrates the derivation of the corresponding Equation (3.6) for the transition from high to low.

$$t_{pHL} = -C_{load}\left[\int_{V_{100\%}}^{V_{D,sat}} \frac{1}{I_{D,sat}} dV_{OUT1} + \int_{V_{D,sat}}^{V_{50\%}} \frac{1}{I_{D,lin}} dV_{OUT1}\right] \qquad (3.6)$$

Setting the high-voltage to V_{DD}, the low-voltage to V_{SS}, and the gate-source voltage to V_{DD} we get equation (3.7) for t_{pHL}, t_{pLH} is obtained analogue.

$$t_{pHL} = \frac{C_{load}}{\mu_n C_{ox}(V_{DD} - V_{Tn})} \frac{L}{W}\left[\frac{2V_{Tn}}{V_{DD} - V_{Tn}} + ln\left(\frac{4(V_{DD} - V_{Tn})}{V_{DD}} - 1\right)\right] \qquad (3.7)$$

$$t_{pLH} = \frac{C_{load}}{\mu_p C_{ox}(V_{DD} - |V_{Tp}|)} \frac{L}{W}\left[\frac{2|V_{Tp}|}{V_{DD} - |V_{Tp}|} + ln\left(\frac{4(V_{DD} - |V_{Tp}|)}{V_{DD}} - 1\right)\right]$$

In these equations C_{OX} denotes the oxide capacitance per area and μ_n, μ_p, denote the electron and hole mobility, respectively. The equations provide an appropriate description for the propagation delay of our basic delay cell and they show up the influence of μ, V_T, V_{DD}, C_{load}, and W/L.

3.2. Voltage controlled delay line

For the voltage controlled delay lines we use current starved inverters as known from voltage controlled oscillators (Fig. 6). The circuitry for linear current control which we use also is well known, see e.g. [Baker]. In order to obtain the propagation delay we now use the constant current $I_{control}$ = $I_{inverter}$ for integration of the basic differential equation (Fig.5). For discharging we get

$$t_{pHL} = \int_{t_{100\%}}^{t_{50\%}} dt = -C_{load} \int_{V_{DD}}^{\frac{1}{2}V_{DD}} \frac{1}{I_{CONT}} dV_{OUT} = \frac{C_{load} V_{DD}}{2 I_{CONT}} \qquad (3.8)$$

showing that the propagation delay will increase with the magnitude of C_{load} and decrease with increasing $I_{control}$.

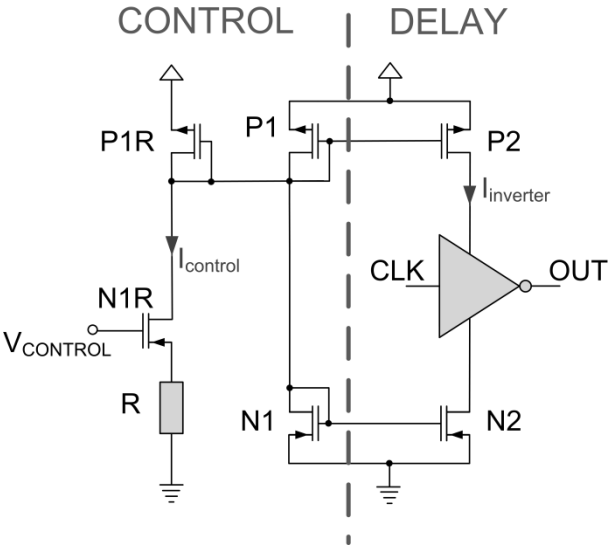


Fig. 6: Voltage controlled delay element. A linearization circuitry, see e.g. [Baker], is used to generate a voltage controlled current $I_{control}$ which is mirrored into the inverters of the basic delay cell. N1R is an NMOS with W/L = 70/0.7 P1R is a PMOS with W/L=14/.07 and R is a 12 kΩ poly-resistor.

The linearization circuitry is designed to operate as follows. If $V_{control}$ is greater than the threshold voltage V_{Tn} of the NMOS N1R, which has been measured to be slightly above 0.5 V, $I_{control}$ causes a voltage drop at R and at the drain of N1R. Neglecting in a first step the temperature coefficient of R, by means of Spice-simulation we ensured that this current feedback causes a linear increase of $I_{control}$ with temperature and with $V_{control}$ in the range of moderate and strong inversion. Only above $V_{control}$ = 3.0V $I_{control}$ levels off. As has been shown by Kim [Kim] the temperature coefficient of R can be used to tailor the temperature sensitivity of the delay element. Operating in this range, differential delay elements can be used in order to widely compensate for the temperature effect, which will be necessary for all smart sensors other than temperature sensors.

If $V_{control}$ is sufficiently below V_{Tn} of the NMOS N1R, the NMOS operates in weak inversion. Due to the low drain current, the IR-drop is negligible. As in this subthreshold region the drain current mainly is driven by diffusion, $I_{control}$ strongly depends on temperature and on $V_{control}$. The delay element can be operated with outmost sensitivity to temperature in this region. According to Eq. 3.8, however, the propagation delay is big and the maximum rate of measurement is accordingly low.

3.3. Cyclic pulse shrinking TDC

The cyclic pulse shrinking TDC carries the measurement pulse along a delay line in a loop structure that contains at least one asymmetric stage. Fig. 7 illustrates the operating principle commonly used.

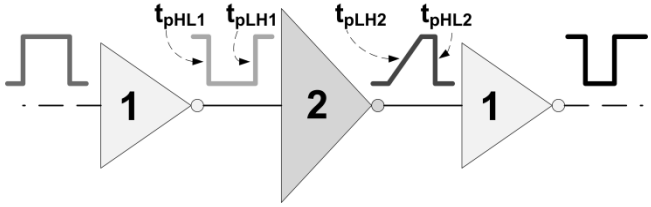


Fig. 7: Operating principle of the cyclic pulse shrinking TDC. Assume for simplicity that the inverter type 1 are symmetric (t_{pHL1} = t_{pLH1}) having its switching points at 50%, indicated by the arrows. Inserting the asymmetric gate type 2 (t_{pHL2} < t_{pLH2}) causes the reduction of the pulse width. The shrinked output pulse is type 1 inverted (not shown) and fed back to the input until it vanishes completely.

Generally, the pulse width P is reduced by

$$\Delta P = -\left(t_{pLH1} - t_{pHL1}\right) + \left(t_{pLH2} - t_{pHL2}\right), \quad (3.9)$$

if ΔP < 0, [Henzler]. ΔP is the time interval corresponding to the least significant bit, t_{LSB}. Applying Equations 3.7 and simplifying for equal threshold voltages V_T for the NMOS and PMOS we arrive at

$$\Delta P = \alpha L^2\left[W_{n1} + W_{p1} - W_{n2} - W_{p2}\right] \left[\frac{1}{\mu_p}\left(\frac{1}{W_{p1}} - \frac{1}{W_{p2}}\right) - \frac{1}{\mu_n}\left(\frac{1}{W_{n1}} - \frac{1}{W_{n2}}\right)\right] \quad (3.10)$$

for transistor gates of identical length L; α then only depends on V_{DD} and V_T:

$$\alpha = \frac{2V_T}{(V_{DD} - V_T)^2} + \frac{1}{(V_{DD} - V_T)} \cdot ln\left(\frac{1{,}5V_{DD} - 2V_T}{0{,}5V_{DD}}\right) \quad (3.11)$$

Obviously, pulse shrinking occurs if the two brackets of Eq. 3.10 have different sign. Note, that the magnitude of the reduction, i.e. t_{LSB}, ideally is determined by appropriate differences in Eq. 3.9, not by the propagation delay itself. Thus t_{LSB} seems to be independent of the fabrication technology used. However, as in the presence of noise small differences are hard to realize, we would also like to point out that t_{LSB} scales with L^2 (Eq. 3.10).

Usually, the asymmetry is incorporated in a coupling stage necessary for resetting the TDC and for input coupling. We use a double NOR-gate according to Tisa et al. [Tisa]; Fig. 8 shows the structure of our TDC.

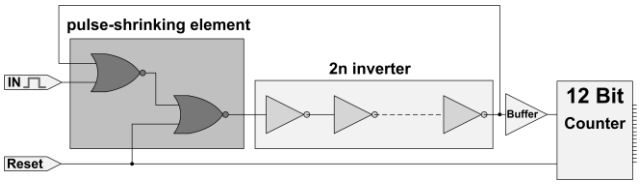


Fig. 8: Structure of our TDC according to [Tisa]. The counter is implemented but must be wired outside the chip.

4. Design and layout

As some of the process data necessary for the design are not explicitly given in the documentation of the process which we have selected for fabrication, we have fabricated a precursor chip containing NMOS and PMOS transistors of different channel length and width, spanning the range of data which we use in the test chip described here. Using our wafer prober equipment we have measured the subthreshold current as a function of gate-source-voltage down to 10^{-11}A, the C(V)-characteristics from accumulation to strong inversion and we have extracted level 1 SPICE parameters including KP and VTO. These data together with those of the process documentation have been used for hand calculations.

4.1. TDC design

The first column of Table 1 shows the design results for the TDC which we obtain using the equations given in section 3.

Table 1: Design-, simulation-, and measurement-data of the TDC.

TDC Performance at 25 °C; process data: typical mean			
	Design	Simulation	Measurement
pulse shrinking / ps	300	350	120
propagation delay inverter / ns	2.8	4.3	2.7
propagation delay delay line / ns	300	460	290
300 ns – interval measurement / ms	0.3	0.4	0.8

We initiated the design process with TDC-W/L-data according to Tisa et al. [Tisa], followed by tuning but maintaining a slightly asymmetric inverter, i.e. the propagation delay time of the low-high transition t_{pLH} is slightly higher than that of the high-low t_{pHL} transition. The target maximum time interval for the TDC measurement was set to about 300 ns in order to allow for calibration using a 20 MHz quartz oscillator. This in turn sets the lower bound for the total propagation delay of the TDC delay line to the same value because the input pulse must finish before the output of the delay line is fed back to the input. Otherwise the TDC takes up its stable high state. Due to the constraints of chip area we use 108 inverters in the delay line. This leads to a propagation delay of 2.8 ns for a single inverter driving the capacitive load as discussed above.

Pulse shrinking is essential for the operation of the cyclic TDC und thus is the most critical data of our design. We use Eq. 3.10 to get an estimation for the usable width of the gates. For hand calculations the maximum capacity of the NMOS-capacitor instead of the complete C(V) characteristics has been included in C_{load} and the input capacity of the asymmetric NOR-gate has been taken into account. With Eq. 3.10, we obtain a pulse shrink of 300 ps at 25 °C. By SPICE-simulation using the BSIM3v3 level 49 model and typical mean process data as well as worst case speed and by a sensitivity analysis with respect to the ratios of the widths of the gates we finally ensured the pulse shrinking in either case.

The magnitude of pulse shrinking determines t_{LSB}, the maximum time resolution of the TDC measurement. As the pulse shrinking is due to propagation delay differences the cyclic pulse shrinking TDC operates with sub-propagation-delay time resolution as shown in Table 1. As the magnitude of pulse shrinking is designed to be constant, this type of TDC at least theoretically does not need linearization measures. The magnitude of pulse shrinking determines the number of runs through the delay line until the input pulse vanishes at the output. This in turn determines the maximum rate of measurement which for our test chip is about 1 kHz for a 300 ns input pulse.

Table 1 shows that the data of our hand calculations coincide roughly within a factor of 2 with the simulated results and with the measured data.

4.2. Differential voltage controlled delay line

For the differential voltage controlled delay lines we combine two current starved delay cells (1,2) as described in section 3.2 to form a differential delay cell. We have arranged these cells in the sequence (1,2)-(2,1)-(1,2) in order to reduce the influence of process variations on the propagation delay at any stage of the delay lines. Each of the delay lines (1,2) consists of 78 current starved inverters and provides analog output pads after 37 and 78 inverters. The slight difference of t_{pHL} and t_{pLH} of the single inverter stages is magnified accordingly. The delay lines can be used to test ring oscillator- and TDC-structures by external wiring. The analog-outputs allow for external selection of the switching point.

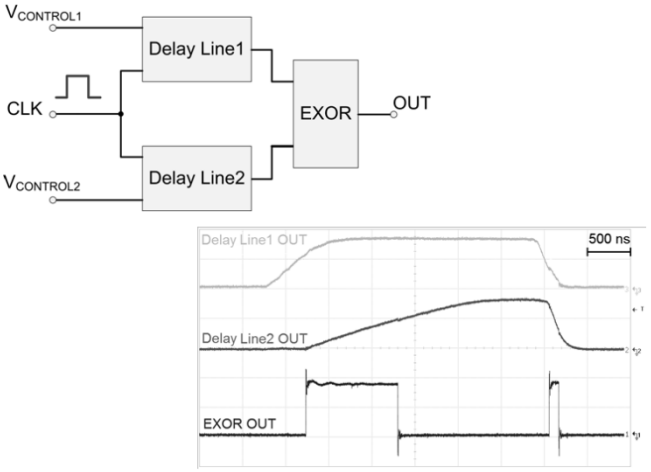


Fig. 9: Signal-to-pulse converter by XORing corresponding outputs of two voltage controlled delay lines. The lower right side shows measured data at the rising and the falling edges of the delayed clock input pulses. EXOR OUT is high iff the amplitude of one of the input pulses is above the switching point of the EXOR.

A signal-to-pulse converter is realized by XORing corresponding outputs (Fig. 9). Using appropriate control voltages the operation range can be varied from weak to strong inversion. Subthreshold operation causes rise times that require an input time interval of

at least 1 ms in order to achieve the switching point of the EXOR. As with the TDC these data are in accordance with simulation and measurement.

4.3. Design flow and layout

The test chip was designed using the Tanner Tools Pro v.14 Design Suite and it was fabricated via Europractice in a 0.35 um CMOS-technology**.** Fig. 10 illustrates the Tanner design flow, Fig. 11 shows a microphotograph of the die. For the main components namely the TDC, the delay lines and the ring oscillator, we made a full custom design whereas for the counter, the EXOR and some buffers we used standard cells of the process.

Fig. 10: Tanner Tools Pro include schematic entry (S-Edit) with netlist extraction, SPICE-Simulation (T-Spice) with graphical post processing (W-Edit), layout editing (L-Edit) with design rule check, extraction of parasitic, layout vs. Schematics and tape-out.

The chip area is 3 mm^2. Obviously, we obtain the high propagation delays at the expense of chip area. Consequently, the performance of the devices will suffer from process variations across the die. This, however, is of less importance for the intended applications of the test chip.

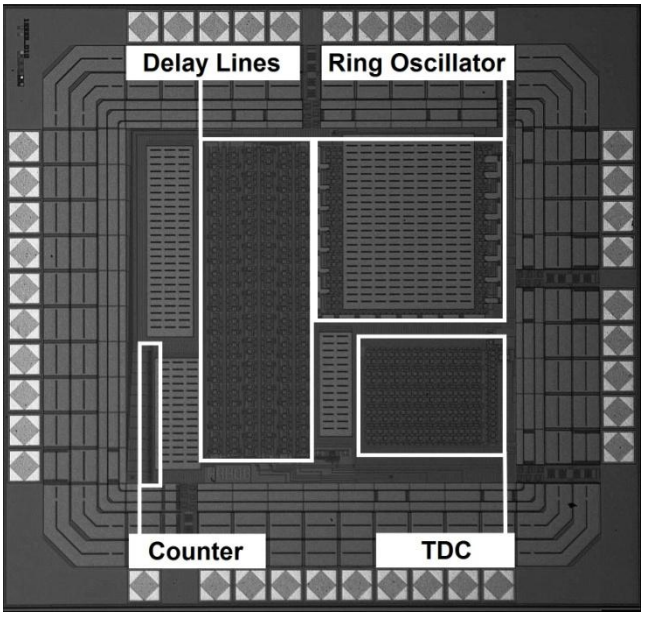


Fig. 11: Microphotograph of the test chip.

5. Test data

5.1. Experimental setup

Characterization of TDCs is a challenging task. For the initial evaluation presented here we used a signal generator (Agilent 81110A) for input pulse generation, a waveform generator (Agilent 33210A) for synchronisation, a frequency counter (Agilent 53131A) for external pulse counting, and an oscilloscope (Agilent DSO 90254a) for measurement of the EXOR output pulse width. As specified by the manufacturer the rise time of the input pulses is about 2 ns with a jitter of 15ps. In order to control the differential delay lines, a two channel voltage supply (Agilent E3631A) was used. The test chips were placed inside a programmable temperature chamber (Heraeus HT 4002). Thermocouples mounted at the upper and lower surface of the test chip monitored the temperature with a resolution of 0.1°C. The measurement process was almost automated via LabView.

For evaluating the components, a configurable printed evaluation board was designed to operate the TDC and both delay lines as single systems. A second evaluation board was developed for the temperature sensor (Fig. 12). An Atmel Atmega 128PV microcontroller (µC) was utilized as control device, the µC was used to activate the sensor system as well as to handle the communication within the test chip and to the PC.

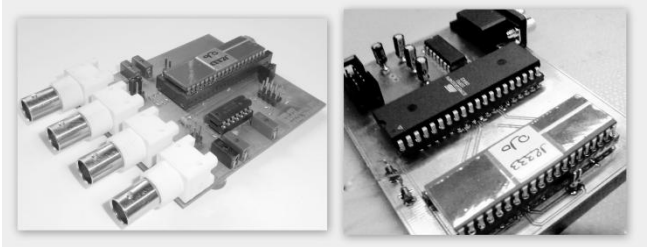


Fig. 12: (a) Evaluation board for single components; (b) Evaluation board for the temperature sensor system showing a test chip in the foreground.

5.2. Differential delay lines

In order to evaluate the performance of the differential delay lines, a square wave generated by the signal generator was applied at CLK (Fig. 9). It will be delayed by each delay line according to the external control voltage, temperature and supply voltage. Consequently the EXOR output pulse width depends on these data. The relationship between EXOR output pulse width of the rising edge and control voltage is shown in Fig. 13.

Consider a curve of constant temperature. While operating the delay line 1 at V_{con1} = 0.4V ($V_{Tn} \approx 0.5V$) a small increase of control voltage 2 will cause a

strong rising output pulse width. We assign this effect to the exponential increase of the drain source current with gate source voltage in the subthreshold region. Fig. 13 shows a smooth rise of EXOR output pulse width if the current controlling NMOS of delay line 2 is operated above V_{Tn}.

Likewise the temperature response of the output pulse width can be learnt from Fig. 13. Consider $V_{con2} = V_{con1} + 500$ mV. This means that the current controlling transistors N1R1 and N1R2 (see Fig. 6) of delay line 1 and delay line 2 operate below and above V_{Tn}, respectively. The resulting output pulse width is getting smaller with increasing temperature, due to the effect that the current through N1R1, operating below V_T, grows faster with temperature than that of N1R2 (see section 3.2).

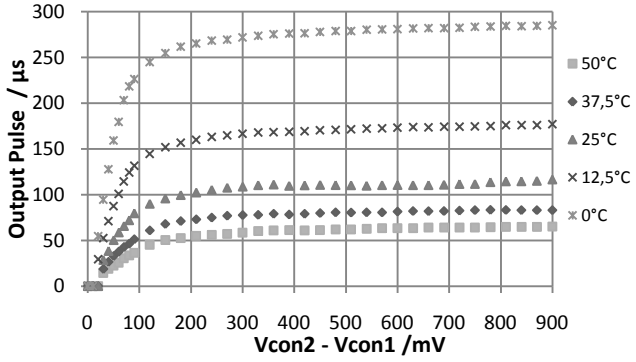


Fig. 13: EXOR output pulse width for the rising edge of the input signal as function of ΔV_{con}; V_{con1}=400 mV.

With increasing control voltages V_{con1}, V_{con2} the temperature sensitivity almost disappears as expected (Figs. 14, 15). Note that the sensitivity of the complete differential delay line for subthreshold operation is of the order of 5 µs/°C. This is obtained at the expense of the rate of measurement. Operating one of the delay lines at V_{con} = 0.4V requires a clock pulse of at least 1 ms in order to achieve the switching point of the EXOR.

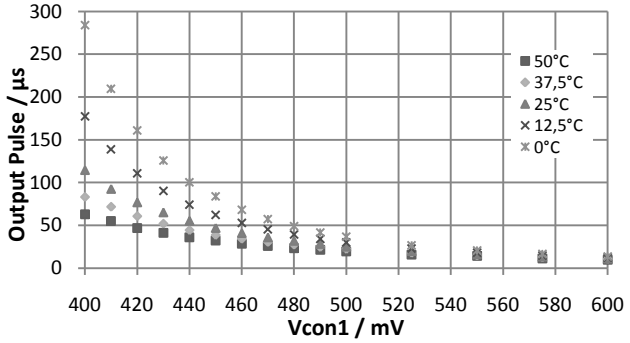


Fig. 14: EXOR pulse width as function of V_{con1}; ΔV_{con} = 900 mV.

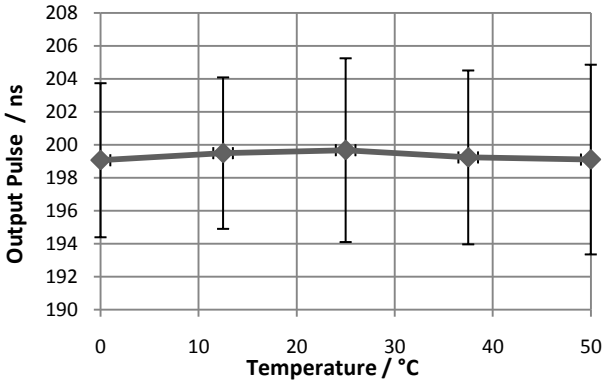


Fig. 15: EXOR pulse width at V_{con1}=2,4 V; V_{con2}=3,3 V.

Fig. 16 demonstrates that the EXOR output of the proposed differential delay line decreases with decreasing supply voltage as can be deduced from Eq. 3.8.

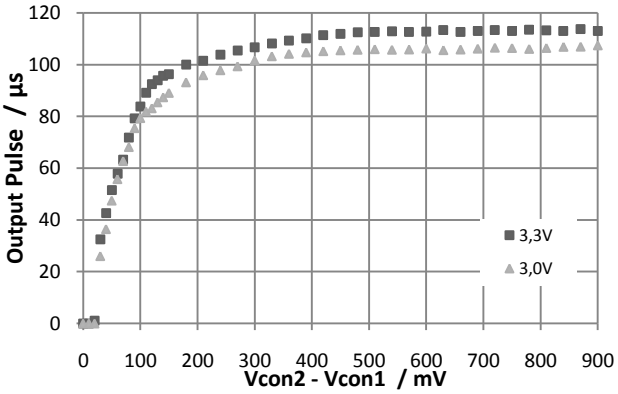


Fig. 16: EXOR pulse width for V_{DD} = 3.3 V and V_{DD} = 3.0 V as a function of ΔV_{con}; V_{con1} = 400 mV; temperature: 25°C.

5.3. Cyclic TDC

In order to evaluate the performance of the cyclic TDC, the digital output code was measured as a function of input pulse width for 0 °C to 37.5 °C with 12.5 °C increment. Test signals with different pulse widths ranging from 15 ns to 300 ns were generated by the Agilent 81110A pulse generator. The Agilent 53132A counter was used to collect 50 output codes of the TDC every 5 ns at a rate of 150 Hz. The operations of the measurement equipment and the TDC test board were triggered by the Agilent 33220A function generator. Figs. 17 and 18 show experimental results (mean of the 50 output codes) over the 37.5 °C temperature range.

The useful input time interval (270 ns to 320 ns) as well as sensitivity increases with increasing temperature and decreasing supply voltage. From the output code for a 280 ns input pulse we calculate a

resolution of 120 ps at 25°C and V_{DD} = 3.3 V. However, the standard deviation of the output code is in the order of 1% and we observe uncorrelated fluctuations in the order of 1 ns to 5 ns at different operating temperatures which currently limit the observed differential and integral linearity, DNL and INL, and the effective number of bits, ENOB, of the device. It is reasonable to assume that measurement errors, corresponding to sub-nanosecond resolution, may be mostly induced by the inherent measurement errors of our initial test setup requiring further refinement. Such effects have also been discussed by [ChenC].

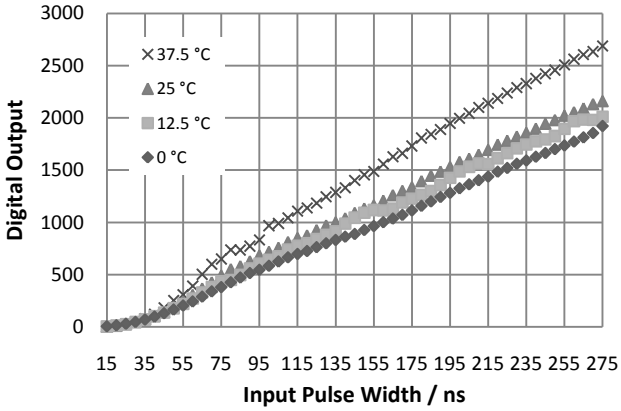


Fig. 17: TDC-output as a function of input pulse width for various temperatures

Fig. 18 shows that the digital output code also differs systematically from the regression of 3.3V. Till an input pulse width of 75 ns the output code is below the regression for all temperatures and then nearly equal up to 215 ns. This effects the INL errors corresponding to the area of operation where the TDC is used. The simulation shows the effects of nonlinearity within that range of short input pulses as well, indicating that we have introduced a nonlinearity with our design that requires further investigation.

Note that the sensitivity of the output code to temperature and supply voltage is contrary to that of the delay line. Hence, an appropriate combination can be used for compensation of both effects. The maximum measurement time of our TDC is about 0.8 ms, depending on the width of the input pulse. The mean current consumption was measured to be 2 µA or less.

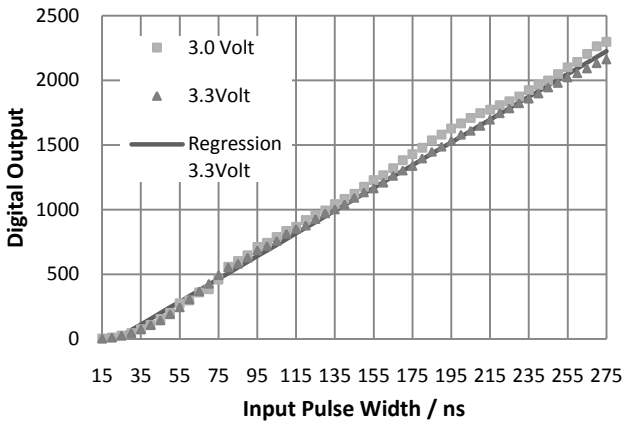


Fig. 18: TDC-output as a function of input pulse width for two supply voltages. The figure also shows the systematic deviation from regression below an input pulse width of 75 ns.

6. Smart temperature sensor

A smart temperature sensor is proposed, using the sensitivity of the TDC output code to temperature directly. Former TDC based digital temperature sensors deploy delay elements as temperature sensitive component [ChenP]. We in contrast operate the differential delay line according to Fig. 15 as pulse generator which at least is less sensitive to temperature then the TDC. Fig. 19 shows the block diagram of the sensor system. A µC (ATMEGA 128PV) generates a pulse between 1ms and 7s length, tides over the measurement time of TDC to the PC and assesses the measurement rate. The µC-generated pulse runs through the delay line, where the EXOR output provides a nearly constant pulse width of 200ns. The TDC output pulses are counted via the on-chip 12 bit ripple counter. After the time-to-digital conversion the counter is readout by the µC. A serial PC to µC communication was realized over an RS232 connector.

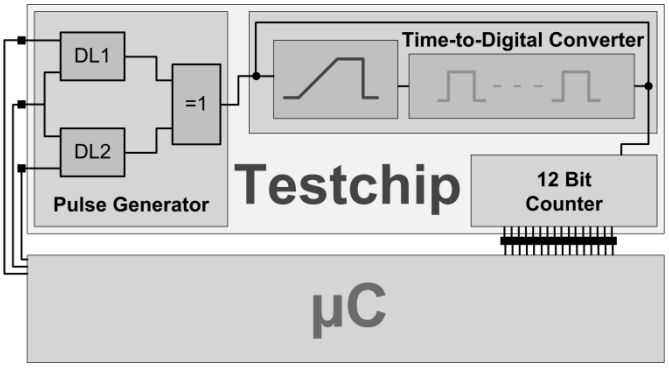


Fig. 19: Block diagram of the smart sensor system containing µC, delay lines, EXOR, time-to-digital-converter and 12 bit counter.

To evaluate the performance of the sensor, the measurement was done every five degrees from 0°C to 50°C at a rate of 50 samples/s (Fig. 20). Under single 3.3V power supply, the power consumption is measured to be 39.6mW containing 33mW from the µC. The proposed temperature sensor has a monotonically increasing output code and an effective resolution measured to within 0.1°C (lower temperatures) and 1.3°C (higher temperatures). State-of-the-art digital temperature sensors feature resolutions about 0.1°C [Chen10].

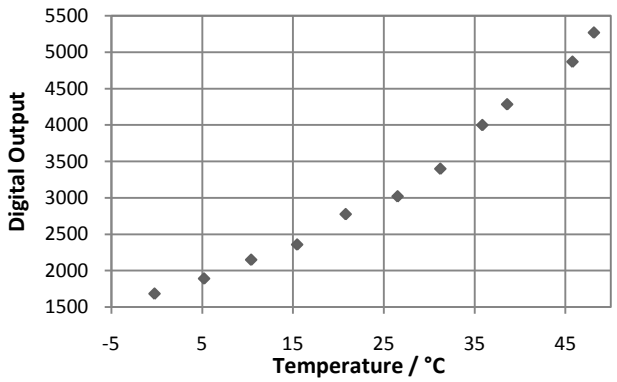


Fig. 20: Sensor output as a function of temperature.

7. Summary and conclusions

We have presented an experimental test chip for time-to-digital converter-(TDC)-based digital sensors that was designed for advanced laboratory courses within our research projects on smart sensors. Using long channel devices we achieve appropriate consistence of modelling, simulation and measurement. A low frequency design allows for easy access to all signals and wiring outside the chip.

In this way voltage controlled differential delay lines can be configured externally either to voltage controlled ring oscillators or to TDCs with voltage controlled resolution. The measurements presented show that in combination with the on-chip TDC these delay lines may further be used as signal-to-pulse generator either for compensation of effects due to temperature and supply voltage or as sensing element. We demonstrated the use of a voltage controlled linearization circuitry to allow for operation in weak, prolonged moderate and strong inversion.

A single stage cyclic pulse shrinking TDC was realized. It was designed for a maximum time resolution of 300 ps, spanning the input range from 20 ns to 300 ns at the expense of chip area and signal to noise ratio. The initial measured data, 120 ps, 15 ns and 300 ns, respectively, show excessive noise and nonlinearity with respect to ideal performance. This at least partly may be due to the experimental setup used and it is of less importance for the intended applications of the test chip. The performance demonstrated already allows for studies of TDC-based sensor concepts including LIDAR experiments of low resolution.

As an example of use we have proposed a smart temperature sensor which in contrast to known approaches operates the differential delay line as a pulse generator which at least is less sensitive to temperature then the TDC. It has a monotonically increasing output code and an effective resolution which at low temperatures is close to the state-of-the art of digital temperature sensors. Further research will concentrate on optimized digital sensors and TDCs.

Acknowledgements

The authors would like to acknowledge the support of the *Bayerisches Staatsministerium für Wissenschaft, Forschung und Kunst* in the context of the *Forschungsschwerpunkt Intelligente Sensorik* at the University of Applied Sciences, Aschaffenburg.

References

[acam] Acam messelectronic gmbh, www.acam.de, last access: 19.07.2010.

[Baker] R. J. Baker. CMOS Circuit Design, Layout and Simulation, Revised Second Edition. Wiley-IEEE Press, 2007.

[Bakker] A. Bakker, J. H. Huijsing. Micropower CMOS Temperature Sensor with Digital Output. IEEE J. Solid-State Circuits, 31(7), 933-937, 1996.

[Becker] W. Becker, A. Bergmann, E. Haustein, Z. Petrasek, P. Schwille, C. Biskup, T. Anhut, I. Riemann, K. Koenig. Fluorescence lifetime images and correlation spectra obtained by multi-dimensional TCSPC. In Proc. SPIE, 1-8, 2005.

[Bruns] U. Brunsmann, Th. Tille. High resolution readout of metal oxide gas sensors using time-to-digital conversion. Electron. Lett., 42(20), 1-2, 2006.

[Chao] A.-S. Chao, S.-J. Chang. A Jitter Characterizing BIST with Pulse-Amplifying Technique. In ATS'09, 379-384, 2009.

[ChenC] C.-C. Chen, P. Chen, C.-S. Hwang, W. Chang. A Precise Cyclic CMOS Time-to-Digital Converter With Low Thermal Sensitivity. IEEE Trans. Nucl. Sci., 52(4),

834-838, 2005.

[ChenP] P. Chen, C.-C. Chen, C.-C. Tsai, W.-F. Lu. A time-to-digital-converter-based cmos smart temperature sensor. IEEE J. Solid-State Circuits, 40(8), 1642-1648, 2005.

[Chen10] P. Chen, C.-C. Chen, Y.-H. Peng, K.-M. Wang, Y.-S. Wang. A Time-Domain SAR Smart Temperature Sensor With Curvature Compensation and a 3σ Inaccuracy of 0.4°C ~ +0.6°C Over a 0°C to 90°C Range. IEEE J. Solid-State Circuits, 45(3), 600-609, 2010.

[Dalal] H. F. Dalal. Implementation of On-chip Thermal Sensor using Off-Leakage Current of a Transistor. Master thesis university of Minnesota, 2010.

[Dudek] P. Dudek, S. Szczepanski, J. V. Hatfield. A High-Resolution CMOS Time-to-Digital Converter Utilizing a Vernier Delay Line. IEEE J. Solid-State Circuits, 35(2), 240-247, 2000

[Henzler] S. Henzler. Time-to-Digital Converters, Springer Series in Advanced Microelectronis, 2010.

[Junn] S. S. Junnarkar, P. O'Connor, P. Vaska, R. Fontaine. FPGA-Based Self-Calibrating Time-to-Digital Converter for Time-of-Flight Experiments. IEEE Trans. Nucl. Sci., 56(4), 2374-2379, 2009.

[Kim] K. Kim, H. Lee, S. Jung, C. Kim. A 366kS/s 400uW 0.0013mm² Frequency-to-Digital Converter Based CMOS Temperature Sensor Utilizing Multiphase Clock. In IEEE CICC, 203-206, 2009.

[Law] M. K. Law, A. Bermak. A 405-nW CMOS Temperature Sensor Based on Linear MOS Operation. IEEE Trans. Circuits Syst. II, 56(12), 891-895, 2009.

[Mänti] A. Mäntyniemi, T. Rahkonen. A CMOS Time-to-Digital Converter (TDC) Based On a Cyclic Time Domain Successive Approximation Interpolation Method. IEEE J. Solid-State Circuits, 44(11), 3067-3078, 2009.

[Napo] P. Napolitano, F. Alimenti, P. Carbone. A Novel Sample-and-Hold-Based Time-to-Digital Converter Architecture. IEEE Trans. Instrum. Meas., 59(5), 1019-1026, 2010.

[Niss] J. Nissinen, P. Palojärvi, J. Kostamovaara. A CMOS Receiver for a Pulsed Time-of-Flight Laser Rangefinder. Proc. 29th ESSCIRC, 1-4, 2005.

[Park] S. Park, C. Min and S.-H. Cho. A 95nW Ring Oscillator-based Temperature Sensor for RFID Tags in 0.13µm CMOS. Korea Advanced Institute of Science and Technology (KAIST), 1153-1156, 2009.

[Rama] V. Ramakrishnan, P. T. Balsara. A Wide-Range, High-Resolution, Compact, CMOS Time to Digital Converter. In VLSID, 1-6, 2006.

[Rich] J. Richardson, R. Walker, L. Grant, D. Stoppa, F. Borghetti, E. Charbon, M. Gersbach, R. K. Henderson. A 32x32 50ps Resolution 10 bit Time to Digital Converter Array in 130nm CMOS for Time Correlated Imaging. In CICC, 77-80, 2009.

[Roberts] G. W. Roberts. A Brief Introduction to Time-to-Digital and Digital-to-Time Converters. IEEE Trans. Circuits Syst. II, 57(3), 153-157, 2010.

[Spencer] M. Spencer, S. Callender. Digital Temperature Sensing in a Variable Supply Environment. EE 241, 1-3, 2009.

[Stasz] R. B. Staszewski, S. Vemulapalli, P. Vallur, J. Wallberg, P. T. Balsara. 1.3 V 20 ps Time-to-Digital Converter for Frequency Synthesis in 90-nm CMOS. IEEE Trans. Circuits Syst. II, 53(3), 220-224, 2006.

[Sze] S. M. Sze. Physics of Semiconductor Devices, Third Edition. Wiley-Interscience, 2006.

[Tisa] S. Tisa, A. Lotito, A. Giudice, F. Zappa. Monolithic Time-to-Digital Converter with 20ps resolution. IEEE ESSCIRC, 465–468, 2003.

[Ueno] K. Ueno, T. Asai, Y. Amemiya. Temperature-to-frequency converter consisting of subthreshold MOSFET circuits for smart temperature-sensor. In TRANSDUCERS 2009, 2433-2436, 2009.

[Weste] N. H. E. Weste and D. M. Harris. Integrated Circuit Design, Fourth Edition. Pearson, Boston, 2010.

[Woo] S.-S. Woo, J.-H. Lee, and S.H. Cho. A Ring Oscillator-based Temperature Sensor for U-Healthcare in 0.13µm CMOS. ISOCC, 548-551, 2009.

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbeit
FACHBEREICH ELEKTROTECHNIK
UND INFORMATIONSTECHNIK

Bewertung von Auswahlalgorithmen für Stromquellen in Current-Steering DAUs im Hinblick auf langreichweitig korrelierte Parameterschwankungen in Mixed Signal Chips

M. Sc. Matthias Schützkowski

Hochschule Darmstadt

Der Genauigkeit eines stromgesteuerten Digital Analog Umsetzers (current steering DAC) sind Grenzen gesetzt, die sich einerseits aus konkurrierenden Anforderungen, wie beispielsweise der Wandelgeschwindigkeit, und andererseits aus physikalischen Widrigkeiten ergeben. Zu letzteren gehören Abweichungen, die fertigungsbedingt auftreten oder die sich zu Laufzeit ergeben. Als Teil einer Masterarbeit an der Hochschule Darmstadt wurden die auftretenden Abweichungen näher untersucht. Ausgehend von einer Analyse der Ursachen, ließen sich Rückschlüsse über eine Systematik in der räumlichen Verteilung der zu erwartenden Parameterschwankungen ziehen. Die nachweislich vorhersagbare räumliche Korrelation von Parameterschwankungen ermöglicht eine Kompensation von Wandelfehlern bereits in der Designphase des DAUs. Durch eine geeignete Auswahl der einzelnen Stromquellen kann die Genauigkeit des DAUs maßgeblich verbessert werden. Vier verschiedene zu diesem Zweck einsetzbare Switchingverfahren wurden in einer Simulationsumgebung implementiert und ihre Ergebnisse auf Grundlage verschiedener Szenarien untersucht.

1. Einführung

Für ein erfolgreiches Design von präzisen analogen Elementen ist in der Mikroelektronik ein Verständnis des Matching Verhaltens in der jeweiligen Technologie enorm wichtig. Der Begriff Matching bezeichnet in diesem Zusammenhang die Übereinstimmung einzelner Bauteile, die idealerweise bei 100% liegt. In der Praxis ist eine vollkommene Übereinstimmung unmöglich. Die Ursachen für Abweichungen sind dabei Bauteilstreuung, Alterung oder Temperatureffekte.

Um die Auswirkung auf die Funktion eines stromgesteuerten Digital Analog Umsetzers nachvollziehen zu können, wird das Thema Matching mit Fokus auf MOS Transistoren in CMOS Technik untersucht. Während sich Kapazitäten in dieser Technologie sehr genau umsetzen lassen, ist die Realisierung von sich nahezu identisch verhaltenden Transistoren schwieriger. Nichtsdestotrotz werden oftmals Transistoren zur Realisierung von Digital Analog Umsetzern verwendet, da beispielsweise rein kapazitive Techniken für viele Anwendungen schlichtweg zu langsam arbeiten.

Der gegenteilige Ausdruck zum Matching ist „Mismatch“. Mismatch bezeichnet die (ungewollte) Variation zwischen einzelnen Bauteilen.

2. Systematische Abweichungen: Ursachen und Charakterisierung

Für die Schwankungen in den Strömen einzelner Stromquellen sind verschiedene Einflüsse maßgeblich:

Abweichungen in der Gateoxiddicke t_ox

Das Gateoxid dient beim MOS Transistor als Isolationsschicht zwischen Gate und Kanal. Das Verhalten des Transistors ist maßgeblich von der Gateoxiddicke abhängig, da diese direkten Einfluss auf die Gatekapazität C_ox hat. Die Oxidschicht wächst durch Oxidation in einem Schritt der Fertigungskette. Über die Fläche des Wafers gesehen, geschieht das jedoch nicht absolut gleichmäßig.

Abweichungen in der Beweglichkeit μ

Die Beweglichkeit hängt von der Verteilung der Substitutionsstörstellen im Kanal unter dem Gate ab und wird von mechanischen Verspannungen beeinflusst, die bei der Verpackung der Chips in Gehäuse entstehen.

Abweichungen in den Abmessungen

Die Abmessungen W/L werden in einem lithografischen Verfahren übertragen und in einem chemischen Verfahren hergestellt. Beide Prozesse unterliegen

Schwankungen und somit zu Abweichungen in der Dimensionierung.

Weitere Abweichungen ergeben sich teilweise erst, wenn die mikroelektronische Schaltung in Betrieb genommen ist:

Temperatur

Die (elektrische) Leistung, die im Betrieb in der Schaltung abfällt, wird in Wärme umgesetzt. Abhängig von der konkreten Umsetzung fällt die Temperaturverteilung über den Chip nicht gleichmäßig aus. Da es sich bei dem Halbleiter um einen Heißleiter handelt, ist dieser Effekt selbstverstärkend: Mit steigender Temperatur nimmt auch die Leitfähigkeit zu und es fällt lokal regelmäßig eine noch höhere Leistung ab.

Degradation

Über die Laufzeit können sich Alterungseffekte bemerkbar machen. So kann beispielsweise die Kanalbeweglichkeit abfallen oder sich die Einsatzspannung durch „gefangene Ladungsträger“ erhöhen.

Zuletzt können sich, bedingt durch das Layout, Abweichungen ergeben. Ein Spannungsabfall an den Versorgungsleitungen sei als eine derartige Fehlerquelle genannt.

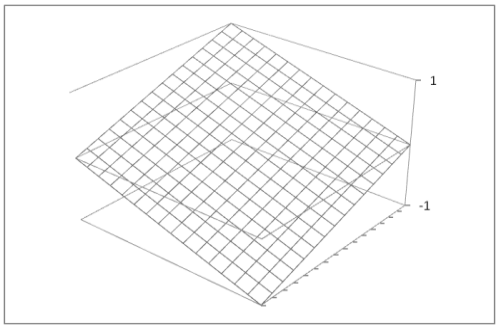


Bild 1: Normalisierter systematischer Fehler in der Oxiddicke t_ox

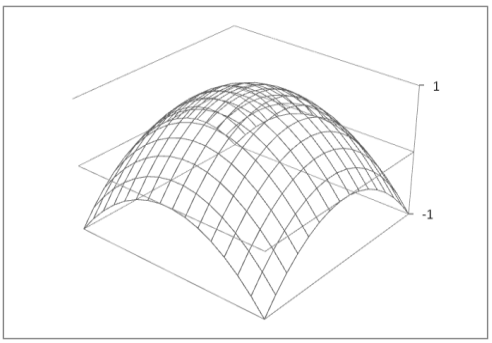


Bild 2: Normalisierter systematischer Fehler aufgrund von Temperatur- und Stresseffekten

3. Switchingverfahren

Die grundsätzliche Idee hinter einem intelligenten Switching Verfahren ist es, Annahmen über die Art und Verteilung von Fehlern zu treffen und diese gezielt gegeneinander zu kompensieren. Durch ein geeignetes Switching Verfahren kann die integrale Nichtlinearität eines Digital Analog Umsetzers verbessert werden.

Dabei ist immer maßgeblich, inwiefern die getroffenen Annahmen über den vorhandenen systematischen Fehler der Realität entsprechen. Abhängig von der konkreten Umsetzung des Switching Verfahrens ist es auch möglich, eine ganze Reihe von systematischen Fehlern zu kompensieren. Im Folgenden sollen verschiedene Ansätze zum Switching vorgestellt und anhand von Testszenarien intensiv untersucht werden.

3.1. Lineares Verfahren

Das lineare Verfahren trifft keine Annahmen über Fehler und addiert abhängig vom Eingangswert des Digital Analog Umsetzers eine Anzahl von Stromquellen auf. Die Auswahl der zu addierenden Stromquellen erfolgt dabei Zeile für Zeile und Spalte für Spalte. Die Reihenfolge der ausgewählten Zellen ist für eine 8x8 Matrix in Tab. 1 aufgeführt. Die schattierten Zellen verdeutlichen die lineare Reihenfolge

Tab. 1: Auswahlreihenfolge der Stromzellen im linearen Verfahren

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

3.2. Interlaced Verfahren

Als leichte Variation des linearen Verfahrens kann das Interlaced Verfahren gesehen werden. Auch hier wird das Zellenfeld Zeile für Zeile und Spalte für Spalte durchlaufen. Es wird dabei aber ein Zeilensprungverfahren angewandt, das in der ersten Iteration jede zweite Zelle überspringt, um sie in der zweiten Iteration auszuwählen. Die wohl bekannteste Anwendung eines solchen Zeilensprungverfahrens ist das Fernsehen, das die Übertragung eines Bildes auf zwei Halb-

bilder verteilt, die erst ineinander „gekämmt" das ganze Bild ergeben. Beim hier vorgestellten Verfahren wird allerdings nicht nur eine zeilen- sondern auch eine spaltenweise Zerlegung vorgenommen. Die Reihenfolge der ausgewählten Zellen ist für eine 8x8 Matrix in Tab. 2 aufgeführt. Die nuancierten Zellen zeigen das Sprungverfahren in einer Zeile.

Tab. 2: Auswahlreihenfolge der Stromzellen im Interlaced Verfahren

1	5	2	6	3	7	4	8
33	37	34	38	35	39	36	40
9	13	10	14	11	15	12	16
41	45	42	46	43	47	44	48
17	21	18	22	19	23	20	24
49	53	50	54	51	55	52	56
25	29	26	30	27	31	28	32
57	61	58	62	59	63	60	64

Gegenüber dem linearen Verfahren ist für die Ansteuerung nach dem Interlaced Verfahren lediglich die Verdrahtung zwischen Zeilen- beziehungsweise Spaltendekoder und Zellenfeld zu ändern. Die einzelnen Signalleitungen müssen dazu ineinander „gekämmt" werden.

3.3. Random Walk

Beim Random Walk Verfahren kommt der Zufall (engl. „random" für „zufällig") ins Spiel. Die Reihenfolge, nach der die Stromquellen zur Darstellung eines bestimmten Eingangswertes ausgewählt werden, ist willkürlich gewählt. Es sind verschiedene Szenarien denkbar:

1. Fest verdrahtet:
   Es ist genau festgelegt nach welchem Schema das Zellenfeld beim Aufaddieren der einzelnen Stromquellen durchlaufen wird. Die Reihenfolge ist einmal festgelegt und gilt als Regel für jede Digital Analog Umwandlung. Die Umsetzung der Ansteuerung geschieht ähnlich zum linearen Verfahren, lediglich die Verdrahtung ändert sich. In der Verdrahtung steckt die „Zufallsinformation". Die Monotonie des Digital Analog Umsetzers ist bei dieser Variante stets gewährleistet.
2. Zu jedem Zeitpunkt absolut zufällig:
   Die Auswahl der aufzuaddierenden Stromquellen erfolgt zufällig zur Laufzeit. Es erfolgt zyklisch eine Auswahl von Zellen entsprechend dem Eingabewert. Diese Variante liefert insbesondere in der Frequenzdomäne gute Ergebnisse. Die Ansteuerung ist relativ komplex. Neben einem Prozessor, der die Auswahl steuert, wird ein Generator zur Generierung von Zufallszahlen benötigt. Monotonie ist nicht gewährleistet.
3. Zufällige Komponente nur bei Eingangswertänderungen:
   Die ausgewählten Zellen ändern sich nur, wenn sich der Eingangswert des Digital Analog Umsetzers ändert. Erhöht sich der Eingangswert, so wird eine der Änderung entsprechend große Menge an Stromquellen zufällig ausgewählt und zu den bis dato ausgewählten Zellen hinzugefügt. Analog dazu werden bei fallendem Eingangswert zufällig ausgewählte Zellen aus der Menge der bis dato ausgewählten Zellen abgewählt. Monotonie ist insofern gewährleistet, als dass fallende Eingangswerte zu fallenden Ausgangswerten führen und steigende Eingangswerte steigende Ausgangswerte hervorrufen. Auch dieses Verfahren liefert insbesondere in der Frequenzdomäne gute Ergebnisse. Die Komplexität der Ansteuerung ist vergleichbar zu der vorgenannten Variante.

Für den Vergleich mit anderen Auswahlverfahren wird die zuerst vorgestellte Variante eingesetzt. Die Reihenfolge zur Auswahl der Stromquellen aus dem Zellenfeld steht fest und ändert sich nicht zu Laufzeit. Die Reihenfolge der ausgewählten Zellen ist für eine 8x8 Matrix in Tab. 3 aufgeführt. Die ersten vier schattierten Zellen zeigen ein Beispiel für die Zufälligkeit der Verdrahtung.

Tab. 3: Auswahlreihenfolge der Stromzellen im Random Walk Verfahren (Beispiel)

4	17	37	14	1	41	54	58
50	12	46	48	18	49	16	11
32	20	23	19	13	3	57	42
51	31	22	38	21	27	29	39
55	34	59	6	25	35	52	10
62	26	33	36	56	40	63	64
7	8	15	47	9	24	30	45
2	53	43	44	28	5	61	60

3.4. Hierarchisches Verfahren

Das Hierarchische Verfahren arbeitet mit einer statischen Reihenfolge zur Auswahl der Stromquellen aus dem Zellenfeld. Das Zellenfeld ist in vier Quadranten aufgeteilt, die bei der Auswahl möglichst gleichberechtigt genutzt werden. Dazu teilt die Ansteuerungs-

logik das Eingangswort in vier möglichst gleiche Teile und übergibt diese in der nächsten Hierarchieebene als Eingangswert an jeden einzelnen Quadranten. Die einzelnen Quadranten arbeiten autark und implementieren den gleichen Mechanismus wie das Elternelement – sie brechen ihren Eingangswert in möglichst gleiche Teile auf und beauftragen ihre vier Unterquadranten mit der konkreten Umsetzung. Die Anzahl der Hierarchieebenen ist von der Größe des Zellenfeldes abhängig. Auf unterster Ebene werden tatsächlich Stromquellen als atomare Elemente der Hierarchie ein- beziehungsweise ausgeschaltet. Die Reihenfolge der ausgewählten Zellen ist für eine 8x8 Matrix in Tab. 4 aufgeführt. Dabei verdeutlichen die schattierten Zellen die systematische Verteilung der Stromquellenauswahl über die gesamte Matrix.

Tab. 4: Auswahlreihenfolge der Stromzellen im hierarchischen Verfahren

1	17	5	21	2	18	6	22
33	49	37	53	34	50	38	54
9	25	13	29	10	26	14	30
41	57	45	61	42	58	46	62
3	19	7	23	4	20	8	24
35	51	39	55	36	52	40	56
11	27	15	31	12	28	16	32
43	59	47	63	44	60	48	64

Die Umsetzung des Hierarchischen Verfahrens ist mäßig komplex. Die Operation des Vierteilens der jeweiligen Eingangswerte lässt sich prinzipiell allein durch Verdrahtung lösen. Der Grund für diese Einfachheit liegt in der Beschaffenheit des binären Zahlensystems. Alle Multiplikationen und Divisionen mit der Zahl zwei oder ihren höheren Potenzen sind durch Schiebeoperationen zu bewerkstelligen. Ein Vierteilen läuft letztendlich auf das Weglassen der zwei niederwertigsten Signalleitungen hinaus. Eine Sonderbehandlung ist für Eingangswerte nötig, die nicht restlos durch vier teilbar sind. Der Rest der Division darf nicht Fallengelassen werden, sondern wird der Reihe nach auf die Quadranten verteilt. Sinnvollerweise bieten die einzelnen Quadranten dafür eine spezielle Schnittstelle (analog zum Carry-In bei Addierern) an.

4. Simulation

4.1. Simulationsumgebung

Nachdem der Digital Analog Umsetzer ohnehin auf Schaltplanebene umgesetzt und mittels T-Spice simuliert wurde, ist es naheliegend, den Vergleich der Switching Verfahren ebenfalls unter Verwendung des Schaltkreissimulators durchzuführen. Neben der Umsetzung der Auswahlverfahren müssen dazu diverse Kategorien systematischer Fehler im Stromzellenfeld erzeugt werden. Dies wäre im Design durch Eingriff in jede einzelne der 256 Stromzellen möglich. Insgesamt scheint dieser Aufwand aber sehr groß und als weiteres Manko ist für jeden reinen Simulationsablauf etwa eine halbe Stunde Rechenzeit des PC zu veranschlagen.

Ein zielführenderer Ansatz, den geplanten Versuch durchzuführen liegt in der Abstraktion. Für den bloßen Vergleich der verschiedenen Switchingverfahren wird in der Tat kein Schaltkreissimulator benötigt. Die Wahl des einzusetzenden Werkzeugs fiel auf das Tabellenkalkulationsprogramm Microsoft Excel. Im Arbeitsblatt der Tabellenkalkulation wird das Feld der Stromquellen nachgestellt. Dabei entsprechen die Zelleninhalte den angenommenen Strömen aus den jeweiligen Stromquellen. Zufällige und systematische Fehler können mithilfe von Formeln in das Zellenfeld eingefügt werden. Die Auswahlverfahren werden ebenfalls in der Tabellenkalkulation nachgestellt. Für jedes Auswahlverfahren werden hierzu Formeln hinterlegt, die zu jedem Eingangswert zwischen null und 256 angeben, welche Stromquellen des Zellenfeldes addiert werden sollen. Die für die Auswertung relevanten Daten wie dynamische und integrale Nichtlinearität lassen sich durch Formeln über alle simulierten Ausgangswerte hinweg ermitteln.

4.2. Fehlermodelle

Auf Basis der betrachteten Ursachen systematischer Fehler und deren Ursachen wurden verschiedene Fehlermodelle vorgefertigt und parametrierbar gemacht. So können verschieden stark ausgeprägte Fehlerbilder zu linearen und parabelförmigen Fehlerbildern bequem generiert werden. Auch Mischformen dieser Fehlerbilder und die Berücksichtigung zufälliger (nicht systematischer) Fehler ist möglich. Bild 3 zeigt die Visualisierung zu einem Fehlerbild, welches die drei Komponenten zufälliger, linearer und parabelförmiger Fehler enthält.

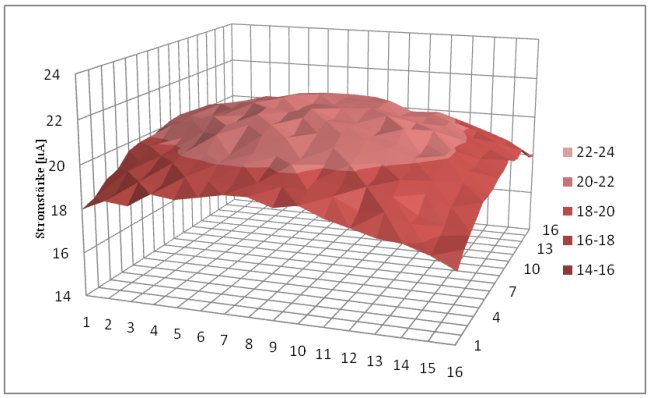


Bild 3: Fehlermodell als Simulationsbasis

4.3. Auswertung

Nach Vorgabe des jeweiligen Fehlermodells konnten die Ergebnisse der vorgestellten Switching Verfahren gemessen und gegeneinander verglichen werden. Insbesondere die integrale Nichtlinearität (INL) wird dabei als Kriterium für die Beurteilung herangezogen. Bild 4 zeigt die INL Kennlinie die für das in Bild 3 dargestellte Fehlerbild aufgenommen wurde.

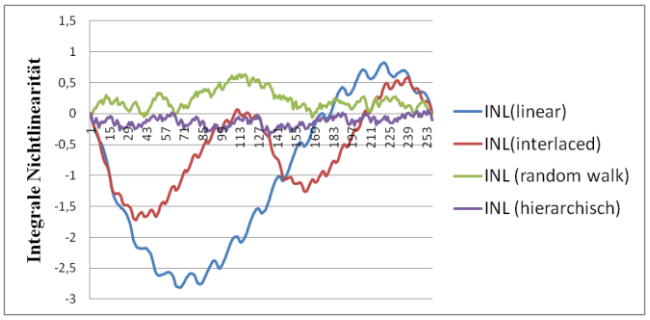


Bild 4: Integrale Nichtlinearität

Exemplarisch ist an diesem Versuch zu erkennen, welchen Einfluss eine intelligente Auswahl der zu addierenden Stromquellen haben kann. Zwischen den verschiedenen Verfahren zur Auswahl der genutzen Stromquellen ergeben sich deutlich unterschiedlich gute Ergebnisse für die erzielte INL.

Nach Durchführung und Auswertung diverser Simulationen wurde offenbar, dass insbesondere das vorgestellte hierarchische Verfahren für die angegebenen Testszenarien sehr gute Werte erzielt.

In Anbetracht der Tatsache, dass eine Umsetzung im Schaltplanentwurf des Digital Analog Umsetzers alleine durch eine geänderte Verdrahtung der Komponenten erfolgen kann, ist das Verhältnis von Aufwand zu Nutzen für die vorgestellten Switching Verfahren sehr gut. Einen Kompromiss zwischen der sehr einfachen Verdrahtung des linearen Switching Verfahrens und der sehr komplexen Verdrahtung des hierarchischen Verfahrens liefert das Interlaced Verfahren. Im Interlaced Verfahren bleibt die grundlegende Herangehensweise des zeilen- und spaltenweisen Schaltens erhalten. Durch Einsatz eines Zeilensprungverfahrens werden systematische Fehlerbilder aber zumindest zum Teil kompensiert.

Die Entscheidung, den Vergleich der Switching Verfahren nicht im Schaltkreissimulator, sondern in einem abstrakten Modell durchzuführen, hat sich als sinnvoll erwiesen. Die Aussagekraft beider Wege ist für die angestrebte Untersuchung gleich. Die Umsetzung innerhalb einer Tabellenkalkulation bietet aber deutliche Vorteile. Die Simulationszeit ist erheblich kürzer. Weiterhin ist die Eingabe der Fehlermodelle in der Tabellenkalkulation sehr viel effizienter umsetzbar. Es ist dem Anwender dabei möglich beliebige Fehlermodelle vorzugeben, diese könnten beispielsweise auch aus einer Messung einer realen mikroelektronischen Schaltung stammen. Die Implementierung der zu vergleichenden Switching Verfahren erfolgt mittels Formeleingabe. Auf Basis einer Berechnungsvorschrift ist die Simulation somit grundlegend um weitere Verfahren erweiterbar. Zuletzt bietet die Verwendung einer Tabellenkalkulation in der Auswertung der simulierten Ergebnisse große Vorteile. Die grafische Aufbereitung von Tabellen in zwei- oder mehrdimensionalen Diagrammen gehört zum Standardrepertoire des eingesetzten Werkzeugs. Auch diverse Funktionen der Statistik können ohne großen Aufwand auf die simulierten Daten angewendet werden. Als ein Vergleichsmerkmal wurde so beispielsweise die Summe der Fehlerquadrate (SFQ) über alle INL Werte der vier Verfahren errechnet.

Mikrostrukturierte Energiewandler für energieautonome Sensoren

Dennis Hohlfeld[1], Ruud Vullers[2], Rob van Schaijk[2]

[1]Hochschule Reutlingen, Alteburgstr. 150, 72762 Reutlingen

[2]Holst Centre / IMEC, High Tech Campus 31, 5656 AE Eindhoven, Die Niederlande

Tel.: 07121/271-7081, Fax: 07121/271-7004, dennis.hohlfeld@reutlingen-university.de

Diese Arbeit präsentiert mikrostrukturierte Energiewandler zur Stromversorgung energieautarker Sensoren. Vibrationsbasierte Energiewandler mit Betriebsfrequenzen von 100 bis 1000 Hz nutzen den piezoelektrischen oder elektrostatischen Effekt. Wir beschreiben ebenso ein drahtloses energieautonomes Sensorsystem, welches von einem Vibrationswandler gespeist wird und alle 15 Sekunden Temperaturmessungen drahtlos überträgt. Mikrostrukturierte Thermoelemente werden darüber hinaus als kostengünstige Lösung für thermoelektrische Generatoren angesehen. Diese sollen bei geringen Temperaturunterschieden und schwachen Wärmeströmen arbeiten, die z.B. für menschliche Körperwärme typisch sind.

1. Einleitung

Mit dem Begriff „Energy Harvesting“ bezeichnet man eine Technologie, die es ermöglicht elektrische Leistung zu gewinnen, die als Umgebungsenergie vorhanden ist und nicht anderweitig genutzt wird. Hierzu zählen Temperaturunterschiede, Vibrationen oder elektromagnetische Strahlung wie z.B. hochfrequente Wellen und sichtbares Licht. Drahtlose Sensormodule können von solchen Energiewandlern profitieren. Die Lebenszeit solcher Module wird durch deren Energiespeicher bestimmt, welcher in den meisten Fällen eine Batterie ist. Das Ersetzen bzw. Aufladen dieser Batterien ist in vielen Anwendungen unerwünscht oder nicht machbar.

Ein Energiewandler kann genutzt werden, um kontinuierlich elektrische Leistung zu generieren und damit das Sensorsystem zu betreiben bzw. den Energiespeicher zu füllen. Damit kommt ihm eine Schlüsselrolle im Einsatz energieautarker drahtloser Sensormodule und Sensornetzwerke zu.

Vibrationsbasierte Energiewandler basieren auf verschiedenen physikalischen Prinzipen. Hier wird in Vibration vorhandene mechanische Energie in elektrische Energie gewandelt und als Leistung zur Verfügung gestellt. Piezoelektrische, elektrostatische und elektromagnetische Wandler zählen zu den gebräuchlichsten Varianten.

Die Vibrationsfrequenzen von Maschinen und Fahrzeugen können bis zu einigen kHz reichen und Anregungsamplituden im Bereich des Mehrfachen der Erdbeschleunigung aufweisen. Die absoluten Vibrationsamplituden liegen im Bereich weniger Mikrometer. Daher ist ein mechanischer Resonator notwendig, der die Anregungsamplituden verstärkt, wie sie für eine effiziente Energiewandlung notwendig sind. Mikrostrukturierung ist besonders gut zur Herstellung solcher Energiewandler geeignet, da sie eine kostengünstige Fabrikation von miniaturisierten Wandlern bei gleichzeitig hoher Präzision und Reproduzierbarkeit erlaubt.

Temperaturunterschiede innerhalb von Objekten oder auf deren Oberfläche (Maschinen, Gebäude, Rohrleitungen) sowie auf der Haut von Tieren und Menschen können ebenso zur Leistungsversorgung von energieautonomen Systemen genutzt werden. Kürzlich sind erste tragbare drahtlose Sensoren mit medizinischen Anwendungen (drahtlose Sauerstoffsättigung, Elektroenzephalogramm) vorgestellt worden. Die werden ausschließlich von mit Körperwärme gespeisten thermoelektrischen Generatoren (TEG) betrieben [1-3]. Der neuartige TEG wird anfänglich in sogenannten Body area networks[1] eingesetzt, welche am Holst Centre entwickelt werden [4].

2. Vibrationsbasierte Energiewandler

Der typische durchschnittliche Leistungsbedarf von drahtlosen Sensoren beträgt bei entsprechender Wahl der Komponenten und des Arbeitszyklus 100 µW. Um

[1] Body Area Network (BAN) steht für eine Übertragungstechnologie in der Telemedizin. Mit dieser Technologie ist eine drahtlose Anbindung von am Körper getragenen medizinischen Sensoren und Aktoren möglich.

einen vollständig autonomen Betrieb zu gewährleisten sind Energiewandler ein Schlüsselelement [5].

2.1. Piezoelektrischer Energiewandler

In dieser Arbeit stellen wir piezoelektrische Energiewandler mit Abmessungen unterhalb 1 cm^2 vor. Diese decken den Frequenzbereich von 200 bis 1200 Hz ab. Aluminiumnitrid (AlN) wurde als piezoelektrisch aktives Material eingesetzt. Obwohl der piezoelektrische Koeffizient von AlN niedriger ist als der Wert für Blei Zirkonat Titanat (PZT) [6], ist AlN in vergleichbarer Weise zur Energiewandlung geeignet. Dies ist in der geringeren Permittivität begründet. Die Leistungsdaten AlN-basierten Energiewandler erreichen, bzw. übertreffen, die Daten von PZT-Wandlern. AlN bietet zudem den Vorteil, dass eine Abscheidung im Sputterverfahren möglich ist, wohingegen PZT komplexere Herstellungsverfahren erfordert.

2.1.1 Wandlerentwurf

Die vorgestellten piezoelektrischen Wandler bestehen aus einer trägen Masse die an einem Balken befestigt ist. Beide Komponenten werden aus Silizium mikrostrukturiert. Das piezoelektrische Material wird ganzflächig auf dem Balken strukturiert und an seiner Ober- und Unterseite mit Elektroden versehen. Diesen bestehen aus Aluminium (obere Elektrode) und Platin (untere Elektrode). Die Wandler sind mit verschiedene Balken- und Massenabmessungen gefertigt worden. Sämtliche in diesem Beitrag gezeigten Ergebnisse wurden an Strukturen mit einer Balkendicke von 45 ± 3 µm und einer AlN-Dicke von 800 nm gewonnen. Die Wandler sind zwischen zwei Glasdeckeln gehäust, in die Vertiefungen in einer Dicke von 400 µm geätzt wurden. Dies erlaubt der Masse eine entsprechende Auslenkung.

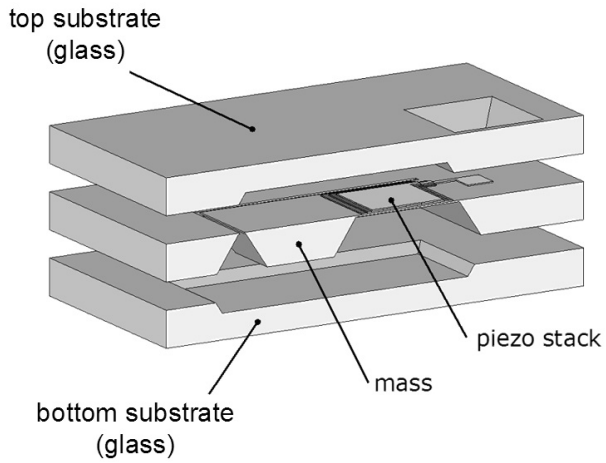


Bild 1: Vibrationsbasierter Energiewandler, verkapselt zwischen zwei Glassubstraten. Die Masse kann sich in beide Richtungen auslenken.

Die Bilder 1 und 2 zeigen das Design des Wandlers und die Verkapselung zwischen zwei Glassubstraten.

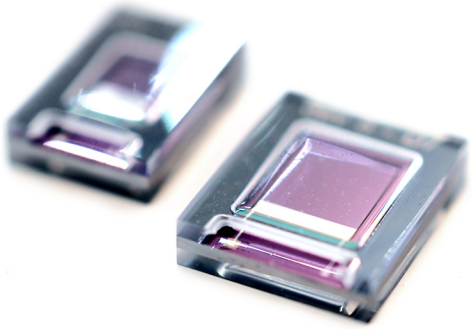


Bild 2: Energiewandler unterschiedlicher Abmessungen (links: 7×12 mm^2 , rechts: 10×12 mm^2) mit Verkapselung zwischen zwei Glassubstraten.

2.1.2 Messergebnisse

Wir haben AlN-basierte piezoelektrische Energiewandler auf einem Schwingtisch charakterisiert. Diese Messumgebung basiert auf einem elektrodynamischen Aktuator und einem Regler um in Frequenz und Amplitude definierte harmonische Anregungen zu generieren. Hierbei wurde die Beschleunigungsamplitude als konstant vorgegeben und die Anregungsfrequenz variiert. Sobald ein Widerstand an den Kondensator des Energiewandlers angeschlossen wird fließt ein Wechselstrom.

Um eine optimale Leistungsanpassung zu erzielen, muss der Wert des Lastwiderstands an den Energiewandler angepasst werden. Die hier charakterisierten AlN-basierten Wandler benötigen Lastwiderstände in der Größenordnung von 0.1 bis 1 MΩ. PZT-basierte Wandler hingegen erfordern Lastwiderstände im Bereich weniger kΩ. Dieser Unterschied ist in den unterschiedlichen Permittivitäten und piezoelektrische Koeffizienten der beiden Materialien begründet. Die Ausgangsspannungen von Wandlern mit AlN bzw. PZT sind daher auch unterschiedlich. AlN-basierte Wandler zeigen Ausgangsspannungen von mehreren Volt, wohingegen PZT-basierte Wandler niedrigere Spannungen erzeugen; dies allerdings bei größeren Stromstärken. Die generierte Leistung ist daher vergleichbar. Dennoch wird die höhere Ausgangsspannung von AlN-basierten Wandlern favorisiert, da eine Gleichrichtung mit herkömmlichen Brückenschaltungen erfolgen kann. Ebenso ist eine Herauf-Konversion der Spannungspegel nicht erforderlich.

Bild 3 zeigt das Resonanzverhalten eines ungehäusten Wandlers. Die Masse kann sich ungehindert in der Luft bewegen. Die Resonanzkurve zeigt eine geringfügige Asymmetrie. Dies kann auf nicht-lineares Verhalten der Balkenaufhängung zurückgeführt werden.

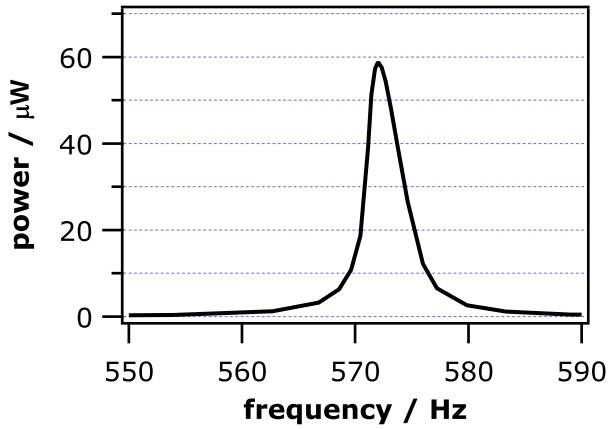


Bild 3: Resonanzkurve mit einer maximalen Ausgangsleistung von 60 µW bei einer Anregung mit 2 g bei 572 Hz..

In Resonanz beträgt die Leistungsabgabe 60 µW. Dies ist vergleichbar mit PZT-basierten Wandlern [6]. Die Anregungsamplitude der Beschleunigung beträgt 2.0 g bei einer Frequenz von 572 Hz. Die doppelte Scheitelwert (Spitze-Spitze) der Massenauslenkung beträgt 895 µm. Aus der Resonanzkurve lässt sich mittels eines Oszillatormodels die Güte ablesen. Diese beträgt 170.

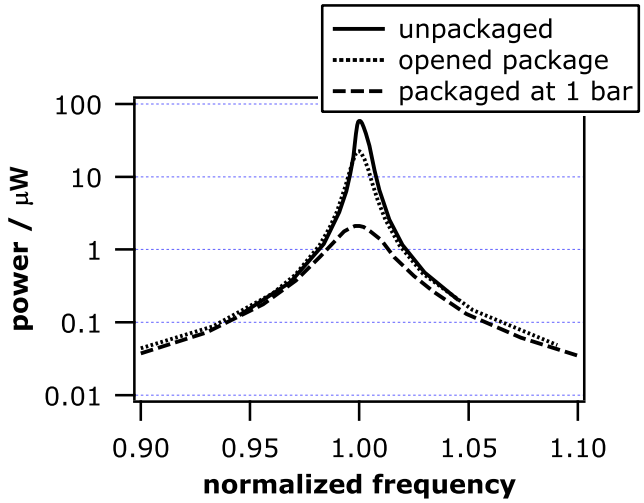


Bild 4: Resonanzkurve dreier Energiewandler unter verschiedenen Luftdämpfungen.

Bild 4 zeigt einen Vergleich von drei Energiewandlern unter verschiedenen Umgebungsbedingungen. Die den Wandler umgebende Luft wirkt sich auf die Dämpfung des Resonators aus. Ein unter Normaldruck gehäuster Wandler hat eine maximale Ausgangsleistung von 2.1 µW. Ein Wandler mit (im Bereich des Endes der Masse) teilweise geöffnetem Gehäuse erbringt 22 µW. Der ungehäuste Wandler zeigt, wie erwähnt, 60 µW Ausgangsleistung. Alle Daten sind bei resonanter Anregung mit 2 g gemessen worden. Die entsprechenden Gütefaktoren betragen 37, 110 und 170.

Die Frequenzachse wurde normalisiert um geringfügige Abweichungen in der Resonanzfrequenz zu kompensieren. Die Luftdämpfung reduziert die maximale Auslenkung der Masse und stellt einen mechanischen Verlustmechanismus dar. Dies macht sich in dem reduzierten Gütewert bemerkbar. Ein weiteres Merkmal ist die vergrößerte Bandbreite der stärker gedämpften Systeme. Die Leistungsabgabe ist jedoch bei allen Frequenzen und Dämpfungswerten durch die Resonanzkurve des offenen Systems begrenzt.

Eine Verbesserung lässt sich erwarten, wenn die Luftdämpfung des gehäusten Wandlers reduzierte wird. Dies lässt sich durch hermetische Kapselung bei reduziertem Druck erreichen. Die Bewegung der Masse wird in diesem Fall nicht durch vorhandene Luft gehindert, was zu größerer Auslenkung und höherer Leistungsabgabe führt.

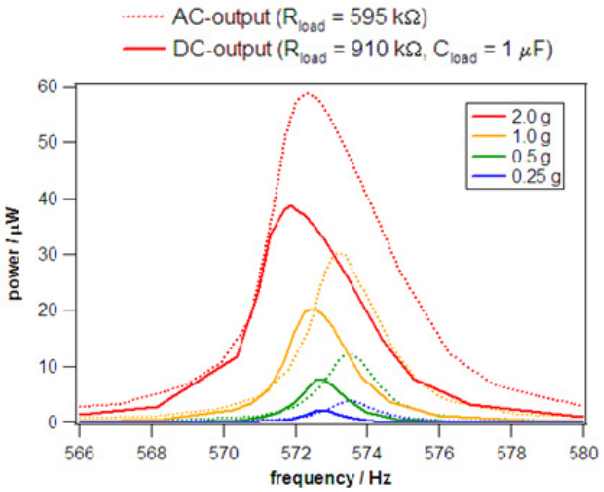


Bild 5: Resonanzkurve der Verlustleistung in einem Widerstand, bei direkter Verbindung zwischen Wandler und Last bzw. bei zwischengeschalteter Gleichrichtung. Bei Verwendung der Gleichspannung entspricht die im Widerstand umgesetzte Leistung 65 % des Wertes bei direkter Verbindung.

2.1.3 Gleichrichtung der Ausgangsspannung

Mit einem Brückengleichrichter aus Schottky-Dioden und einem Kondensator lässt sich die Wechselspannung des Energiewandlers in eine Gleichspannung umformen. Die Dioden weisen in Durchlassrichtung einen niedrigen Spannungsabfall von 300 mV auf. Zugleich zeigen sie in Sperrrichtung einen extrem niedrigen Leckstrom von 20 nA. Um eine ausreichende Glättung der Spannung zu erzielen, sollte die Kapazität des Kondensators größer als 1 µF sein. Die bei Gleichspannung verbrauchte Leistung ist auf 65 %

des Wertes reduziert, der bei direktem Anschluss des Lastwiderstands an den Kondensator des Energiewandlers erreicht wird (siehe Bild 5).

2.1.4 Drahtloses energieautonomes System

Die oben beschriebene Anordnung zur Generierung von Gleichspannungen mit vibrationsbasierten Energiewandlern wurde zur Versorgung eines drahtlosen Temperatursensors verwendet (siehe Bild 6). Dadurch wurde das System vollständig unabhängig von anderen Energiequellen, wie z.B. Batterien. Der Energiewandler ist die einzige Energiequelle im System.

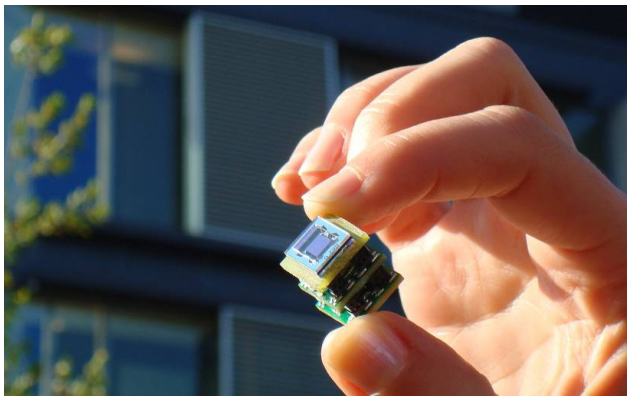


Bild 6: Drahtloses energieautonomer Temperatursensor mit vibrationsgetriebenem piezoelektrischen Energiewandler.

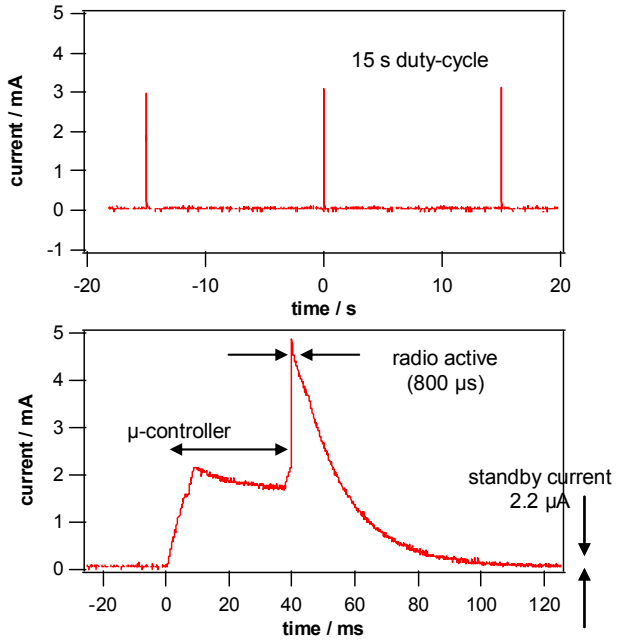


Bild 7: Stromaufnahme des energieautonomen Funkmoduls. Ein energieautonomer Betrieb konnte mit einem Zyklus von 15 s erreicht werden. Der piezoelektrische Energiewandler wurde mit einer Frequenz von 300 Hz erregt.

Das System nutzt ein energieeffizientes Funkmodul, Mikrocontroller und Spannungswandler (Ausgangsspannung 2.2 V). Die Stromaufnahme im Standby-Modus beträgt 2.2 µA. Die Sensordaten werden in einem Zyklus von 15 s per Funk versendet. Bei dieser Periode wird die durchschnittliche Leistungsaufnahme des Moduls durch die Leistungsabgabe des Energiewandlers ausgeglichen. Zu Beginn der aktiven Phase des Moduls ist der Mikrocontroller für 40 ms in Betrieb, Im Anschluss versendet das Funkmodul die Daten in einem Funkimpuls von 800 µs Dauer. Die Charakteristik der Stromaufnahme ist in Bild 7 dargestellt.

2.2. Elektrostatischer Energiewandler

Eine elektrostatische Energiewandlung erfordert einen variablen Kondensator. Die Kapazität die durch zwei parallele Elektroden gebildet wird ist durch die Fläche und den Abstand der Platten sowie ein eventuell vorhandenes Dielektrikum zwischen den Elektroden bestimmt. Änderungen dieser Parameter führen zu Variationen in der Kapazität. Durch Kombination eines variablen Kondensators mit einer Ladungsquelle kann ein elektrostatischer Energiewandler hergestellt werden.

Ähnlich wie ein piezoelektrischer Energiewandler nutzt auch ein elektrostatischer Wandler einen mechanischen Resonator aus einer trägen Masse und einem oder mehreren Aufhängungselementen. Vibrationen in der Umgebung eines solchen Energiewandlers können resonante Oszillationen anregen. Die Auslenkung der Masse wird dann genutzt um die bewegliche Elektrode eines Plattenkondensators zu bewegen. Verschiedene Schaltungstechniken sind realisierbar, die die Kapazitätsänderung nutzen, um elektrische Leistung aus der im Resonator gespeicherten mechanischen Energie zu gewinnen.

Mikrostrukturierte elektrostatische Energiewandler können entweder eine Änderung der Elektrodenüberlappung oder des Elektrodenabstands nutzen. Bei Wandlern mit veränderlicher Elektrodenüberlappung kann die Elektrodenfläche entweder parallel [8-11] oder senkrecht [12] zur Oberfläche des Substrats orientiert sein. Wandler die eine Änderung des Abstands der Elektroden nutzen sind ebenso untersucht worden. Auch hier können die Elektroden parallel [13-14] oder senkrecht [15-16] zur Ebene des Substrats angeordnet werden.

2.2.1 Wandler mit variabler Elektrodenüberlappung

In diesem Abschnitt wird ein mikrostrukturierter elektrostatischer Energiewandler beschrieben. Die Auslenkung der seismischen Masse erfolgt parallel zum Substrat und verändert damit den Überlapp der Elekt-

roden. Der variable Kondensator wird über ein Elektret elektrisch vorgespannt. Die Strukturen sind ausgelegt, um typische Vibrationsfrequenzen industrieller Maschinen zu nutzen, d.h. Frequenzen im Bereich 500 – 2500 Hz. Der Wandler wird aus mehreren Substraten gefertigt, die mittels klebender Verbindungen zusammengefügt werden. Zur Fertigung der variablen Kapazität werden drei Substrate benötigt: ein Glassubstrat trägt eine Elektrodenanordnung, ein weiteres Substrat aus Silizium beinhaltet den mechanischen Resonator mit der Gegenelektrode. Das dritte Substrat trägt das Elektret (siehe Bild 8).

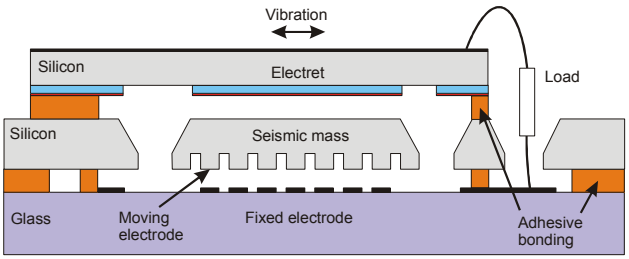


Bild 8: Elektrostatischer Energiewandler mit variabler Elektrodenüberlappung. Der Kondensator wird über ein Elektret elektrisch vorgespannt.

Die maximale Ausgangsleistung wurde zu 380 nW bestimmt. Dieser Wert wurde bei einer Vorspannung von 10 V und einem Lastwiderstand von 1 MΩ ermittelt. Eine Optimierung des Herstellungsprozesses lässt eine 10 bis 100-fache Steigerung der Ausgangsleistung auf 4 bis 40 µW erwarten.

2.2.2 Wandler mit variablem Elektrodenabstand

Diese Wandlervariante nutzt zwei Siliziumsubstrate, welche eutektisch verbunden werden. Die seismische Masse wird aus dem unteren Substrat gefertigt. Das obere Substrat wird genutzt, um die Elektroden und die Aufhängungselemente zu strukturieren.

Dieser variable Kondensator nutzt Elektroden die senkrecht zum Substrat orientiert sind (siehe Bild 9). Eine Elektrodenanordnung ist mechanisch mit dem Substrat verbunden, wohingegen die anderen Elektroden mit der seismischen Masse verbunden sind. Eine Auslenkung der Masse führt zu einer Änderung des Elektrodenabstands und einer Änderung der Kapazität. Da die beweglichen Elektroden elektrisch kontaktiert werden müssen, muss an der Verbindungsstelle zur Masse sowohl ein elektrischer als auch ein mechanischer Kontakt sichergestellt sein. Daher wurde eine Gold-basierte eutektische Verbindungstechnik gewählt.

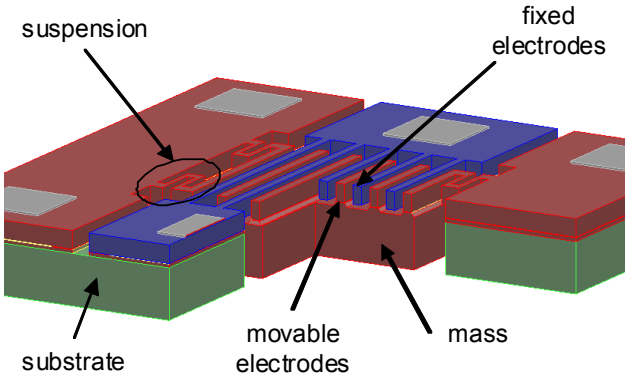


Bild 9: Elektrostatischer Energiewandler mit variablem Elektrodenabstand. Die Elektroden und die Masse sind übereinander angeordnet.

Das zugrundeliegende Funktionsprinzip basiert auf einem variablen mikrostrukturiertem Kondensator. Dessen Kapazität verändert sich bei resonanter Anregung des mechanischen Oszillators. Ladung wird einer Spannungsquelle entnommen, auf einem variablen Kondensator isoliert und in einen größeren Speicherkondensator transferiert. Das elektrische Potential der Ladung wird dabei erhöht, so dass der Speicherkondensator sich durch wiederholtes Zuführen von Ladung auflädt. Die erforderliche Energie wird dem mechanischen Resonator entnommen, indem die Elektroden entgegen der Wirkrichtung der elektrostatischen Kraft bewegt werden. Diese Energiewandlung verursacht eine Zunahme des Potentials der Ladung die auf dem variablen Kondensator gespeichert ist. Das Funktionsprinzip des Energiewandlers ähnelt daher dem einer elektronischen Ladungspumpe zur Heraufsetzung von Spannunsgpegeln.

Der variable Kondensator ist ein Schlüsselelement in der zugehörigen Wandlerschaltung (siehe Bild 10). Zwei Schalter (S1, S2) kontrollieren die Lade- und Entladevorgänge in der Schaltung.

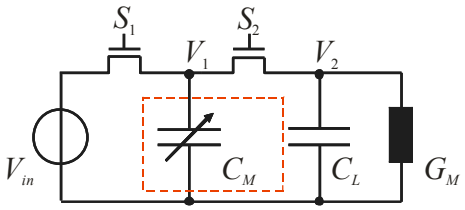


Bild 10: Elektrische Schaltung für einen elektrostatischen Energiewandler. Anstatt der aktiven Schaler können ebenso Dioden als passive Schalter eingesetzt werden.

Der Einsatz von getakteten induktiven Rückkopplungen kann die externe Spannungsquelle ersetzen. Die Quelle zur Bereitstellung von Ladung kann daher

durch den Energiewandler selbst generiert und betrieben werden.

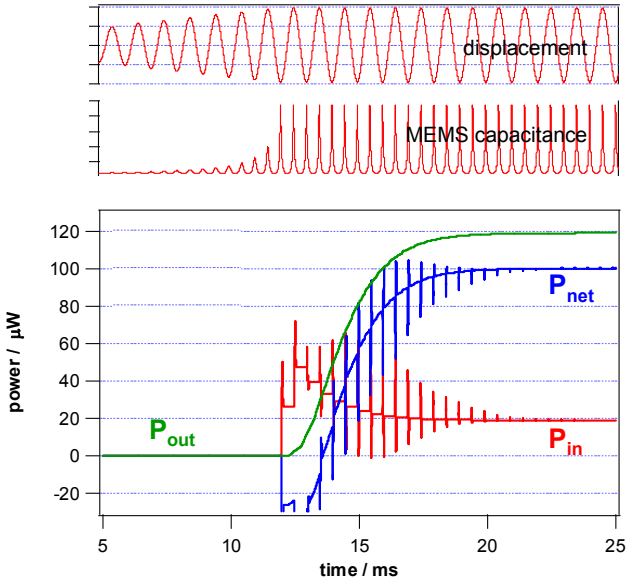


Bild 11: Anfangsphase der elektrostatischen Energiewandlung. Zu Beginn wird der Spannungsquelle elektrische Leistung entnommen, um den variablen Kondensator wiederholt zu laden.

Der Einsatz von Simulationstechniken auf Systemebene ermöglicht, die Ausgangsleistung zu bestimmen. Zu Beginn wird der variable Kondensator wiederholt aus der Spannungsquelle geladen. Diese Vorgänge führen zu einer Leistungsentnahme aus der Quelle. Diese Anfangsphase ist in Bild 11 dargestellt.

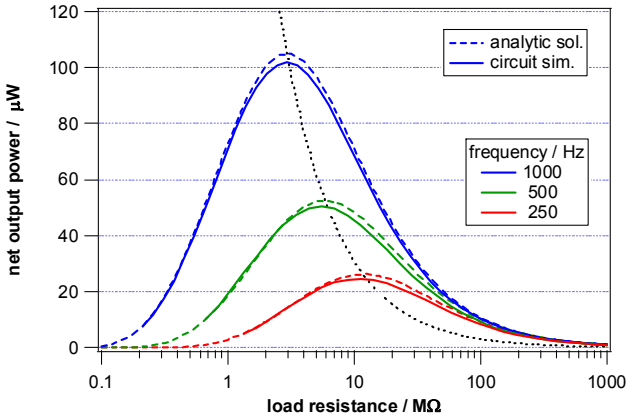


Bild 12: Anpassung des Lastwiderstands. Der Wert des Lastwiderstands bei maximaler Leistungsabgabe wird auch durch die Betriebsfrequenz bestimmt.

Nachdem sich ein Gleichgewicht zwischen gewonnener und verbrauchter Leistung eingestellt hat, ist die Leistungsentnahme aus der Quelle geringer. Um eine Aussage über den tatsächlichen (netto) Leistungsgewinn machen zu können, kann ihr Einfluss jedoch nicht vernachlässigt werden. Führt man diese transienten Systemsimulationen wiederholt für unterschiedliche Werte des Lastwiderstands durch, so erhält man die Daten aus Bild 12

Es konnte gezeigt werden, dass der aus Simulationen ermittelte optimale Lastwiderstand hervorragend mit den Ergebnissen aus einer analytischen Betrachtung übereinstimmt [17]. Die Ausgangsleistung wird linear durch die Betriebsfrequenz beeinflusst.

3. Thermoelektrische Generatoren

Die Energieausbeute mikrostrukturierter Thermogeneratoren (siehe Bild 13) wird derzeit durch technologische Randbedingungen beschränkt. Die Höhe des Thermoelements muss so groß wie möglich sein. Ein Wert von 10 bis 15 µm würde die theoretisch maximal erzielbare Energieausbeute erzielen; wohingegen die in dieser Arbeit vorgestellten Thermoelemente mit einer Höhe von 6 µm praktisch nutzbare Leistungen ergeben sollten. Bei Anwendungen mit geringen Wärmeflüssen sollte die Anzahl an Thermoelementen pro Flächeneinheit maximiert werden, um nutzbare Ausgangsspannungen zu erhalten. Daraus folgt eine laterale Abmessung der Thermoelemente im Bereich weniger µm. Die Lithographie beschränkt demnach die Energieausbeute des thermoelektrischen Generators. Um gute Ausgangsleistungen zu erzielen, sollte der Abstand zweier Thermopaare weniger als 2 µm betragen; und dies über Höhenunterschiede von 6 bis 10 µm.

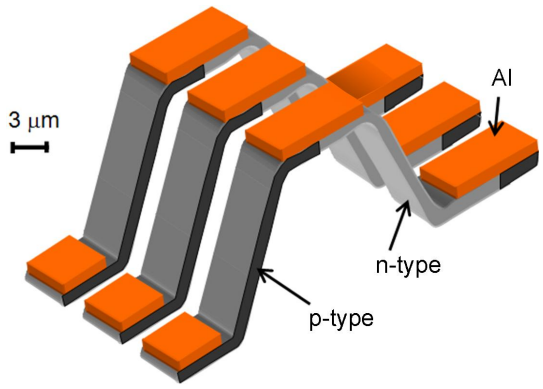


Bild 13: Thermoelemente in „Säulengang“-Anordnung (drei Thermoelemente sind gezeigt).

Der Einsatz von Belichtungswerkzeugen mit großer Tiefenschärfe erbringt nicht nur eine quantitative Verbesserung in der Energieausbeute thermoelektrischer Generatoren, sondern ermöglicht auch neue Entwurfsfreiheiten. Ebenso werden die Herstellung und die Integration erheblich vereinfacht. Letztendlich ermöglicht diese Technologie die Realisierung von thermoelektrischen Generatoren mit nahezu theoretisch optimaler Energieausbeute.

In dieser Arbeit stellen wir Fortschritte in der Herstellung von mikrostrukturierten Thermogeneratoren unter Verwendung fortschrittlicher Belichtungswerkzeuge vor. Die Herstellung von Thermoelementen mit einer Höhe von 6 µm und einer Breite von 3 µm konnte erfolgreich demonstriert werden. Dadurch konnten die Simulationsergebnisse aus [18] bestätigt werden.

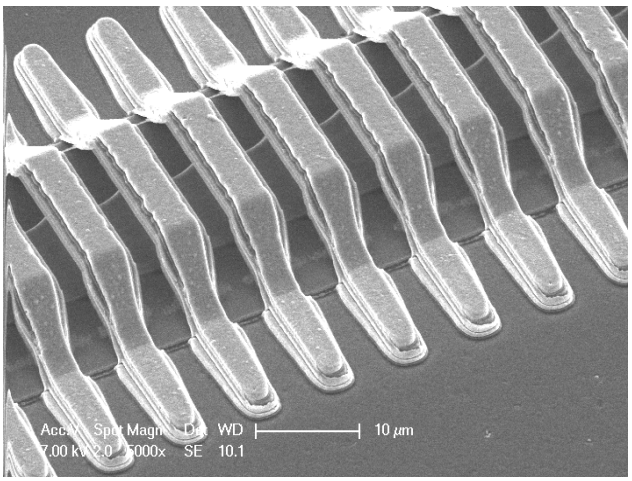


Bild 14: Mikrostrukturierte Thermoelemente mit einer Höhe von 6 µm. Leiterbahnen aus Aluminium wurde über einer Opferschicht aus Siliziumoxid hergestellt. Das Oxid wurde nasschemisch entfernt.

3.1. Herstellung

Als Opferschicht wird eine 6 µm dicke Schicht aus Siliziumoxid hergestellt. Auf diese werden 150 nm Siliziumnitrid und anschließend 1 µm polykristallines Silizium (poly-Si) abgeschieden. Nach dem Ätzen des Oxids entstehen Stufen in einer Höhe von 6 bzw. 7 µm. Das Aufschleudern von Fotolack über diese Topographie führt zu dünnen Lackschichten auf den erhöhte Bereichen und dickeren Schichten auf dem Boden. Die Energie zur vollständigen Belichtung der Lackschichten hängt von deren Dicke ab. Eine ausreichende Belichtungsdauer für die Lackschicht am Boden führt zur Überbelichtung der dünneren Lackschicht auf den oberen Bereichen. Dies kann durch Anpassung der Geometrie der zu strukturierenden poly-Si Bereiche kompensiert werden. Dem Effekt der Überbelichtung trägt man Rechnung, indem die Strukturen auf der Fotomaske in geeigneter Weise breiter entworfen werden. Damit sind die Maskenstrukturen für Elemente auf den oberen Bereichen breiter als für Elemente die auf dem Boden belichtet werden sollen. In ähnlicher Weise lässt sich der Verlauf des poly-Si auf der Kante des Oxids kontrollieren. Die Thermoelemente wurden anschließend freigeätzt. Zu diesem Zweck wurde das poly-Si mit Aluminium bedeckt und entsprechend strukturiert. Eine nasschemische Ätzung entfernt das Oxid unter den poly-Si/Al Stegen. Die fabrizierte Struktur ist in Bild 14 gezeigt.

4. Zusammenfassung

In dieser Arbeit wurden mikrostrukturierte Energiewandler vorgestellt, die elektrische Leistung aus Vibrationen bzw. Temperaturdifferenzen gewinnen. Die erforderlichen Technologien zur Herstellung solcher Energiewandler sind ebenso beschrieben worden. Die Leistung eines vibrations-basierten Wandlers wurde benutzt, um einen drahtlosen Temperatursensor zu betreiben. Die Messergebnisse wurden alle 15 s mittels Funk übertragen.

Literatur

[1] Leonov, V., Fiorini, P., Sedky, S., Torfs, T., and Van Hoof, C., “Thermoelectric MEMS Generators as a Power Supply for a Body Area Network,” Proc. 13th Int. Conf. on Solid-State Sensors, Actuators and Microsystems (Trans-ducers’05), IEEE, 2005, pp. 291-294.

[2] Torfs, T., Leonov, V., Van Hoof, C., and Gyselinckx, B., “Body-Heat Powered Autonomous Pulse Oximeter,” Proc. IEEE Int. Conf. on Sensors, IEEE, 2006, pp. 427-430.

[3] Van Bavel, M., Leonov, V., Yazicioglu, R. F., Torfs, T., Van Hoof, C., Posthuma, N., and Vullers, R. J. M., “Wearable Battery-Free Wireless 2-Channel EEG Systems Powered by Energy Scavengers,” *Sensors & Transducers*, 2008, Vol. 94, No. 7, 2008, pp. 103-115.

[4] Gyselinckx, B., Van Hoof, C., Ryckaert, J., Yazicioglu, R., Fiorini, P., Sedky, S., Torfs, T., and Leonov, V., “Human++: Autonomous Wireless Sensors for Body Area Networks,” Proc. Custom Integrated Circuit Conference, IEEE, 2005, pp. 13-19.

[5] S. Roundy, P.K. Wright and J.M. Rabaey, Energy scavenging for wireless sensor networks, Kluwer Academic Publishers, ISBN 1-4020-7663-0

[6] M. Renaud, T. Sterken, A. Schmitz, P. Fiorini, C. Van Hoof and R. Puers, Piezoelectric harvesters and MEMS technology: fabrication, modeling and measurements, Transducers pp. 891, 2007

[8] T. Tsutsumino, Y. Suzuki, and N. Kasagi, "Electromechanical Modeling of Micro Electret Generator for En-ergy Harvesting," in Solid-State Sensors, Actuators and Microsystems Conference, TRANSDUCERS, 2007, pp. 863-866.

[9] T. Sterken, P. Fiorini, G. Altena, C. Van Hoof, and R. Puers, "Harvesting Energy from Vibrations by a Micromachined Electret Generator," in TRANSDUCERS, 2007, pp. 129-132.

[10] A. M. Paracha, P. Basset, F. Marty, A. V. Chasin and T. Bourouina, "A high power density electrostatic vibration-to-electric energy converter based on an in-plane overlap plate (IPOP) mechanism" in DTIP, 2007, p. 7.

[11] M. A. Mahmoud, E. F. El-Saadany, and R. R. Mansour, "Planar Electret Based Electrostatic Micro-Generator," in PowerMEMS , 2006, pp. 223-226.

[12] U. Bartsch, A. Trautmann, P. Ruther, J. Gaspar, and O. Paul, "Electrostatic Transducers for Micro Energy Harvesting Based on SOI Technology," in Solid-State Sensors, Actuators and Microsystems Conference, TRANSDUCERS 2007, pp. 141-144.

[13] I. Kuehne, A. Frey, D. Marinkovic, G. Eckstein, and H. Seidel, "Power MEMS—A capacitive vibration-to-electrical energy converter with built-in voltage " *Sensors and Actuators A: Physical*, 142, pp. 263-269, 2008.

[14] P. D. Mitcheson, P. Miao, B. H. Stark, E. M. Yeat-man, A. S. Holmes, and T. C. Green, "MEMS electro-static micropower generator for low frequency operation," *Sensors and Actuators A-Physical*, 115, pp. 523-529, 2004.

[15] Y. Chiu, C.-T. Kuo, and Y.-S. Chu, "MEMS design and fabrication of an electrostatic vibration-to-electricity energy converter," *Microsystem Technologies*, 13, pp. 1663-1669, 2007.

[16] G. Despesse, T. Jager, S. Basrour, J. J. Chaillout, B. Charlot, J. M. Léger, and A. Vassilev, "Fabrication and characterization of high damping electrostatic micro devices for vibration energy scavenging," in DTIP, Montreux, Switzerland, 2005.

[17] D. Hohlfeld, G. Altena, M. Goedbloed, Rob van Schaijk, “A new gap-closing electrostatic energy harvester employing electrodes vertically integrated with a seismic mass”, Proc. MicroMechanics Aachen, 2008.

[18] Leonov, V., Wang, Z., Pellens, R., Gui, C., Vullers, R. J. M., Su, J., “Simulations of a non-planar lithography and of performance characteristics of arcade microthermopiles for energy scavenging,” Proc. 5th Int. Energy Conversion Eng. Conf. (IECEC), AIAA-2007-4782, pp. 1-15.

Technische Universität Darmstadt

MPC-Dienste: ihr Beitrag für die Lehre und Forschung in der Mikroelektronik

Prof. Dr. Dr. h.c. mult. Manfred Glesner

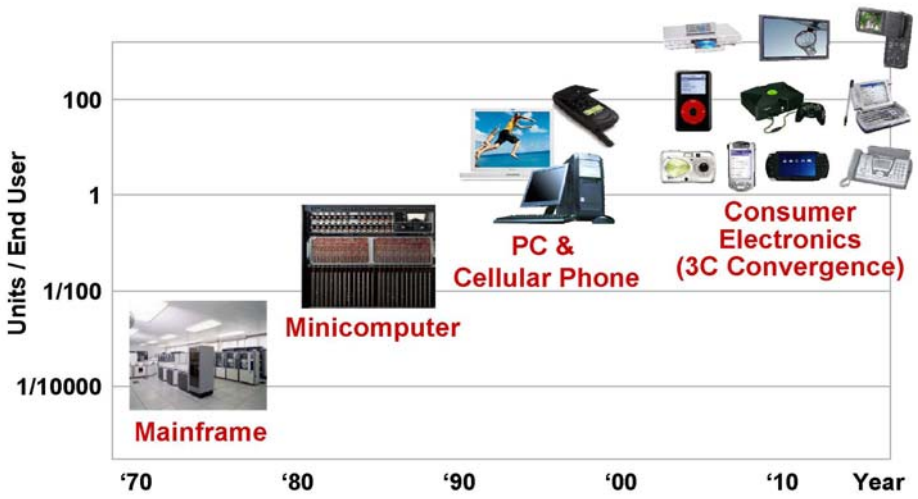
With the support of:
Massoud Momeni, Sebastian Pankalla, Christopher Spies, Petru Bacinschi

http://www.mes.tu-darmstadt.de/
glesner@mes.tu-darmstadt.de

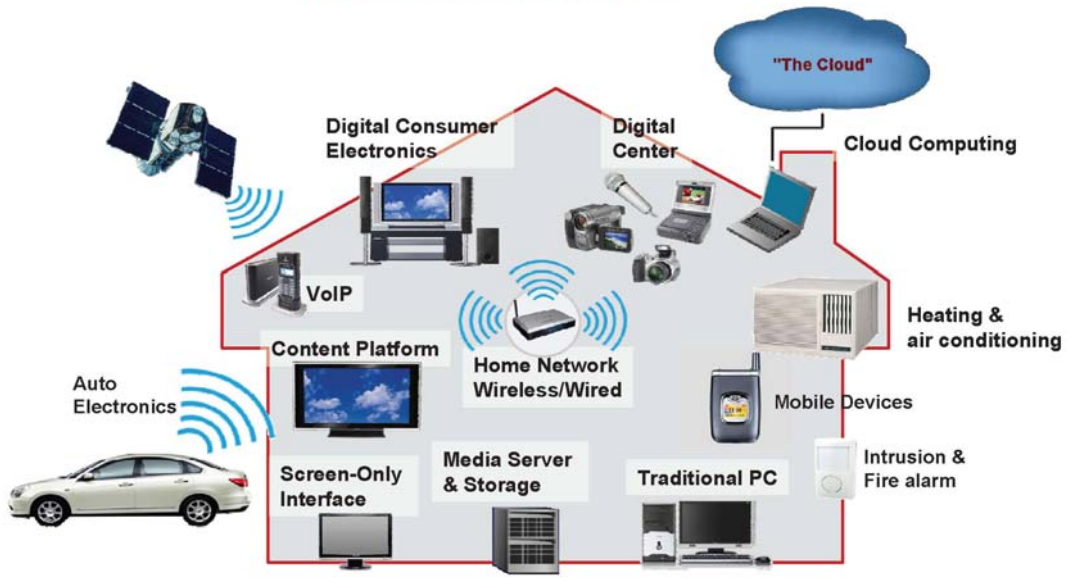
Outline

- **Status of Microelectronics Industry**
- **History: Multi-Project Chip Adventures of Carver Mead and Lynn Conway**
- **TU Darmstadt and the "E.I.S. Zeit"**
- **Status of Worldwide MPC Activities Today**
- **Requirements for Education in a Globalized World**
- **Future Perspectives: More Moore or More than Moore**
- **Summary and Conclusions**

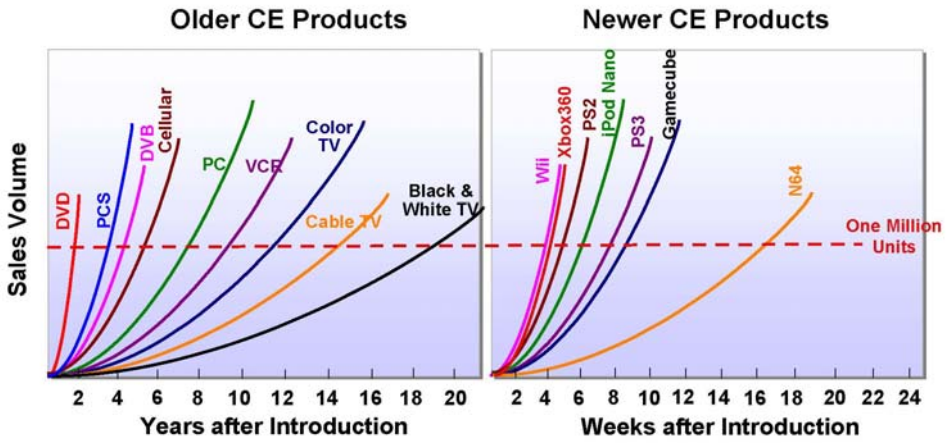
History of Electronics Products



Electronic Products Drive Productivity and Entertainment

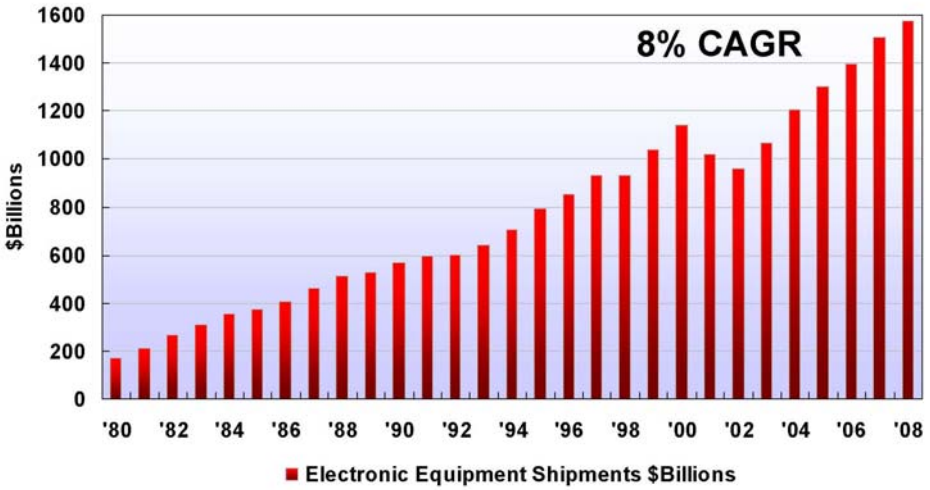


New Products Enjoy Much Faster Ramp



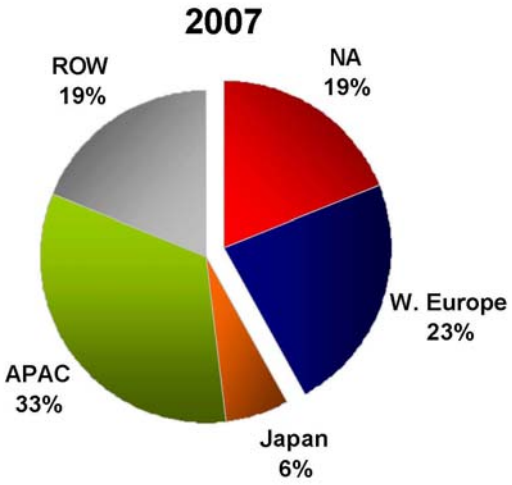
Source: Semico Research Co.

Electronic Product Shipments



Source: IC-Insights; iSuppli; Gartner

Share of Electronics Consumption Growth by Region



Electronics consumption using IMF GDP as proxy

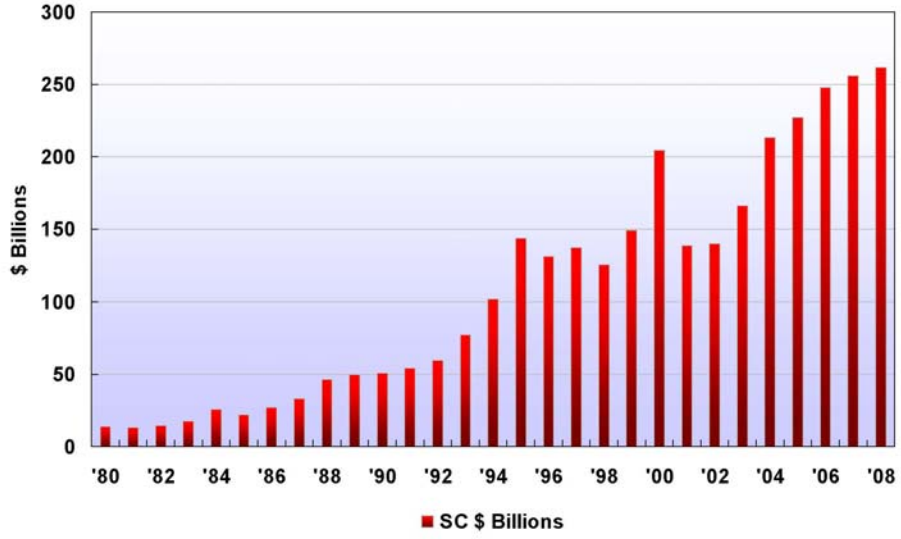
8

Semiconductor Industry's Contribution to WW GDP and Electronics Industry

Source: IMF, iSuppli, WSTS

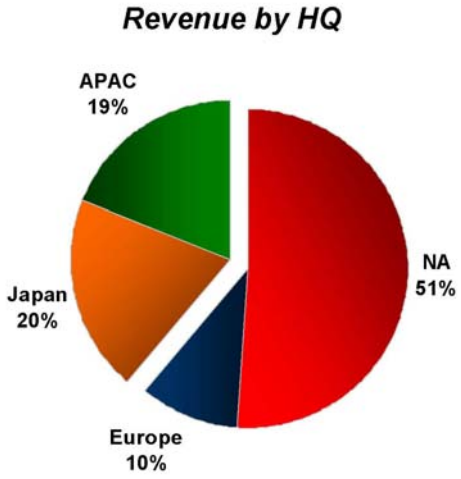
10

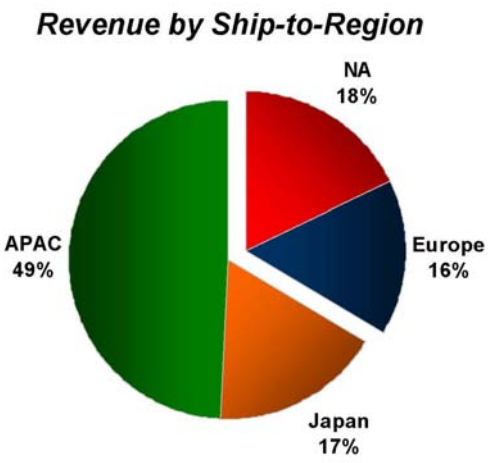
Semiconductor Revenues



11

2007 IC Revenue Breakdown





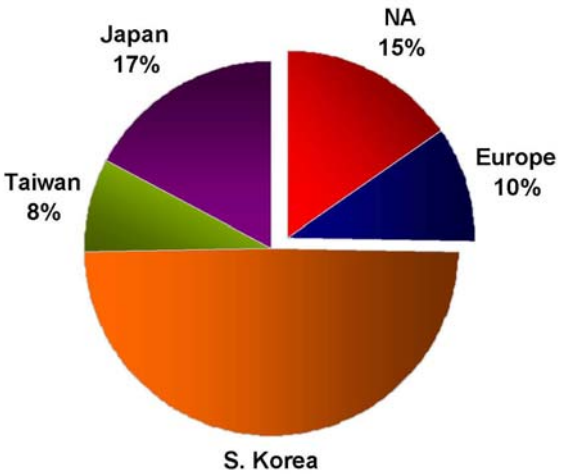
12

Migration of Semiconductor Industry to Asia

- Late 1970's to mid 1980's: Major Japanese systems companies (NEC, Fujitsu, Hitachi, Toshiba, MEI, Sony, etc.) begin to vertically integrate. With their scale, efficiency and quality, by the mid 1980's, about half of all semiconductors by value are being made in Japan.
- Mid 1980's to early 2000's:
  - In the 1980's, Korean companies (Samsung, Hyundai, LG) start to emulate the Japanese model. By the mid-1990's, the Koreans are already in position to challenge the Japanese in the DRAM market and later in FLASH.
  - Also in the 1980's, other Asian countries enter into the industry first with MNC investment and then with establishment of indigenous chip industry with strong government support. New business model begins to emerge with founding of TSMC and UMC in Taiwan in the late 1980's. The same model is emulated by other Asian countries like Singapore and Malaysia.
- After early 2000's: China with its huge electronics manufacturing base and local market begin to invest heavily in the semiconductor industry. Success has been somewhat limited so far.

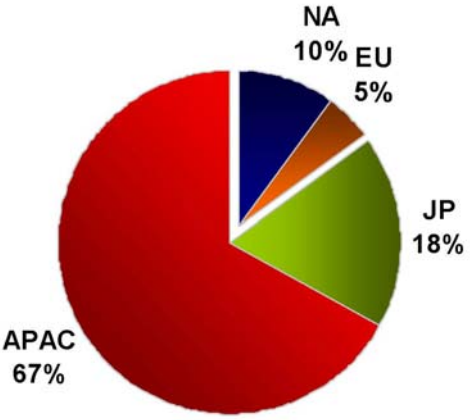
14

Asia dominates Memory…



16

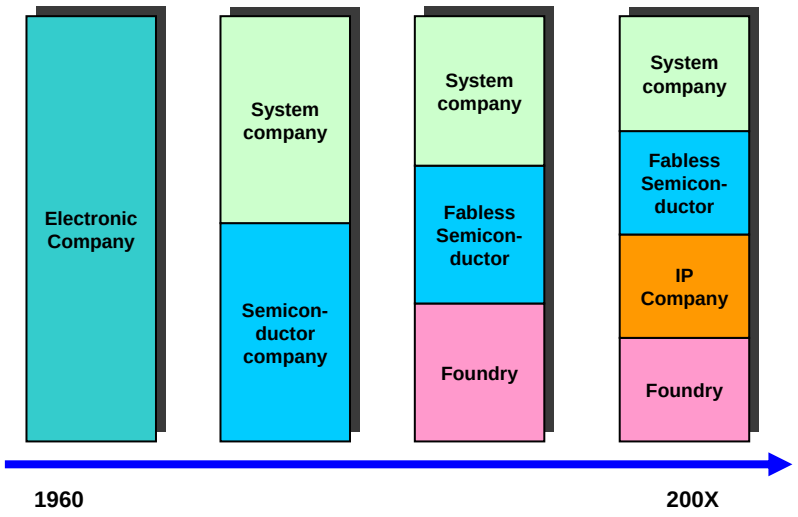
85% of New Wafer Fab Capacity is in Asia



Source: SMA World Fab Watch

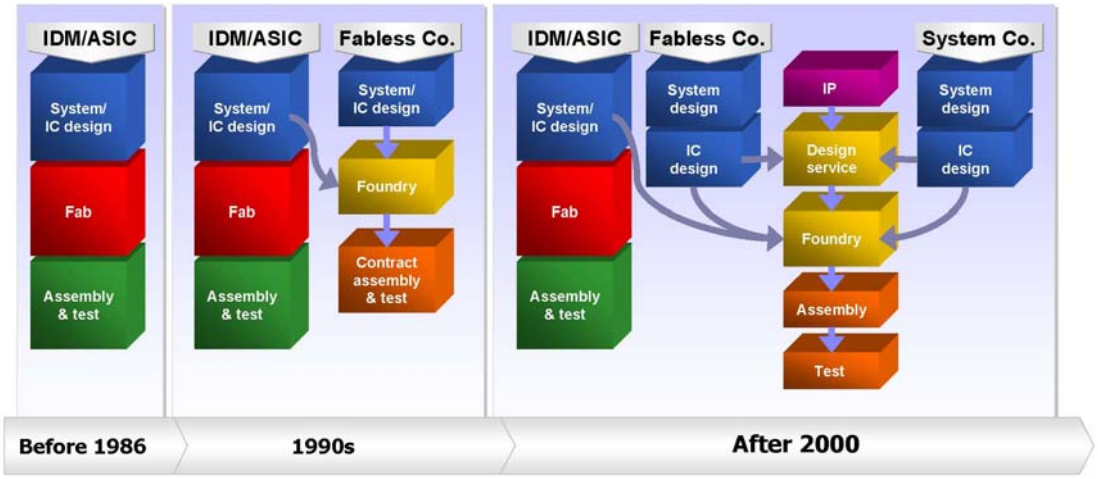
18

Vertical Disintegration



- **New business models driven by extremely high fab costs**
- **Europe's independent fabless companies grow faster than the traditional semiconductor industry (51% vs. 32% in 2000)**

Evolving Foundry Business Model



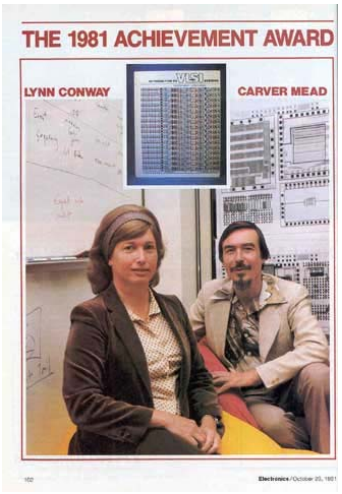
19

Growth of Fabless Segment

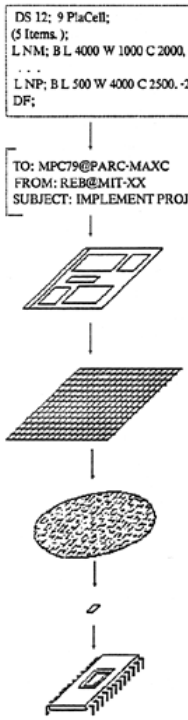
Company	2007 Revenue	
Qualcomm	$5.6B	US
Nvidia	$4.0B	US
Broadcom	$3.8B	US
SanDisk	$3.4B	US
Marvell	$2.8B	US
Mediatek	$2.4B	TWN
Xilinx	$1.8B	US
LSI Logic	$1.8B	US
Altera	$1.3B	US
Novatek	$1.0B	TWN

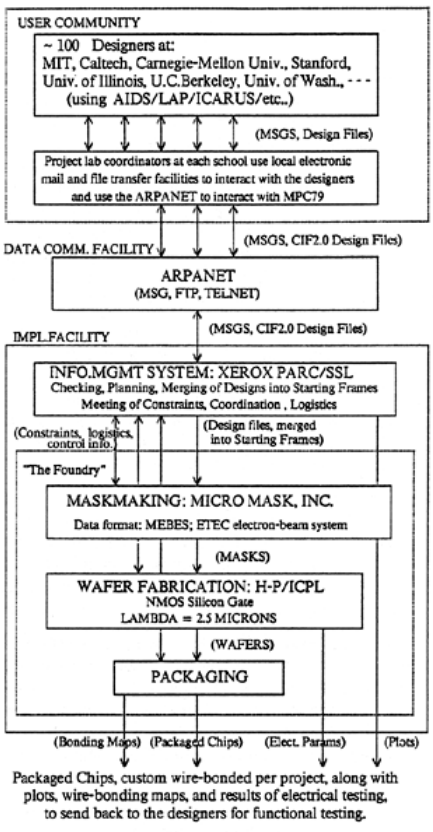
21

1981 Electronics Award for Creating a Common Design Culture for the LSI ERA



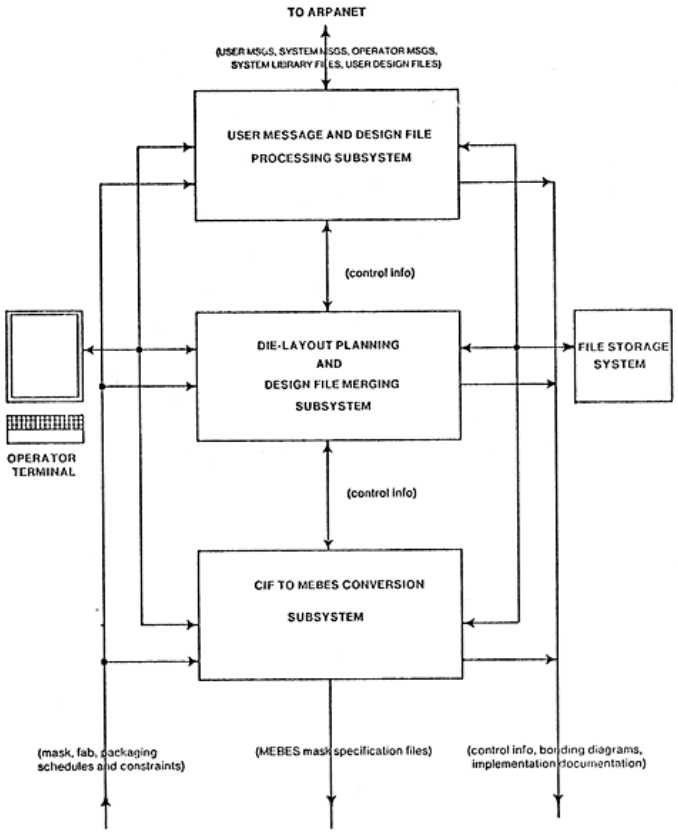
MPC79 Flowchart:





Flowchart of the overall MPC97 activity:

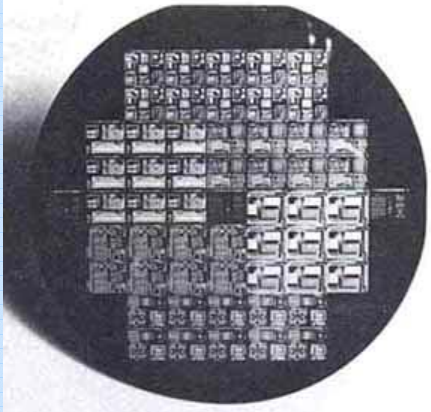
- About a dozen universities joined to participate in MPC79
- Network adventure via ARPANET
- Informational messages distributed for project lab coordination: basic library cells, I/O pads, and PLA cells

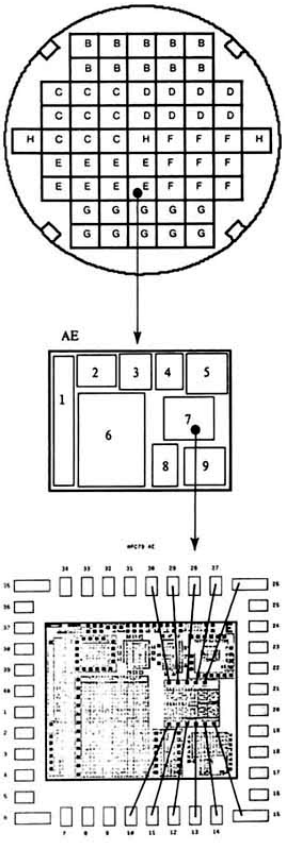


Implementation System to Merge the MPC79 Projects:

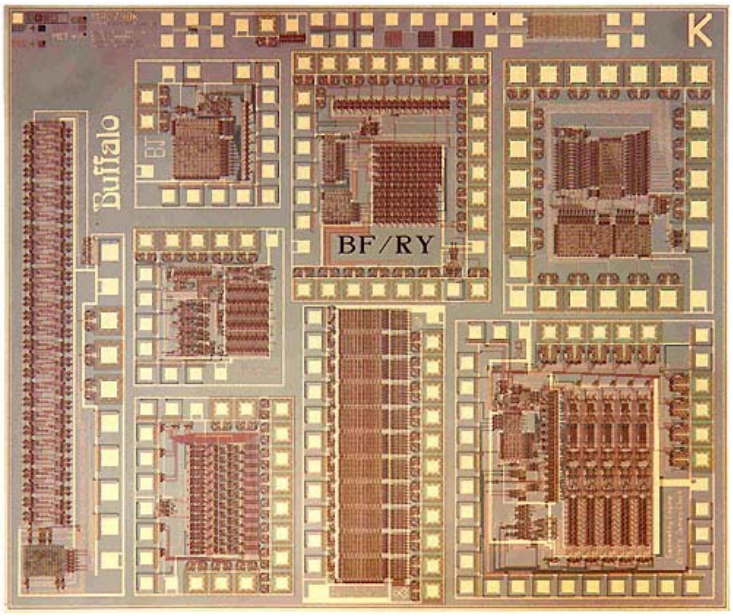
- Alan Bell at XEROX PARC operated the implementation system
- MEBES mask specification: 82 projects from 124 participating designers merged into 12 die-types distributed over two mask sets
- Wafer fabrication by HP ICPL

Details of MPC79 Designs

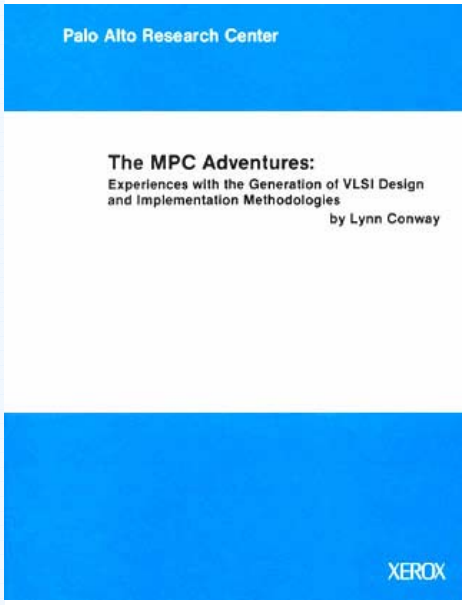




MPC79: First Propotype of the "Geometry Engine" Graphics Processor



First Textbook: Introduction to VLSI Systems (Carver Mead and Lynn Conway)

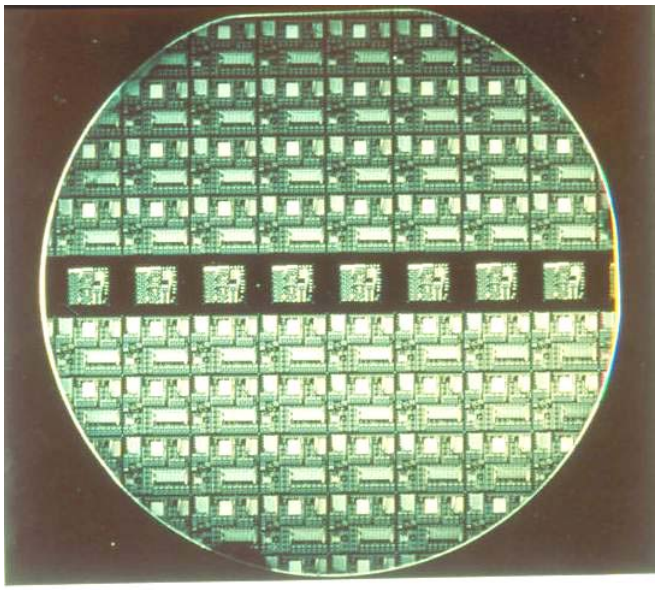


- Origin: 1981 Second Caltech Conference on VLSI
- New form of Internet-based MPC implementation infrastructure
- Rapid-prototyping of many student design projects
- DARPA initiative: 110 universities worldwide
- Book sold in over 70 000 copies
- MPC technology transferred to USC-ISI in 1981: start of MOSIS

E.I.S.

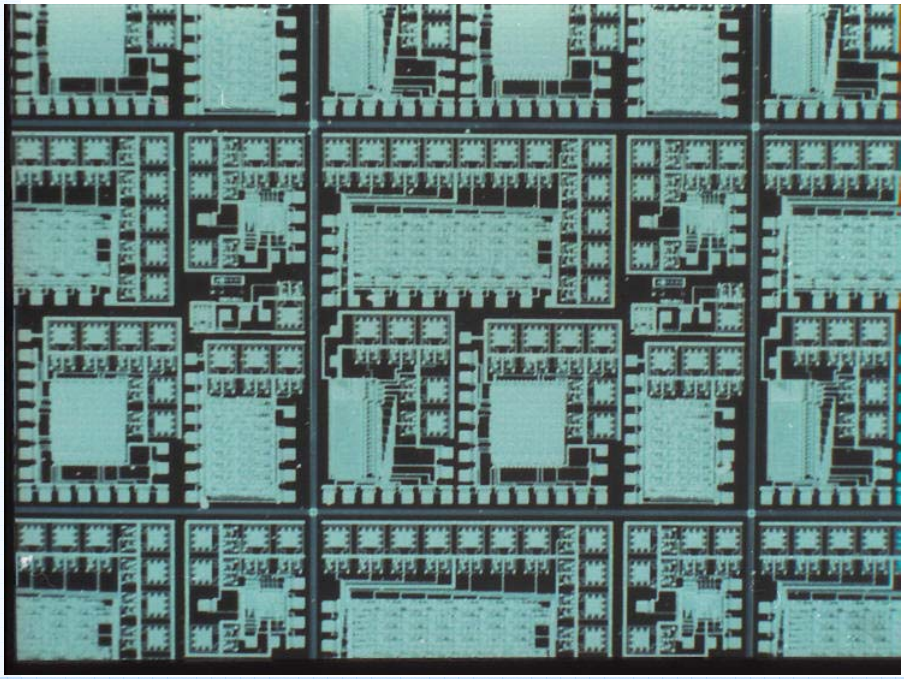
- **E.I.S. = „Entwurf integrierter Schaltungen“**
  - Winter 1978
    - C. Mead & L. Conway VLSI Entwurfs-Kurs am M.I.T.
  - Notwendigkeit von VLSI-Aktivitäten in Deutschland
  - 1983
    - Riesenhuber wird Forschungsminister
    - GMD - Birlinghoven als Projektkoordinator ausgesucht
    - E.I.S.-Projekt auf 3 Jahre und 1,5 Jahre Verlängerung genehmigt
    - 35 Mio DM Fördersumme
    - Beteiligung von 30 Lehrstühlen aus 14 Universitäten
  - 1985
    - Eintritt von Siemens in das E.I.S.-Projekt
    - Zusätzliche 20 Mio DM Förderung

1. Multi-Chip Wafer an der TUD

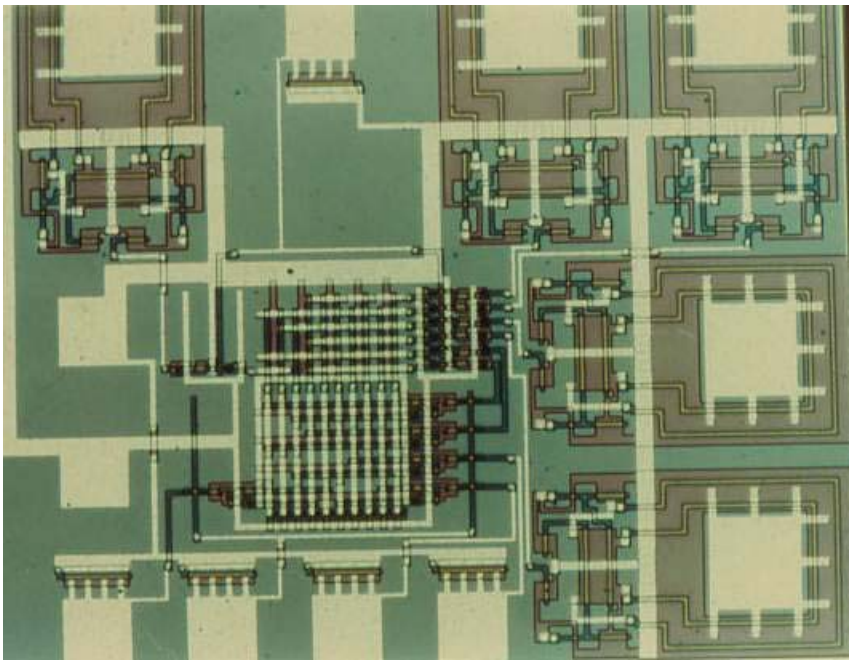


Wafer mit studentischen Entwurfsprojekten

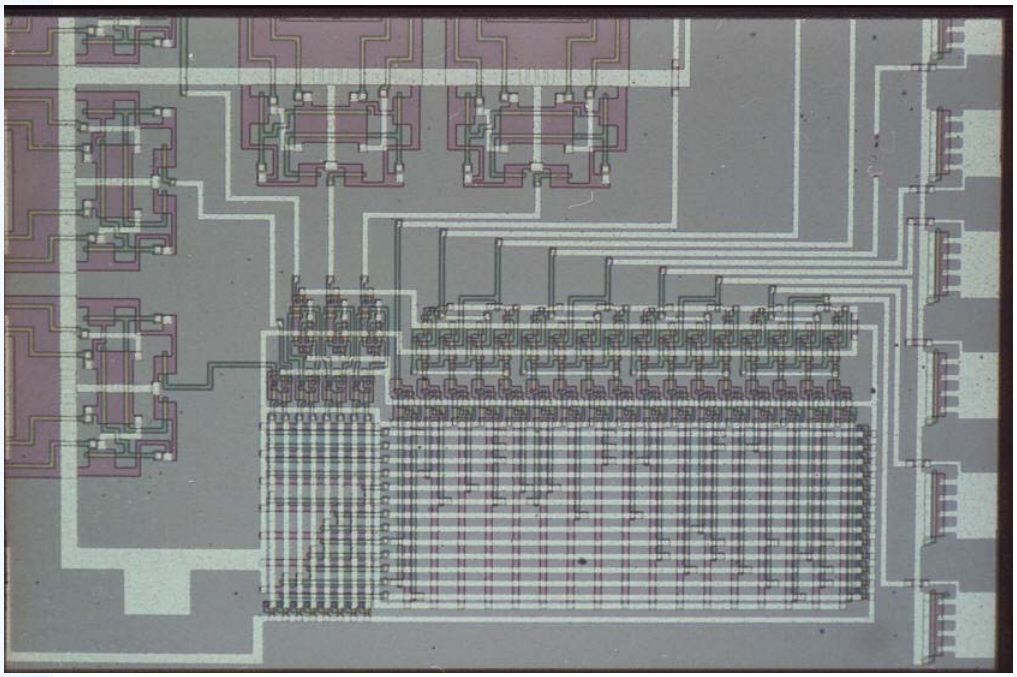
1. Multi-Chip Wafer an der TUD



Mead-Conway Ampelsteuerung



4-Bit PLA Addierer auf 1. DA MPC



Eurochip

- Starttreffen in DA am 23.01.1986

EIS-Zeit an der TH Darmstadt

Tagung: „Ingenieursausbildung in der Mikroelektronik"

(byl). An der TH Darmstadt ist die EIS-Zeit angebrochen. Gemeinsam mit 14 anderen Universitäten ist die Technische Hochschule am Projekt „Entwurf Integrierter Schaltungen" (E.I.S.) beteiligt, das von der Gesellschaft für Mathematik und Datenverarbeitung koordiniert wird.

EIS soll eine Lücke schließen, die zwischen Technik und Wissenschaft klafft und sich ständig vergrößert. Denn einerseits werden die elektronischen Bausteine immer kleiner und leistungsfähiger. Im gleichen Maß werden Entwurf und Test dieser Schaltungen, die teilweise noch mit Bleistift auf Papier skizziert werden, komplizierter. Der Prototyp eines sogenannten VLSI-Chips („Very large scale integration" – ein hochstintegrierter elektronischer Baustein, der mehr als 100 000 Schaltelemente enthält) kann heute schneller hergestellt als konstruiert werden. Das Nadelöhr bei der Herstellung der neuen Computerbausteine sind Aufbau und Test der Schaltungen, die wichtig für die Leistungsfähigkeit der Mikroelektronik- und Informatik-Industrie sind.

Den wissenschaftlichen und personellen Nachholbedarf an den Universitäten soll EIS schließen. Denn nach Ansicht von Experten werden in den neunziger Jahren 100 000 „Chipdesigner" benötigt, die neue Computerschaltungen mit speziell für diesen Zweck ausgerüsteten Grafik-Rechnern entwerfen können. Heute gibt es erst sehr wenige dieser Fachleute.

Bei einer zweitägigen Veranstaltung an der TH Darmstadt zum Thema „Ingenieurausbildung in der Mikroelektronik" diskutierten Wissenschaftler aus England, Irland, Frankreich, Holland, Belgien von der europäischen Kommission in Brüssel und aus der Bundesrepublik Möglichkeiten, die Ausbildung an den Hochschulen zu verbessern und einander anzunähern. Auch wurden Möglichkeiten einer engeren europäischen Zusammenarbeit besprochen.

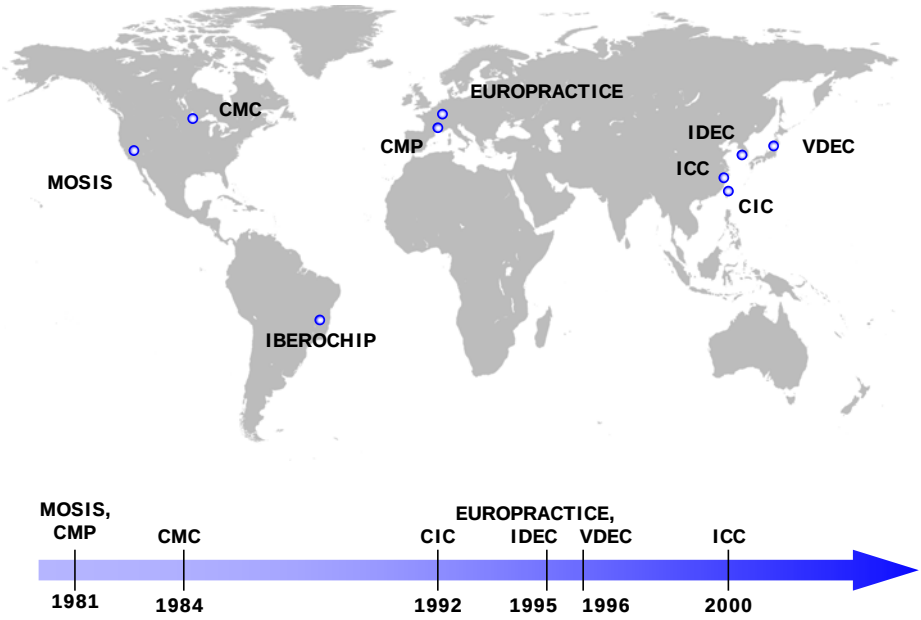
Am zweiten Tag ging es dann um eine Bestandsaufnahme des 1983 begonnenen EIS-Projektes, das völlig neue Dimensionen der studentischen Ausbildung eröffne, wie Diplom-Ingenieur Herwig C. Heckl von der Gesellschaft für Mathematik und Datenverarbeitung erläuterte. Diese Prophezeiung gilt auch in finanzieller Hinsicht. Denn die Ausbildung eines Ingenieurs in der Mikroelektronik soll in einigen Jahren ebenso zu Buche schlagen, wie die eines Mediziners. Ursache für diese inflationäre Entwicklung: Das Studium müsse praxisnah gestaltet werden. „Das Testen eines Mikrochips muß trainiert werden." Dafür seien aber Geräte erforderlich, die erst ab einer Million Mark aufwärts zu haben sind.

Darmstadt, so Heckl, biete sich wegen seiner zentralen Lage und wegen der großen Anstrengungen, die auf diesem Gebiet bisher hier unternommen wurden, als ein Zentrum für die wissenschaftliche Lehre im Rahmen des EIS-Projektes an.

Vor dem Tunnel: Licht einschalten

(DE). In der kommenden Woche werden von Montag (27.) bis einschließlich Freitag (31.) im Wilhelminentunnel Wartungsarbeiten im Versorgungsteil durchgeführt. Für die Dauer dieser Arbeiten müssen die Notstromaggregate abgeschaltet werden, so daß es bei einem eventuellen Stromausfall recht finster in der Röhre wäre. Darum bittet das städtische Tiefbauamt die Autofahrer, in der nächsten Woche auch tagsüber an den Tunneleinfahrten das Abblendlicht einzuschalten.

National Services Around The World (1)



National Services Around The World (2)

Europe	EUROPRACTICE	1995	http://www.europractice.com/ http://www.europractice-ic.com/
Taiwan	CIC	1992	National Chip Implementation Center http://www.cic.org.tw/
Canada	CMC	1984	Canadian Microelectronics Corporation http://www.cmc.ca/
France	CMP	1981	Circuits Multi Project http://cmp.imag.fr/
China	ICC	2000	Shanghai Integrated Circuit Design Center http://www.icc.sh.cn/
Korea	IDEC	1995	IC Design Education Center http://idec.kaist.ac.kr/
USA	MOSIS	1981	MOS Implementation System http://www.mosis.org/
Japan	VDEC	1996	VLSI Design and Education Center http://www.vdec.u-tokyo.ac.jp/

Europe: EUROPRACTICE

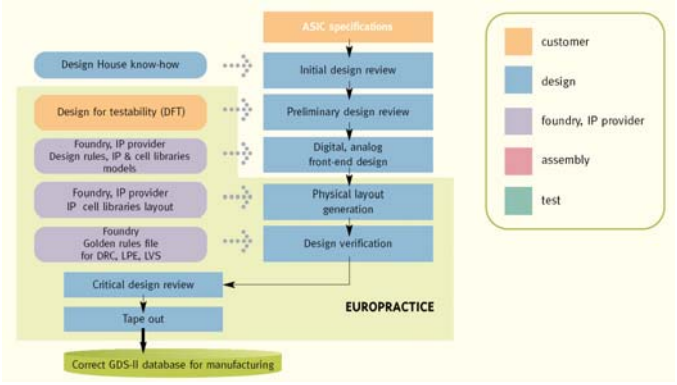
- **EUROPRACTICE was launched by the European Commission in Oct. 1995.**
- **The EUROPRACTICE IC Services are offered by the following consortium:**
  - IMEC—Interuniversitair Micro-Elektronica Centrum (Leuven, Belgium)
    - Coordinator
    - Academic Support
    - Industrial Support
    - MPW & Small Volume
  - STFC—Science & Technology Facilities Council (Swindon, United Kingdom)
    - CAD Support
    - Academic Support
  - Fraunhofer IIS—Fraunhofer-Institut für Integrierte Schaltungen (Erlangen, Germany)
    - Industrial Support
    - MPW & Small Volume

Europe: EUROPRACTICE—ASIC Design

- **Partners:**

Foundry	Library	Assembly	Test
AMIS	Faraday	ASE	ASE
austriamicrosystems	ARM-ARTISAN	HCM	MASER Engineering
IHP			Microtest
TSMC			
UMC			

- **ASIC Design:**



Source: EUROPRACTICE Activity Report 2008

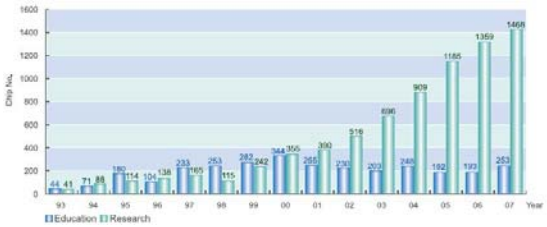
Taiwan: CIC—Profile

- **The National Science Council, based on the resolution of the Fourth National Science and Technology Conference, initiated the Chip Implementation Center (CIC) Project in 1992**

- **CIC focuses its efforts on four main areas:**
  - providing an IC/system design environment
  - providing chip fabrication and measurement services
  - promoting technology for IC/system design and international collaborations
  - improving IC/system design services in southern Taiwan

- **Accomplishments:**
  - in relation to providing IC/system design services, the center integrated **8 well-known IC/System design flows** and provided **2438 hotline services** for academic use.
  - in relation to providing IC fabrication and measurement services, the center helped to produce **1721 chips**, including 253 educational chips and 1468 advanced process chips **in 2007**.
  - in relation to promoting IC/system design technology, the center offered **184 courses to 8441 students**.
  - in relation to improving IC/system design services in southern Taiwan, the center provided **1028 measurement services** to academia and industry in southern Taiwan.

Taiwan: CIC—Chip Implementation Services



Number of chip implementations

IC fabrication processes

Technology	2007	2008	2009	2010	2011
0.35μm 2P4M CMOS (TSMC)					
0.18μm 1P6M CMOS (TSMC)					
0.18μm 1P6M CMOS (UMC)					
0.13μm 1P8M MS CMOS (TSMC)					
0.13μm 1P8M RF CMOS (TSMC)					
90nm CMOS					
65nm CMOS					
0.35μm 3P3M SiGe BiCMOS(TSMC)					
0.18μm SiGe BiCMOS					
0.15μm PHEMT (WIN)					
0.1μm PHEMT/MHEMT					
CMOS Image Sensor					
Backside CMOS MEMS Post Process					
CMOS Bio/MEMS Post Process					
0.35μm CMOS MEMS Post Process					
0.18μm CMOS MEMS Post Process					

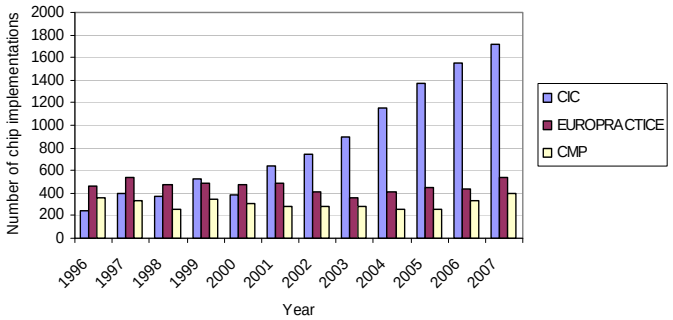
Source: http://www.cic.org.tw/cic_v13/english/aboutus/chip.jsp

France: CMP

- **Circuits Multi Project (CMP) is a broker for various ASIC & MEMS manufacturing technologies for prototyping & low volume production, incl. ICs, MEMS, and MCMs**
- **CMP offers products and services for prototyping in:**

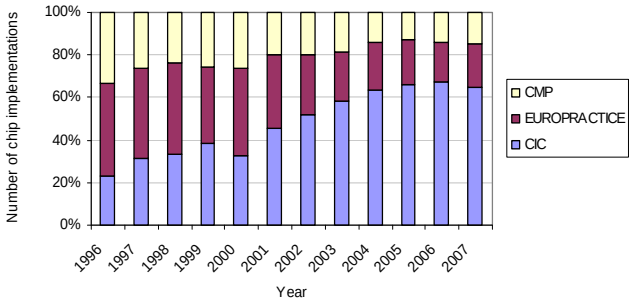
	Foundry/Partners	Technology
Microelectronics	austriamicrosystems	0.35um CMOS, 0.35um SiGe, 0.35um CMOS-Opto, 0.35um CMOS HV, 0.35um CMOS HV EEPROM
	STMicroelectronics	45nm CMOS 7LM, 65nm CMOS 7LM/65nm SOI, 90nm CMOS 7LM, 130nm CMOS 6LM/130nm SOI, 0.25um SiGe:C BiCMOS
	OMMIC	0.2um HEMT GaAs HEMT
MEMS/Microfluidics	CMP/austriamicrosystems	0.35um CMOS bulk micromachining
	MEMSCAP	PolyMUMPs, MetalMUMPs, SOIMUMPs
	CMP/OMMIC	0.2um HEMT bulk micromachining

Number of chip implementations (CMP, CIC, EUROPR.)



Absolute number of chips per service

Contribution of each IC service



Canada: CMC

- **Canadian Microelectronics Corporation (CMC) was launched in 1984 through a university, industry and Natural Sciences and Engineering Research Council of Canada (NSERC) initiative in Kingston, Ontario, Canada**

- **CMC offers products and services for prototyping in:**

	Foundry/Partners	Technology
Microelectronics	STMicroelectronics	45nm CMOS, 65nm CMOS, 90nm CMOS
	IBM	0.13um CMOS
	TSMC	0.18um CMOS, 0.35um CMOS
	AMIS	0.5um CMOS
	DALSA	0.8um High Voltage CMOS
	Gennum	Bipolar Linear Array
MEMS/Microfluidics	Micronit Microfluidics	Sensonit
	MEMSCAP	PolyMUMPs, MetalMUMPs, SOIMUMPs
	DALSA	Bulk Si micromachining
Photonics/Optoelectronics	Canadian Photonics Fabrication Center	III-V prototyping
	Canadian Photonics Fabrication Center	Si-based prototyping

China: ICC

- **Shanghai Integrated Circuit Design Center (ICC) was established in 2000, in 2008 changed its name to Shanghai Integrated Circuit Technology and Industry Promotion Center, directly under the Shanghai Municipal Science and Technology Commission**

- **Partners:**

Foundry	Library	Assembly	Design Service
TSMC	Synopsis	Changjiang	Global Unichip
GSMC	VeriSilicon		
SMIC	ARM-ARTISAN	**Mask**	
Chartered		Dupont	
HJTC			

- **Available technologies:**

Foundry/Partners	Technology
TSMC	0.13um CMOS, 0.18um CMOS, 0.25um CMOS, 0.35um CMOS
GSMC	0.13um CMOS, 0.18um CMOS
SMIC	0.13um CMOS, 0.18um CMOS, 0.35um CMOS HV EEPROM
Chartered	0.13um CMOS, 0.18um CMOS
HJTC	0.18um CMOS

Korea: IDEC

- **IC Design Education Center (IDEC) is located at KAIST (Korean Advanced Institute of Science and Technology) and was established in 1995 with the support and commitment of the Korean government and major semiconductor firms in Korea.**

- **Available technologies:**

Foundry/Partners	Technology
Samsung	0.13um CMOS, 0.18um CMOS
MagnaChip/Hynix	0.18um CMOS, 0.35um CMOS
Dongbu HiTek	0.13um CMOS, 0.18um CMOS
KEC	0.5um CMOS, 0.4um Bipolar

USA: MOSIS

- **MOS Implementation System (MOSIS) plan is the earliest MPW plan in the world and was set up under the support of the American Defense Department (Defense Advanced Research Projects Agency, DARPA) in 1981.**

- **Funding:**
  - DARPA, NSF, SIA, SRC, AMI, and MOSIS itself
  - Ex: 1991 – 85% direct funding by Government Agencies (DARPA, NSF)
  - Since 1994: 100% self-sustaining

- **Available technologies:**

Foundry/Partners	Technology
IBM	45nm/65nm/90nm/0.13um/0.18um/0.25um CMOS, 0.13um/0.18um/0.25um/ 0.35um/0.50um SiGe BiCMOS
TSMC	0.13um/0.18um/0.25um/0.35um CMOS
ON Semiconductor	0.35um HV CMOS, 0.5um CMOS, 0.7um HV CMOS
austriamicrosystems	0.18um/0.35um CMOS, 0.18um/0.35um/0.8um HV CMOS, 0.35um SiGe BiCMOS
Peregrine	0.25um SOS processes (GA and GC), 0.50um SOS processes (FC and FA)

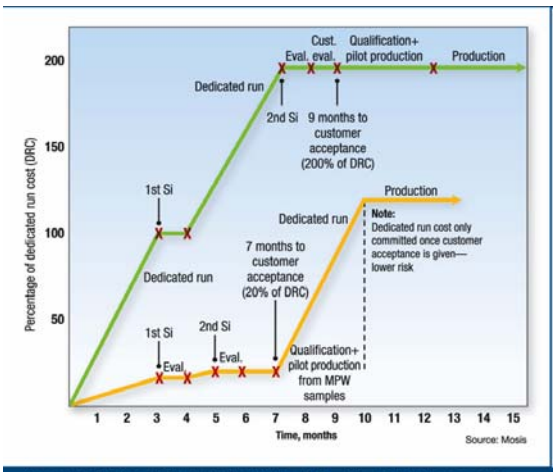
Japan: VDEC

- **VLSI Design and Education Center (VDEC), was established in May 1996, is located at the University of Tokyo and shared by users throughout Japan.**
- **Available technologies:**

Foundry / Partners	Technology
eShuttle	65nm CMOS
ON Semicoductor	1.2um CMOS
Rohm	0.18um CMOS, 0.35um CMOS
Hitachi	0.18um CMOS
NEC	Bipolar 0.8um
OKI	0.15um CMOS SOI
ASPLA	90nm CMOS

Ease of Design Using MPW Runs

- "Using MPW services is not just about cutting the direct costs of development. The major cost saving is in design time [...] reduced cost at which respins can be undertaken, the whole development cycle shrinks." [W. Hansford, 2007]

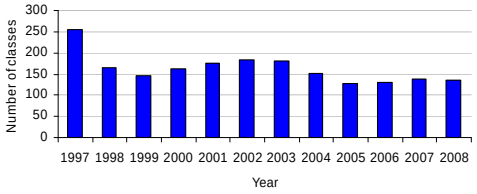


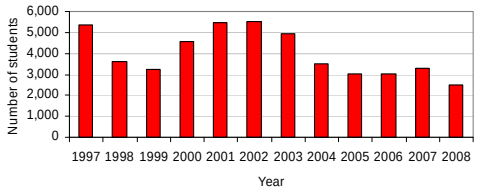
Comparison of doing a multiproject wafer (MPW) run followed by a dedicated run, or going straight into a dedicated wafer run (defined as 'dedicated run cost,' or DRC).

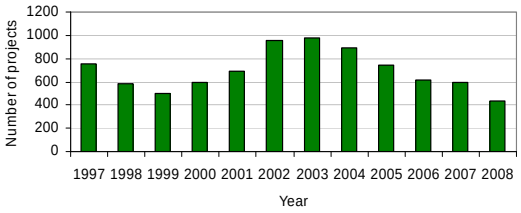
Source: http://www.eetimes.com/news/latest/showArticle.jhtml?articleID=197001431

Interaction between IC Services and Academia (1)

- **The MOSIS Educational Program (MEP) supports educational institutions in the development of instructional and research programs in ICs since 1986:**
  - instructional component for support of scheduled classes in VLSI design
  - research component to support unfunded research conducted by graduate students and faculty
  - Offered at no cost to the universities (with minimal restrictions based on available MOSIS supported technologies and fabrication scheduling)



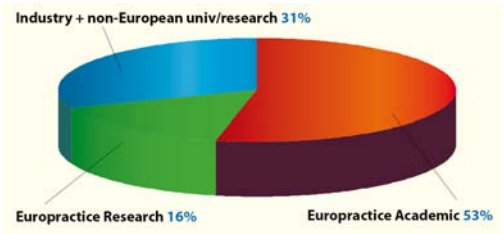




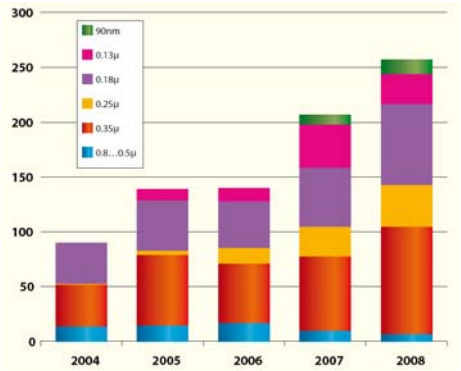
 Source: http://www.mosis.com/products/mep/stats/mep_class_projects.html

Interaction between IC Services and Academia (2)

- **EUROPRACTICE MPW designs in 2008:**



- **mini@sic designs per gatelength in 2008:**
  - introduced in 2003 in order to stimulate universities and research to prototype small designs offering considerably lower minimum prototyping charges for small ASIC designs



 Source: EUROPRACTICE Activity Report 2008

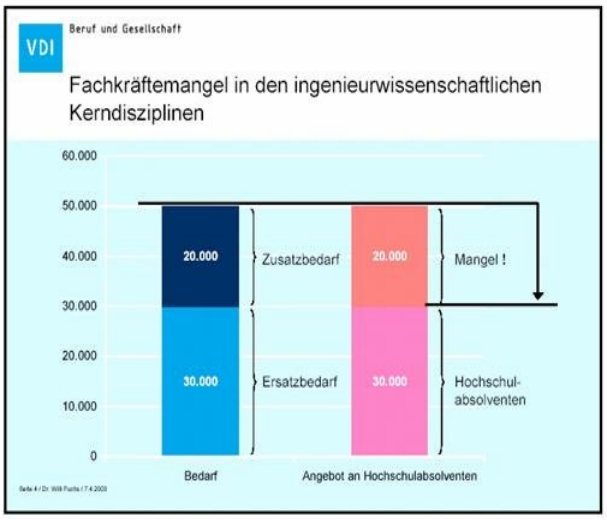
Internationalization in Education (1)

- **Principles:**
  - Internationalization should be flexible, avoiding heavy sunk costs in institutions
  - The framework for internationalization should enable and support faculty-driven activity
  - Internationalization requires an infusion of leadership, resources, and commitment
- **Methods:**
  - Inviting international visitors (“global scholars”)
  - International exchange of graduate students
  - Supplemental funding of international research and educational projects
- **Administration:**
  - Providing faculty leadership and advocacy
  - Ensure high-quality administrative support
  - Reviewing and revising administrative policies
  - Establishing a central place for international activities
  - Improving communication and visibility

Source: Princeton in the World, October 2007

Lack of Talents Lack of Specialized Workforce

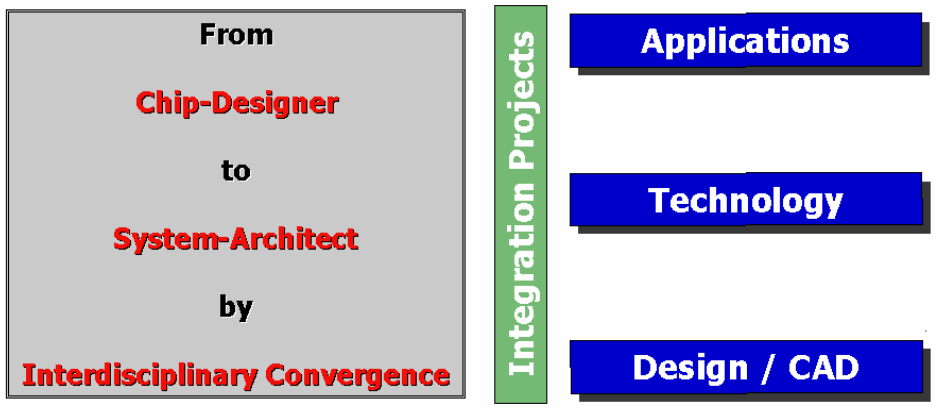
- Companies worldwide are struggling to find the engineering specialists they need to keep up with global competition
- In germany alone, it is estimated that industry could use at least 20,000 engineers more per year
- The number of graduates does not satisfy industry´s demand



Source: VDI

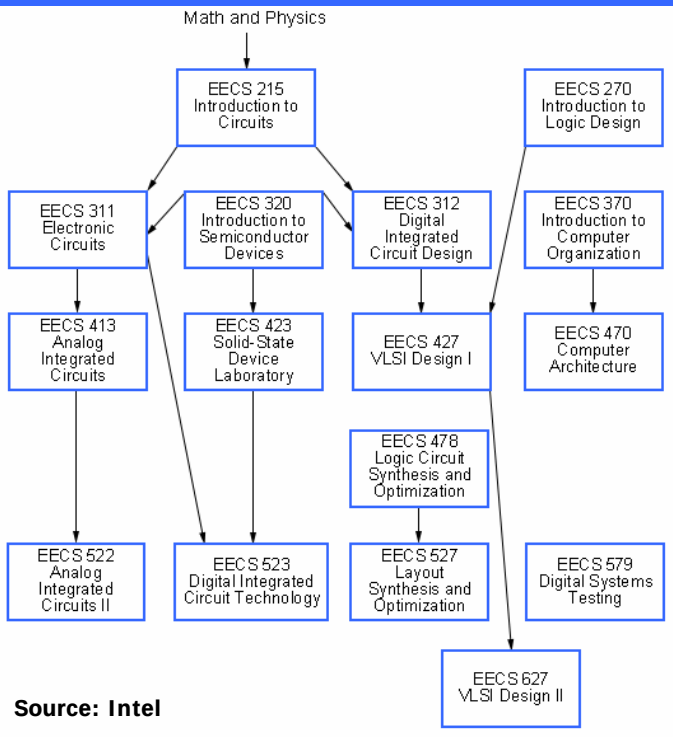
Educational Challenges: New Required Skills

- **New skills** are required for the SoC-Design
- Challenge for the **Universities**



Source: Hugo De Man, IMEC

Example: Intel VLSI Curriculum



- However, the university still needs to employ lecturers and fund the program
- A program like the one shown here already requires quite a few lecturers

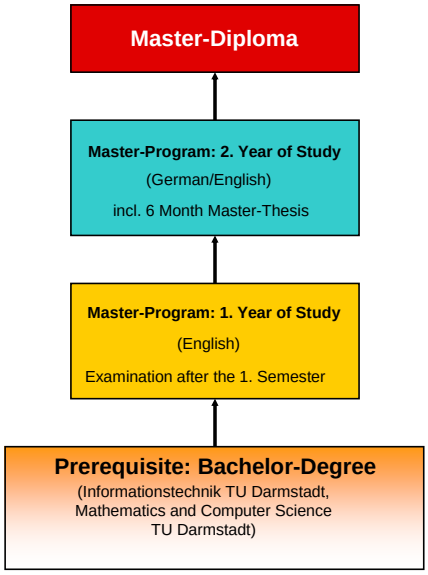
Source: Intel

TUD: ICE Master Program

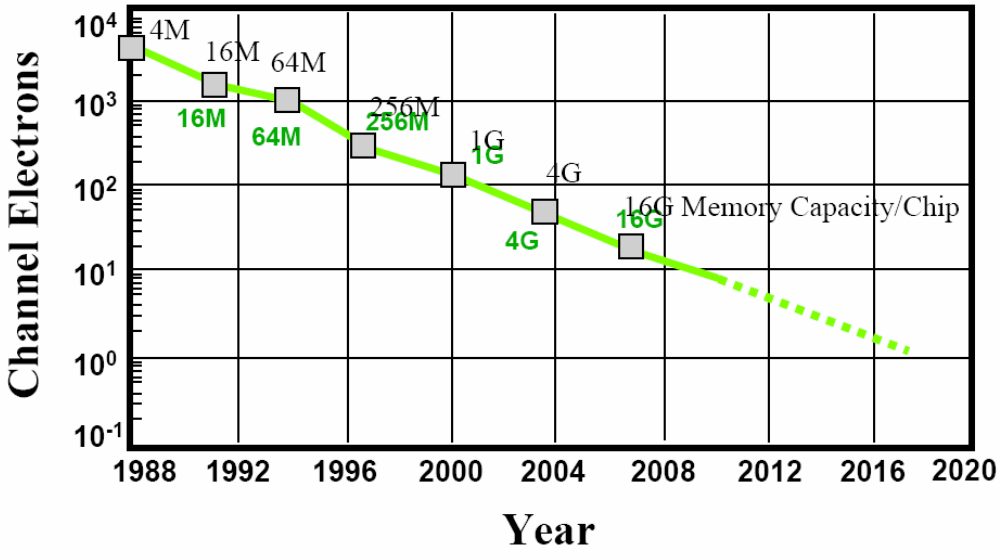
- **International Master Program**

Information and Communication Engineering

- **Start: WS 2001/02**
- **Acceptance Quota:**
  - **~50 Students per year**

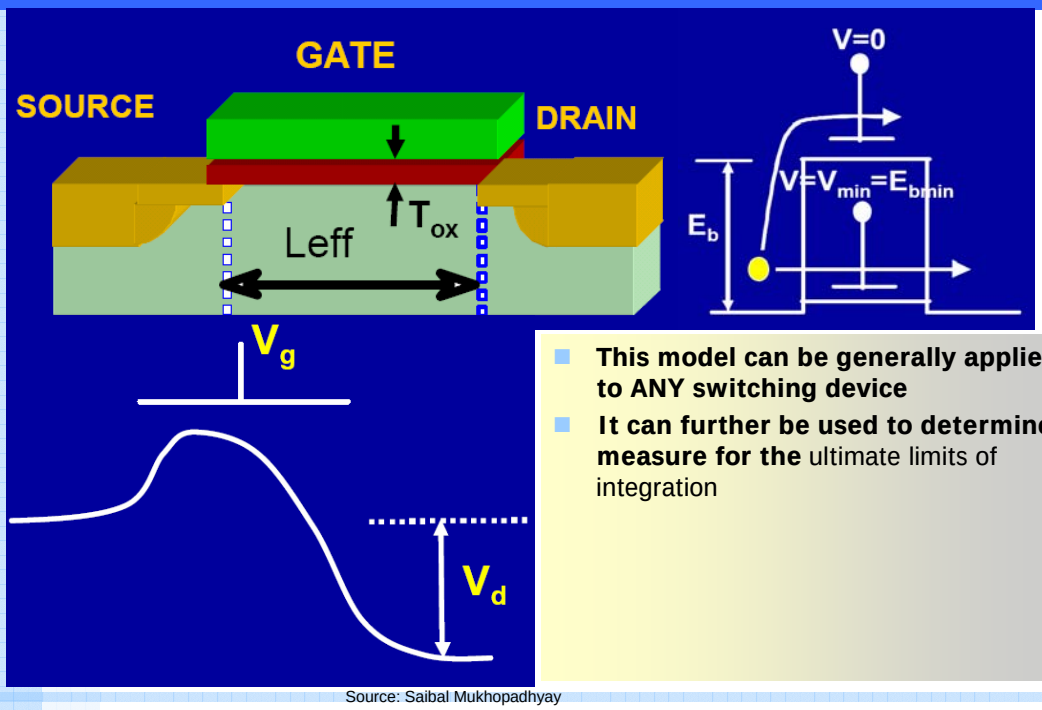


Nano-Technology And Charge Control Issues



Based on: Stephen M. Goodnick, IEEE Nano 2003

Barrier Model of MOSFET



- **This model can be generally applied to ANY switching device**
- **It can further be used to determine a measure for the** ultimate limits of integration

Source: Saibal Mukhopadhyay

Ultimate Limits Of Integration

- **Maximum device density**

$$n_{\max} = \frac{1}{L_{\min}^2} = 4.7 \cdot 10^{13} \frac{\text{devices}}{\text{cm}^2}$$

- **The power dissipation per unit area of this limit technology is given by**

$$P = \frac{n_{\max} \cdot E_{b,\min}}{t_{\min}} = 3.7 \frac{\text{MW}}{\text{cm}^2}$$

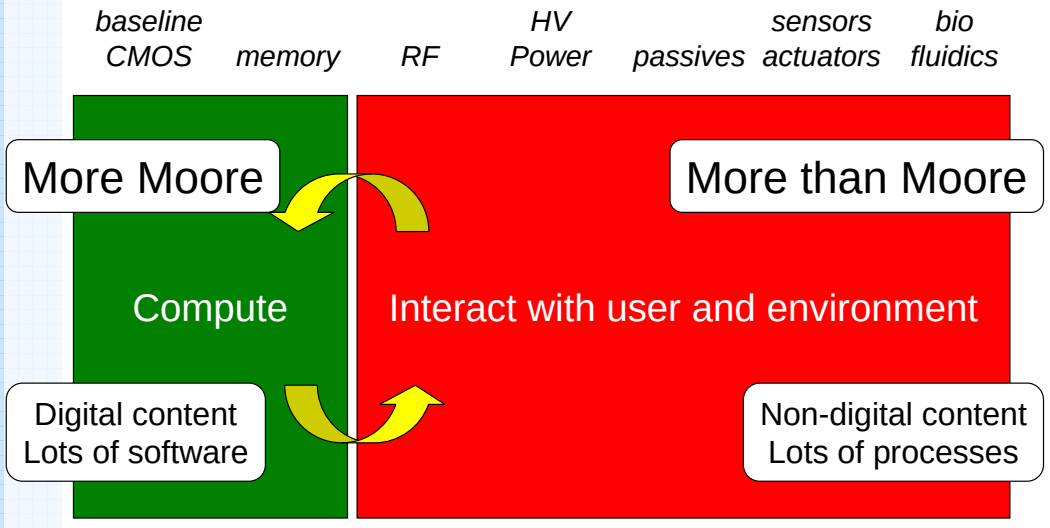
- **For comparison:**
  - Surface of the sun: 6 kW/cm²
  - Forced water cooling: 800 W/cm²
  - ITRS 2016: < 100 W/cm²

Based on: Zhirnov et al.: „Limits to Binary Logic Switch Scaling – A Gedanken Model", Proc. IEEE, Nov. 2003

How Close Will We Get To The Fundamental Limits ?

2018 FET	Fundamental limits
a_{min}: Channel length **7 nm**	$x_{\min} = a = \frac{\hbar}{\sqrt{2mkT\ln 2}} = 1.5nm(300K)$
E_{sw}: Switching energy **3×10^{-18} J**	$E_{sw} > kT\ln 2 = 3 \times 10^{-21}$ J
E_b: S-Ch barrier height **~0.7 eV**	$E_b > kT\ln 2 = 0.02$ eV
Electrons/switching event **~32**	1
Energy/electron **9×10^{-20} J~22 kT**	3×10^{-21}J ~ kT

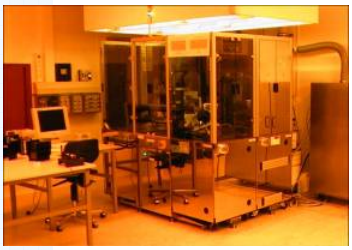
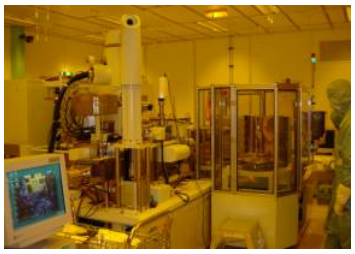
Digital systems compute & interact, leading to two implementation routes



Philips Semiconductors 19 September 2005

More Moore to come

- **On the basis of what we know now we expect Moore's law to continue for at least an other decade, with critical dimensions reaching 22, 15 or may be 10 nm**
- **What we will gain is more density**
  - Lower cost per function
  - More complex functions
- **Speed improvement will be limited**

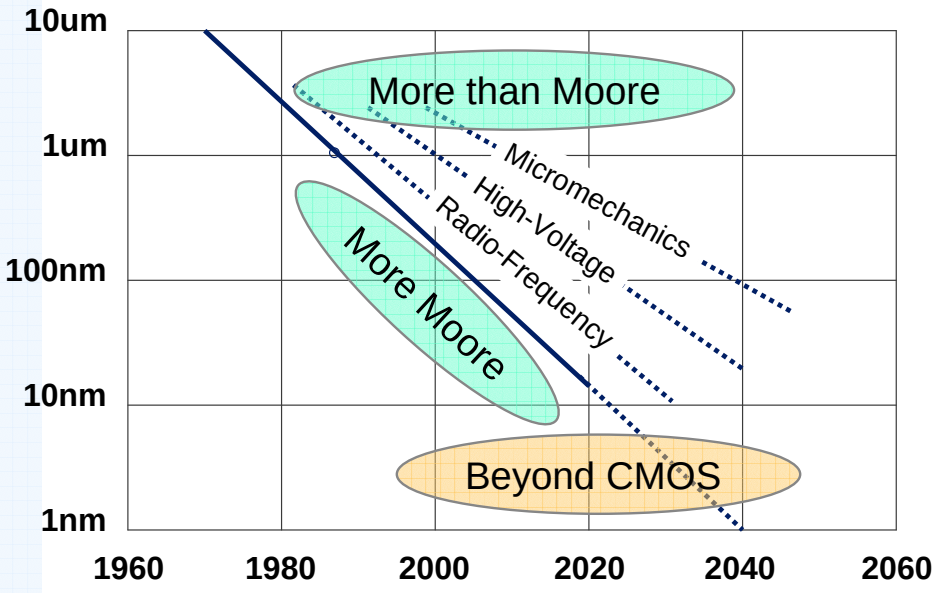




Philips Semiconductors 19 September 2005

'More than Moore'

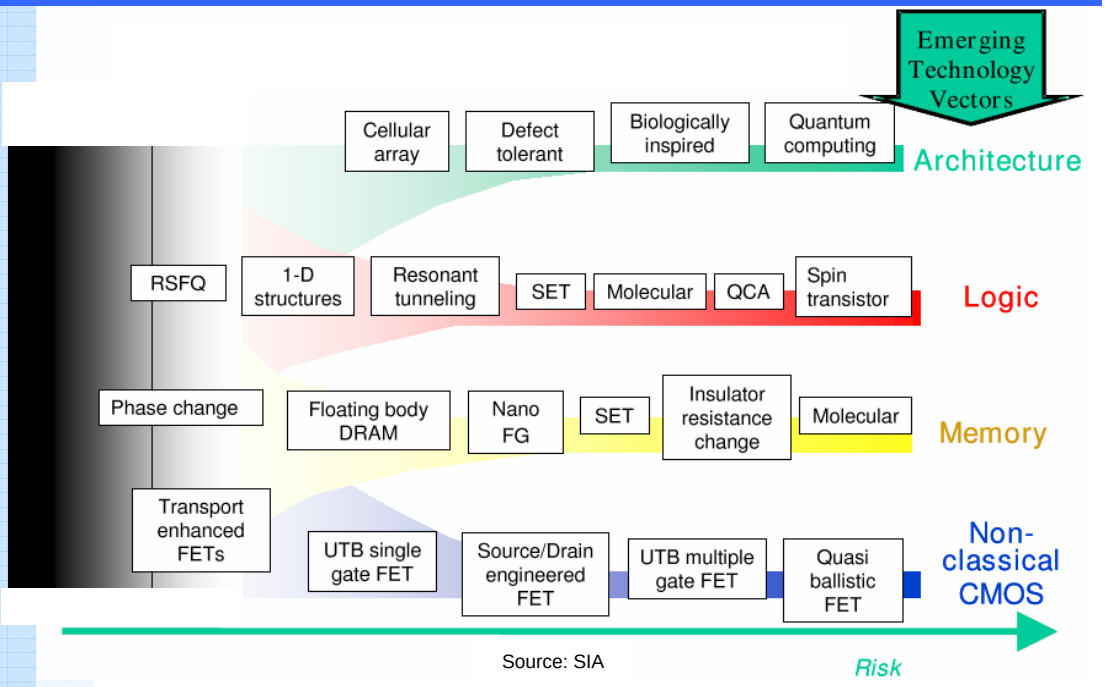
- **Heterogeneous silicon technologies for non-digital functions**
  - Power, RF, passives, sensors and actuators (MEMS), fluidics
- **Building on existing CMOS manufacturing infrastructure**
  - Lithographic scaling lags Moore's law
- **System-in-Package: integrate all functions in one module**
  - Packaging is a functional element / key differentiator
- **Involves cross-disciplinary innovation**
  - Nano-electronics, nano(thermo)mechanics, nano-biology ...
- **Differentiation is in technology mix and system partitioning**

Technology roadmap summary



Philips Semiconductors 19 September 2005

Emerging Technology Sequence



Source: SIA

Emerging Research Logic Devices

Device	 FET	 RSFQ	 1D structures	 Resonant Tunneling Devices	 SET	 Molecular	 QCA	 Spin transistor
Cell Size	100 nm	0.3 µm	100 nm	100 nm	40 nm	Not known	60 nm	100 nm
Density (cm^{-2})	3E9	1E6	3E9	3E9	6E10	1E12	3E10	3E9
Switch Speed	700 GHz	1.2 THz	Not known	1 THz	1 GHz	Not known	30 MHz	700 GHz
Circuit Speed	30 GHz	250–800 GHz	30 GHz	30 GHz	1 GHz	<1 MHz	1 MHz	30 GHz
Switching Energy, J	2×10^{-18}	$>1.4\times10^{-17}$	2×10^{-18}	$>2\times10^{-18}$	$>1.5\times10^{-17}$	1.3×10^{-16}	$>1\times10^{-18}$	2×10^{-18}
Binary Throughput, $GBit/ns/cm^2$	86	0.4	86	86	10	N/A	0.06	86

Source: SIA

Summary: Future Perspectives

- **Microelectronics will run successfully on the Roadmap for the next 20 Years**
- **Aspects for System Integration and Design for Manufacturability will dominate**
- **Adapt to Rapid Changes of Semiconductor Industry Value Chain**
- **New capabilities for SMEs: Improved Design Methods and Architectures for competitive Products on an international Market**
- **Worldwide competition for the transition to Nanoelectronics**
- **Lack of European Scientists and Engineers**

MULTI PROJEKT CHIP GRUPPE

Hochschule Aalen
Prof. Dr. Bartel, (07361) 576-4182
manfred.bartel@htw-aalen.de

Hochschule Albstadt-Sigmaringen
Prof. Dr. Rieger, (07431) 579-124
rieger@hs-albsig.de

Hochschule Esslingen
Prof. Dr. Lindermeir, (0711) 397-4221
walter.lindermeir@hs-esslingen.de

Hochschule Furtwangen
Prof. Dr. Rülling, (07723) 920-2503
rue@hs-furtwangen.de

Hochschule Heilbronn
Prof. Dr. Gessler, (07940) 1306-184
gessler@hs-heilbronn.de

Hochschule Karlsruhe
Prof. Dr. Koblitz, (0721) 925-2238
rudolf.koblitz@hs-karlsruhe.de

Hochschule Konstanz
Prof. Dr. Burmberger, (07531) 206-255
gregor.burmberger@htwg-konstanz.de

Hochschule Mannheim
Prof. Dr. Paul, (0621) 292-6351
g.paul@hs-mannheim.de

Hochschule Offenburg
Prof. Dr. Jansen, (0781) 205-267
d.jansen@fh-offenburg.de

Hochschule Pforzheim
Prof. Dr. Kesel, (07231) 28-6567
frank.kesel@hs-pforzheim.de

Hochschule Ravensburg-Weingarten
Prof. Dr. Ludescher, (0751) 501-9685
ludescher@hs-weingarten.de

Hochschule Reutlingen
Prof. Dr. Kreutzer, (07121) 271-7059
hans.kreutzer@hochschule-reutlingen.de

Hochschule Ulm
Prof. Dipl.-Phys. Forster, (0731) 50-28180
forster@hs-ulm.de

www.mpc.belwue.de