

Bericht über Arbeiten zu Asynchronen Wave Pipelines

Stephan Hermanns und Sorin Alexander Huss
Technische Universität Darmstadt
{hermanns, huss}@iss.tu-darmstadt.de

Zusammenfassung

Das Projekt VLSI-Entwurf mit Asynchronen Wave Pipelines hat zum Ziel, den Entwurf digitaler CMOS-Schaltungen im Giga-Hertz-Bereich unter Verwendung *asynchroner Wave Pipelines* zu erforschen und die Anwendbarkeit dieser neuartigen Architektur für den VLSI-Systementwurf zu untersuchen. Es sollen quantitative Aussagen über den Einfluß von PTV-Schwankungen (Prozeß, Temperatur, Spannung) und Datenabhängigkeiten auf AWP hergeleitet werden. Mathematische Modelle zur Unterstützung eines semi-automatischen Entwurfs und des Vergleichs mit anderen Controller-Architekturen sind zu erarbeiten. Der Entwurf von asynchronen Wave Pipeline Controllern, speziell im Hinblick auf eine Anbindung an AWP-Datenpfade, soll erforscht und Ausdrucksmächtigkeit, Geschwindigkeit, Fläche, Energie und Betriebssicherheit formal mit anderen Ansätzen verglichen werden. Die Systemeinbindung in synchrone und elastische asynchrone Systeme soll untersucht und geeignete Interfaces entworfen werden. Fragestellungen der Testbarkeit sollen bearbeitet und als Ergebnis eine Teststrategie formuliert werden. Die gewonnenen Erkenntnisse sollen in ein prototypisches CAD-Werkzeug einfließen, das die zeitintensiven Schritte des AWP-Entwurfs weitgehend automatisiert. Da das Layout bei Frequenzen über 1 GHz entscheidende Bedeutung erlangt, sind bestehende Verfahren zur Auslegung und Platzierung der Treiber sowie Abschirmungsmaßnahmen gegen Übersprechen und Rauschen zu bewerten und geeignet weiterzuentwickeln.

Einordnung der AWP-Architekturen

Fast alle kommerziellen Produkte sind derzeit synchron getaktet, da diese Entwurfsmethodik deutliche Vorteile aufweist. Der Nachteil des synchronen Entwurfsstils liegt insbesondere bei hohen Taktfrequenzen in den Kosten für Chipfläche, Verlustleistung, Entwurfsaufwand sowie in verringerter Betriebsfrequenz bedingt durch Clock Skew und Schwankungen durch Prozeß, Temperatur und Spannung.

Die Schwierigkeiten beim Entwurf von Taktnetzen ha-

ben dem Asynchronentwurf in den letzten zehn Jahren wieder mehr Auftrieb verschafft [ASYNC-Bib, ASNYC-Conf, IEEE-Proc]. Asynchronentwurf ist aus mehreren Gründen interessant: (1) Die oben beschriebenen, mit der Taktverteilung zusammenhängenden Probleme werden entschärft, indem der globale Takt durch lokales Handshaking ersetzt wird. Das Request-Signal zeigt hier neue Daten an, welche mittels eines Acknowledge-Signals quittiert werden. Daraus resultiert eine elastische und energieeffiziente Arbeitsweise, da Berechnungen nur dann durchgeführt werden, wenn neue Daten und Platz für die Ergebnisse vorhanden sind. Handshaking mit Request/Acknowledge-Signalen ermöglicht den modularen Entwurf im Baukastensystem. (2) Im Gegensatz zu einem globalen, synchronen Taktschema ist die Komplexität des lokalen Handshakings im wesentlichen unabhängig von der Systemgröße. (3) Asynchrone Schaltungen sind attraktiv für die Bereiche Low-Power/Low-Noise [Paver98] und High Performance [Rotem99]. Das Low-Power-Potential resultiert aus der Eigenschaft des Protokolls, daß ohne Daten auch keine Schaltvorgänge stattfinden. Dies steht im Gegensatz zu synchronen Systemen, in denen auch unbenutzte Teile getaktet werden, wenn nicht Clock-Gating angewendet wird. (4) Der Wegfall des synchronen Taktsignals stellt eine erhebliche Einsparung dar, da es das Signal mit der größten Last und Schaltaktivität ist, welches über die Hälfte des dynamischen Verbrauchs beanspruchen kann [Dan97]. Hauptnachteile des asynchronen Entwurfsstils liegen in der Hazardproblematik und der Schwierigkeit, das Handshaking effizient zu implementieren. Wird die Granularität des Handshakings zu fein, können sogar erhöhter Stromverbrauch und reduzierte Geschwindigkeit die Folge sein.

Ein aussichtsreicher Weg zu Pipelining mit hohem Durchsatz bei geringer Latenz ist *Wave Pipelining* [Cotten69, Burlison98]. Eine kombinatorische Logik wird hier mit einer Zykluszeit getaktet, die kürzer als die Latenz der Logik ist. Mehrere Datenwellen sind so in der Logik gleichzeitig aktiv, ohne von Registern getrennt zu sein. Die Gate-Kapazitäten dienen als dynamische Speicherelemente für die Datenwellen. Ebenso wie eine konventionelle synchrone Pipeline ist die synchrone Wave Pipeline von syn-

chron getakteten Speicherelementen begrenzt. Jedoch ist das Abnehmen der Daten am Ausgang schwieriger, und die Phase des Taktsignals an den Registern muß mit speziellen Delay-Elementen angepaßt werden. Im Gegensatz zu einer konventionellen Pipeline ist der Durchsatz der Wave Pipeline nicht durch den längsten Pfad in einer Stufe, sondern durch die *Differenz* zwischen maximaler und minimaler Verzögerung bestimmt. [Burlison98] hat gezeigt, daß synchrone Wave Pipelines nur in bestimmten, disjunkten Frequenzbereichen arbeiten. Damit eine Datenwelle auf einem schnellen Pfad eine vorhergehende Welle auf einem langsamen Pfad nicht zerstört, müssen alle Pfade in der Logik möglichst die gleiche Verzögerung aufweisen, und zwar unter Berücksichtigung von Schwankungen in Temperatur, Versorgungsspannung und Fertigungsprozeß. Die Wahl der Schaltungstechnik für die Logik ist hier ausschlaggebend. In der Literatur wurden hierfür ECL/CML [Wong93], statische CMOS-Logik [Nowka95, Klass94], biased CMOS [Gray94], Domino [Lien95, Mathew] sowie verschiedene Pass-Transistor-Familien [Ghosh, Zhang] vorgeschlagen. Zusammenfassend bieten Wave Pipelines mehr Durchsatz bei geringerer Latenz und Fläche, weniger Registern und weniger Belastung des Taktsignals. Der Preis hierfür ist die Notwendigkeit balancierter Verzögerungen und das kompliziertere Timing. Obwohl schon seit Jahrzehnten bekannt, gilt diese Technik als unsicher und exotisch, ihr Einsatz ist bisher auf SRAMs beschränkt [Heald98].

Pipeline-Architekturen sind von großer Bedeutung für High-Performance-Systeme und bilden ab 1998 den Schwerpunkt der Arbeiten, die zur Entwicklung der asynchronen Wave Pipelines, AWP's führten und in [Hauck98b, Hauck99a, Hauck99b, Hauck99c, Hauck99d, Hauck00a, Hauck00b] dokumentiert sind. Die zugrundeliegende, neue Idee ist die Kombination der beschriebenen inelastischen asynchronen Arbeitsweise mit Wave Pipelining. Abbildung 1 zeigt ihre generische Anordnung. Daten sind da-

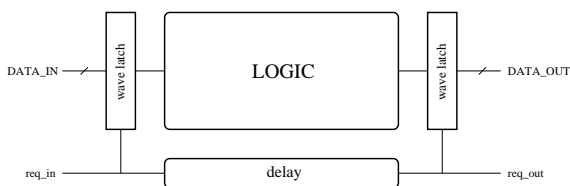


Abbildung 1. Generische AWP-Architektur

bei mit einer Protokollinformation (Pegel oder Flanke) auf der Request-Leitung assoziiert. Daten und assoziierter Request laufen kohärent durch die Logik bzw. das Delay-Element. Wenn die Delay-Variation zwischen den Pulsen einer Daten-/Request-Welle ihren maximal zulässigen Wert erreicht hat, werden die Datenwellen im Wave Latch wieder neu ausgerichtet, indem auf langsame Bits gewartet wird. Die Struktur in Abbildung 1 ist somit beliebig konkate-

nierbar. Nicht nur in der Logik befinden sich gleichzeitig mehrere Datenwellen, auch im Delay-Element sind mehrere Requests vorhanden: Logik *und* Delay-Element funktionieren nach dem Wave Pipelining-Prinzip. Die kombinatorische Logik muß jedoch für den Wave Pipeline-Betrieb ausbalanciert werden. Für die Wave Latches existieren je nach Schaltungstechnik und 2- oder 4-Phasenprotokoll verschiedene Varianten. Die genaue Modellierung des längsten Pfades im Delay-Element in Gegenwart von Prozeß-, Temperatur- und Spannungsschwankungen sowie datenabhängiger Laufzeiten ist kritisch für die Stabilität und den Durchsatz der AWP.

Als Haupthindernis für höheren Durchsatz wurde das Acknowledge-Signal identifiziert. Dieses ist notwendig für Elastizität, Kennzeichen eines voll implementierten Handshakings. Elastizität vereinfacht zwar den modularen Systemaufbau, bedingt aber kompliziertere Speicherelemente und resultiert in geringerer Leistung. Dieses Signal ist jedoch entbehrlich, wenn man auf Elastizität verzichten kann. Die in eigenen Vorarbeiten vorgeschlagenen Pipelines verwenden nur den Request, der auch als lokales, durchgeschleiftes Taktsignal angesehen werden kann. Im Gegensatz zu synchronem Wave Pipelining arbeiten AWP's auf einem durchgehenden, nach oben begrenzten Frequenzband. Die Arbeitsweise ist immer noch asynchron, jedoch inelastisch, d. h. anfallende Daten müssen konsumiert werden, bevor sie überschrieben werden. Das Ziel ist, die Schaltungen so einfach und damit so schnell wie möglich zu gestalten und dabei von den Vorteilen des Asynchronbetriebs zu profitieren: Keine globale Taktverteilung, inhärenter Low-Power-Modus und modularer Entwurf. Die Abwesenheit der Elastizität läßt den Systementwurf mit AWP's wieder mehr synchron erscheinen.

Arbeiten im VIVA Programm

Quantitative Analyse der AWP

Von entscheidender Bedeutung für die praktische Einsetzbarkeit der AWP ist der Einfluß von Datenabhängigkeiten und PTV-Schwankungen auf die Delay-Variationen im Datenpfad und auf die Kohärenz der Signale im Datenpfad und auf der Request-Leitung. Der Einsatz von Self-Resetting CMOS (SRCMOS) eliminiert die Datenabhängigkeiten weitgehend, da der 1-0-Übergang am Ausgang durch den Precharge bewirkt wird, der von den Eingangsdaten unabhängig ist. Der 0-1-Übergang am Ausgang hängt vom NMOS-Teil ab und kann so gestaltet werden, daß jeweils genau ein Pfad zu Masse existiert. Die henerische Struktur eines SRCMOS-Gatters zeigt Bild 2. Alle Gatter an derselben sequentiellen Position in der Logik besitzen NMOS-Zweige mit der gleichen Anzahl von in Serie geschalteten NMOS-Transistoren, so daß an den

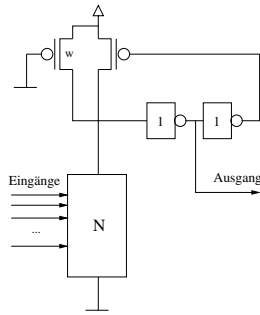


Abbildung 2. Generische Struktur eines Self-Resetting CMOS Gatters

Eingängen eines Gatters Impulse verschiedener Ausgänge simultan eintreffen. Durch die gepulste Arbeitsweise weist die Reihenschaltung in NMOS-Teil selbst eine nahezu konstante Verzögerung auf. Geringe Datenabhängigkeiten ergeben sich, wenn durch Charge Sharing vorhergehender Schaltvorgänge das Precharge-Potential absinkt und somit den Spannungshub für den NMOS-Teil für folgende 0-1-Übergänge absenkt. Mehr Datenabhängigkeit entsteht, wenn zur Reduzierung der Latenz Parallel/ODER-Strukturen im NMOS-Teil zugelassen werden, da dann die Anzahl der schaltenden Pfade nicht a priori bekannt ist. Diese Delay-Variationen lassen sich gut beherrschen. Einen größeren Einfluß hat hier die Anpassung der Ausgangstreiber an den Fanout. Irregulärer Fanout führt ohne Anpassungen der Ausgangsstufen zu Delay-Variationen, die nicht zu vernachlässigen sind.

Zur quantitativen Beschreibung dieser Effekte wird zur Zeit unter Verwendung von MOS-Device-Parametern, Transistor-Modellgleichungen sowie der Self-Resetting-Struktur der Gatter ein mathematisches Modell erstellt, das Aussagen zur Delay-Variation erlaubt. Da wir hier primär an Komponenten mit nur lokalen Verbindungen wie z. B. Arithmetikblöcken interessiert sind, genügt eine einfache RC-Modellierung der Interconnects ohne Induktivitäten und Transmission-Line-Effekte. Die Entwicklung des Modells ist Teil der aktuell laufenden Arbeiten. Das Transistor-Sizing-Problem für AWP's mit SRCMOS ist nicht trivial aufgrund der gegenseitigen Abhängigkeiten von NMOS-Teil, Precharge/Feedback-Teil und Fanout. Die gefundenen Gleichungen sollen als ein Problem zur Minimierung der Delay-Variation unter Einhaltung einer Latenzschränke zu formulieren. Es soll mittels Optimierungsverfahren angegangen werden. Im in der Entwicklung befindlichen Modell finden sich die Parameter für die Temperatur und die Betriebsspannung aus den Transistorgleichungen wieder. Unter der Annahme, daß keine Prozeßschwankungen vorliegen, können mit diesem Modell schon Aussagen über den Toleranzbereich von Temperatur und Spannung

gemacht werden. Die Auswirkungen der Prozeßschwankungen auf Kanallänge und -breite, Threshold-Spannung, NMOS/PMOS-Tracking, Gate-Doping, Interconnect-Breite und -dicke sind statistischer Natur und analytisch aufwendig zu erfassen, da im Modell bisher konstante Parameter wie MOSFET-W/L durch Wahrscheinlichkeitsverteilungen zu ersetzen sind. Die Parameter driften bisher auf einem Chip gewöhnlich in dieselbe Richtung – davon kann bei zukünftigen Chipflächen von 10 cm^2 nicht mehr ausgegangen werden. Von grundsätzlichem Interesse ist hier die statistische Simulation mittels Monte-Carlo-Verfahren unter Verwendung der am Prozeß gemessenen Parameter für die Verteilungsfunktionen. Im Gegensatz zu worst-case-Simulation liefert ein solches Vorgehen realistischere Ergebnisse auf Kosten höheren Rechenaufwandes und erlaubt es, den Schaltkreis hinsichtlich Delay-Variationen besser zu optimieren. Zusammenfassend sollen also die Delay-Variationen — bedingt durch Datenabhängigkeiten, Temperatur und Spannung — analytisch modelliert und optimiert werden. Prozeßschwankungen sollen durch Monte-Carlo-Simulation quantifiziert werden. Praktische Ergebnisse über die Empfindlichkeit eines SRCMOS-Gatters auf PTV-Variationen zeigten sich am Beispiel der Schaltungen zur Puls-Pegel-Wandlung [HeHu01, Herm01]. Eines der vorgeschlagenen Gatter zur Pulserzeugung, das ebenfalls auf SRCMOS basiert, wurde auf seine Empfindlichkeit gegenüber PTV-Schwankungen anhand von drei Größen untersucht. (1) Die Zeit bis ein Puls am Ausgang als Antwort auf eine Flanke an einem Steuereingang erscheint, liegt zwischen 154 ps bei $\langle -5^\circ\text{C}, 3.63\text{V} \rangle$ und 237 ps bei $\langle 85^\circ\text{C}, 2.97\text{V} \rangle$. (2) Durch Prozeßvariationen schwankt diese Antwortzeit unter Nominalbedingungen $\langle 27^\circ\text{C}, 3.30\text{V} \rangle$ zwischen 134 ps und 241 ps und liegt damit in der gleichen Größenordnung der VT-Delay-Variationen. (3) Die Pulsdauer variiert unter Nominalbedingungen zwischen 270 ps und 338 ps. Die betrachtete Stichprobe bestand jeweils aus 500 Exemplaren. Die Schwierigkeiten, kurze Pulse einer Breite von 300ps und weniger über statische CMOS-Logik zu propagieren, wurden wieder bestätigt. In Kürze beginnen Arbeiten zur Steuerung der Ausgangspulsbreite eines SRCMOS-Gatters ohne die Latenz zu erhöhen. Weiter soll festgestellt werden, wie sich Inter- und Intra-Die-Variationen in skalierten Technologien auswirken. Es wird davon ausgegangen, daß diese Fragestellungen nach Ablauf der ersten zwei Jahre geklärt und gelöst sind.

Aus der Tatsache, daß der Durchsatz der AWP hauptsächlich von Verzögerungsdifferenzen anstatt absoluter Verzögerungen bestimmt wird, ergibt sich die Frage, wie sich der Durchsatz bei Absenkung der Betriebsspannung verhält. Bei konventionellen Pipelines sinkt der Durchsatz mit sinkendem V_{dd} linear ab, ebenso wie die Latenz linear in V_{dd} zunimmt. Das ist bei AWP's nicht der Fall, da die Differenz theoretisch verschwindet, wenn die Latenzen in glei-

cher Weise zunehmen. Tatsächlich wurde bei Simulationen erkennbar, daß die Delay-Variation in AWP's mit SRCMOS zumindest nicht das lineare Verhalten der Latenz aufweist und das die Absenkung von V_{dd} bis zu einer bestimmten Grenze den Durchsatz wenig beeinflusst. Die Verringerung von V_{dd} ist ein einfaches und wirksames Mittel zur Verringerung der aufgenommenen Energie, da diese von der Spannung im Quadrat abhängt. Untersuchungen sind zur Klärung der Frage vorgesehen, ob dieser Effekt tatsächlich an realen Chips meßbar ist oder von anderen Effekten überlagert wird. Zusammen mit der Eigenschaft, daß AWP's die Taktinfrastruktur und die Anzahl der Register weitgehend reduziert und durch die Ausbalancierung der Verzögerungen keine Glitches zuläßt, könnte sich als Ergebnis dieser Untersuchungen ergeben, daß AWP's nicht nur höheren Durchsatz als konventionelle Pipelines bieten, sondern dabei auch wesentlich energieeffizienter sind. Dies wird zukünftig von Bedeutung sein, wenn die Stromverteilung und durch Schaltvorgänge bedingtes di/dt -Rauschen auf V_{dd} und V_{ss} zentrale Probleme werden.

Systementwurf mit AWP's

Die Vorarbeiten im Institut beschäftigten sich hauptsächlich mit der Schaltungstechnik für Feed-forward-Datenpfade. Vollständige VLSI-Systeme besitzen zusätzlich Register zur Speicherung von Daten, FSM's zur Steuerung sowie Interfaces und Speichermodule.

Die AWP kommt ohne Pipeline-Register aus, jedoch werden Register an verschiedenen anderen Stellen benötigt, z. B. als Register-File für die ALU in einem Prozessor, als Schieberegister, Seriell/Parallel-Wandler oder als Barrel-Shifter. Für Register-Files wurden Interfaces entworfen, die den gepulsten Datenpfad effizient an Speicherzellen ankopeln, die Signalpegel speichern [HeHu01, Herm01]. Dazu wurde eine Komponente entworfen, die die gepulste Ausgabe in Pegel wandelt und synchron mit dem Request ausgibt. Das Einlesen der Pulse in Single-Rail-Kodierung geschieht passiv und ist weitgehend immun gegen Daten-Request-Skew. In diesem Zusammenhang wurde die Möglichkeit untersucht, Metastabilität beim synchronen Einlesen der asynchronen Ausgabe einer AWP durch zusätzliche Delay-Elemente zu vermeiden. Die optimale Verzögerung zur Vermeidung von Metastabilität existiert nach mit Gleichung 1, wenn $T - t_{setup} - t_{hold} - \Delta t_{var,P}^{max} - |\Delta t_{var,P}^{min}| - \Delta t_{var,P,delay.prevMS}^{max} - |\Delta t_{var,P,delay.prevMS}^{min}| > 0$ ist. Andernfalls sind die PTV-Variationen zu groß, um eine Verzögerung finden zu können. Die durch die AWP realisierte Operation braucht dabei M Taktzyklen, die $\Delta t_{var,P}$ beschreiben die PTV-Variationen des Datenpfades, die $\Delta t_{var,P,delay.prevMS}$ die des zusätzlichen Verzögerungselementes, P ist die Wahrscheinlichkeit, mit der eine gefertigte Schaltung die Δt einhält.

Der Einsatz eines Synchronisierers zum synchronen Einlesen scheidet aus, da die Wahrscheinlichkeit eines Synchronisationsfehlers auf beliebig kleine positive Werte gedrückt werden kann, die Richtigkeit der Entscheidung des Synchronisierers dennoch nicht sicher ist.

$$t_{delay,prevMS} = \frac{T - t_{setup} - t_{hold} - \frac{\Delta t_{var,P}^{max} - |\Delta t_{var,P}^{min}|}{2}}{\frac{\Delta t_{var,P,delay.prevMS}^{max} - |\Delta t_{var,P,delay.prevMS}^{min}|}{2} + M \cdot T + t_{hold} + |\Delta t_{var,P}^{min}| + |\Delta t_{var,P,delay.prevMS}^{min}|} - t_{delay.reg} + t_{delay.Static \rightarrow Pulse} + t_{delay.AWP} + t_{delay.Pulse \rightarrow Static} \quad (1)$$

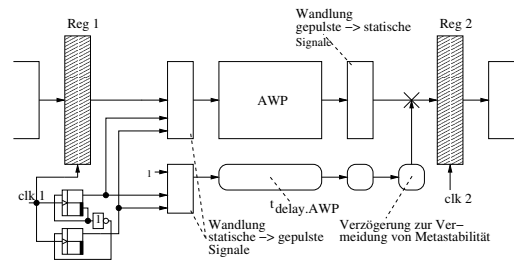


Abbildung 3. Verbindung von AWP und synchroner Pipeline

Schieberegister sind schwieriger zu handhaben, da die Möglichkeit vorhanden sein muß, den Schiebepuls beliebig anzuhalten und wieder fortzusetzen. Dies steht im Gegensatz zu der Eigenschaft der Wave Pipeline, daß Daten von selbst durch die Logik propagieren und nicht aufzuhalten sind. Daher sind geeignete Schieberegisterarchitekturen zu entwickeln, die bei Ausbleiben von Requests die Daten an einer Stelle anhalten und speichern. Die Schieberegister sollen sich nahtlos in ein AWP-System einfügen und keinen Flaschenhals für den Durchsatz darstellen.

Die Möglichkeit, daß synchrone, mit dem Request „getaktete“ Register-Files und Schieberegister am besten in ein asynchrones Wave Pipeline-System passen könnten, stellt kein Problem an sich dar, sondern ist nur ein Hinweis darauf, daß AWP-Architekturen zwischen konventionellen synchronen und klassischen asynchronen Handshake-Architekturen anzusiedeln sind. Es ist von Vorteil, wenn diese Komponenten nicht neu entworfen werden müssen, da sie aus dem synchronen Entwurf schon bekannt sind. Der Gesichtspunkt der Schnittstellen zwischen AWP's und einem synchronen Umfeld ist aber von allgemeinerer Bedeutung. Aufgrund der Dominanz des synchronen Entwurfsstils ist es nicht zu erwarten, daß globale Taktsignale völlig von den Chips verschwinden, so daß AWP's in der Lage sein müssen, mit synchronen Teilsystemen zu kommunizieren. AWP's erhalten diese Eigenschaft, indem der

synchrone Takt und das Request-Signal geeignet miteinander verbunden werden. Dazu wurden Schaltungstechniken entwickelt, die ein Minimum an Latenz und ein Maximum an Sicherheit bieten. Die Anbindung einer asynchronen Wave Pipeline an ein synchrones Umfeld, die Verbindung von Takt und Request sowie die zusätzlichen Verzögerungen zur Vermeidung von Metastabilität sind in Abbildung 3 gezeigt. Es ist zu untersuchen, ob das vorgeschaltete Register (Reg_1 in Abbildung 3) nicht entfallen kann, da statische Eingangssignale für eine AWP nicht nötig sind. Dazu sind verschiedene FF-Implementierungen auf die Länge des Entscheidungs- und Halteintervalls zu untersuchen und mit den Werten der Schaltung zur Pulserzeugung zu vergleichen.

AWP-Datenpfade erfordern sehr schnelle Controller, so daß der Einsatz der Schaltungstechnik aus den Datenpfaden auch in den Controllern notwendig wird, da die Latenz der Controller sonst zu groß wäre für die Steuerung von schnellen AWP-Datenpfaden. Die Verwendung von Wave Pipelining in Controllern ist ein neuartiges Konzept, das durch die AWP-Datenpfade ermöglicht wird. Die Unterschiede zu existierenden Methodiken für asynchrone Steuerwerke sollen herausgearbeitet und formal gefaßt werden und eine Vorgehensweise zum praktischen Entwurf von Controllern für AWP-Datenpfade etabliert werden. Die Arbeiten in diesem Bereich haben begonnen. Dazu werden nach einer formalen Betrachtung der inhärenten Hazard-Freiheit, AWP-FSMs hinsichtlich Fläche, Geschwindigkeit und insbesondere Energieverbrauch anhand von Benchmark-Beispielen mit anderen asynchronen FSMs verglichen. An dieser Stelle fließen Modellgleichungen ein, die zur Zeit entwickelt werden.

Automatisierung des Entwurfs

Basierend auf den dann vorliegenden Modellen der quantitativen Analyse soll die Layout-Erzeugung teilweise automatisiert werden. Ziel ist ein semi-automatischer Entwurf, der von einer Netzliste ausgeht. Die zentrale Aufgabe dabei ist die Berechnung der erforderlichen Treiberstärken für jedes Gatter und Generierung entsprechender Layouts unter Einbeziehung von Interconnect-Einflüssen. Die Erzeugung der Gatter-Layouts soll über die Parametrisierung geeigneter Layout-Templates erfolgen. Die Realisierung des CAE-Programms erfolgt in C und der Sprache *SKILL* des *Cadence Design Framework II*. Das hat den Vorteil, daß die Software-Infrastruktur des eingesetzten CAE-Systems weiter genutzt werden kann. Der Interconnect-Entwurf kann vorerst manuell erfolgen, wobei die parasitären Eigenschaften der Verbindungsstruktur in die Berechnung backannotiert werden. Es ist aber geplant, bekannte Verfahren zur Verdrahtung von Analog-Schaltungen auf ihre Eignung zur automatisierten Interconnect-Generierung

zu untersuchen. Bei zunehmender Skalierung der CMOS-Prozesse und steigender Größe der Systeme nehmen die Verzögerungen der Interconnects die dominierende Rolle ein. Daher ist der Einfluß der Interconnects auf die Delay-Variationen zu berücksichtigen, besonders bei nichtlokalen Verbindungen über Datenpfad-Blöcke hinweg. Im Gegensatz zu den Gattern können Interconnects nicht in einer Bibliothek vorgehalten werden, und ihr genaues Verzögerungsverhalten ist nur leicht kontrollierbar, wenn ein Datenbündel parallel nebeneinander gelegt werden kann. Im allgemeinen sind aber die verwendeten Layer, Vias und Geometrien der einzelnen Bitleitungen in einem Datenbündel nicht identisch. Darüberhinaus muß der Einfluß von Übersprechen durch Layout-Maßnahmen eng kontrolliert werden, um die datenabhängige Verzögerung von Verbindungen zu minimieren. Es ist geplant, eine Software zur Unterstützung des Interconnect-Layouts zu erstellen, die bei gegebener Anordnung der einzelnen Zellen eine gute Interconnect-Struktur findet und dafür das Layout generiert.

Literatur

- [ASYNC-Bib] T. Verhoeff and A. Peeters, *The Asynchronous Bibliography*, <http://www.win.tue.nl/cs/pa/wsina/async.html>.
- [ASNYC-Conf] IEEE ASYNC Conference Series, 1994, 1996, 1997, 1998, 1999.
- [IEEE-Proc] *Proceedings of the IEEE*, Special Issue on Asynchronous Circuits and Systems, Feb. 1999.
- [Paver98] N. C. Paver, P. Day, C. Farnsworth, D. L. Jackson, W. A. Lien, and J. Liu, "A Low-Power, Low Noise, Configurable Self-Timed DSP", *Proceedings Fourth International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 32–42, Mar. 1998.
- [Rotem99] S. Rotem et al., "RAPID: An Asynchronous Instruction Length Decoder", *Proceedings Fifth International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 60–70, Apr. 1999.
- [Dan97] D. Dobberpuhl, "Circuits and Technology for Digital's StrongARM(TM) and ALPHA Microprocessors", *Proceedings 17th Conference on Advanced Research in VLSI*, pp. 2–11, 1997.

- [Cotten69] L. Cotten, "Maximum-rate pipeline systems", *1969 AFIPS Proc. Spring Joint Computer Conf.*, vol. 34, Montvale, NJ: AFIPS Press, pp. 581–586, May 1969.
- [Burlleson98] W. Burlleson, M. Ciesielski, F. Klass, and W. Liu, "Wave-Pipelining: A Tutorial and Research Survey", *IEEE Transactions on VLSI*, vol. 6, no. 3, pp. 464–474, Sep. 1998.
- [Wong93] D. Wong, G. De Micheli, and M. Flynn, "Designing High-Performance Digital Circuits Using Wave Pipelining: Algorithms and Practical Experiences", *IEEE Trans. on CAD*, vol. 12, no. 1, January 1993.
- [Nowka95] Kevin J. Nowka, "High-Performance CMOS System Design Using Wave Pipelining", PhD Thesis, Computer Systems Laboratory, Stanford University, August 1995.
- [Klass94] E. Klass, *Wave Pipelining: Theoretical and Practical Issues in CMOS*, PhD Thesis, Delft University, Sep. 1994.
- [Gray94] C. Gray, W. Liu, and R. Cavin, *Wave Pipelining: Theory and CMOS Implementation*, Kluwer Academic Publishers, 1994.
- [Lien95] W. Lien and W. Burlleson, "Wave-Domino Logic: Theory and Applications", *IEEE Transactions on Circuits and Systems*, vol. 42, no. 2, pp. 78–91, Feb. 1995.
- [Mathew] Sanu Mathew and Ramalingam Sridhar, "Efficient Clocking of a Wave-Domino Pipeline", *Proceedings IEEE Intern. Symp. on Circuits and Systems*, pp. 1832–1835, 1997.
- [Ghosh] Debabrata Ghosh and S. K. Nandy, "Design and Realization of High-Performance Wave-Pipelined 8×8 Multiplier in CMOS Technology", *IEEE Trans. on VLSI*, vol. 3, no. 1, pp. 36–48, March 1995.
- [Zhang] X. Zhang and R. Sridhar, "CMOS Wave Pipelining Using Transmission Gate Logic", *Proceedings of the IEEE International ASIC Conference and Exhibit*, Rochester, NY, 1994.
- [Heald98] R. Heald et al., "64-KByte Sum-Addressed-Memory Cache with 1.6-ns Cycle and 2.6-ns Latency", *IEEE Journal of Solid-State Circuits*, vol. 33, no. 11, pp. 1682–1689, Nov. 1998.
- [Chappell91] T. Chappell et al., "A 2-ns Cycle, 3.8-ns Access 512-kb CMOS ECL SRAM with a Fully Pipelined Architecture", *IEEE Journal of Solid-State Circuits*, vol. 26, no. 11, pp. 1577–1585, Nov. 1991.
- [Hauck98b] O. Hauck and S. A. Huss, "Asynchronous Wave Pipelines for High Throughput Datapaths", *Proceedings IEEE 5th International Conference on Electronics, Circuits and Systems*, pp. 283–286, Lissabon, Sep. 1998.
- [Hauck99a] O. Hauck, M. Garg, and S. A. Huss, "Efficient and Safe Asynchronous Wave-Pipeline Architectures for Datapath and Control Unit Applications", *Proceedings IEEE 9th Great Lakes Symposium on VLSI*, pp. 38–41, Ann Arbor, Mar. 1999.
- [Hauck99b] O. Hauck, M. Garg, and S. A. Huss, "Two-Phase Asynchronous Wave-Pipelines and their Application to a 2D-DCT", *Proceedings IEEE 5th International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC)*, pp. 219–228, Barcelona, Apr. 1999.
- [Hauck99c] O. Hauck and S. A. Huss, "Asynchronous Wave-Pipelines with Improved Logic and Latch Circuits", *Proceedings IEEE 2nd Workshop on Design, Test, and Applications (WDTA)*, pp. 13–16, Dubrovnik, June 1999.
- [Hauck99d] O. Hauck, S. Mietens und S. A. Huss, "Giga-Hertz SRT-Division mit asynchronen Wave-Pipelines", *Tagungsband GI/GMM/ITG Fachtagung Entwurf Integrierter Schaltungen (E.I.S.)*, S. 29–36, Darmstadt, Sep. 1999.
- [Hauck00a] O. Hauck and S. A. Huss, "Circuit Design for SRCMOS Asynchronous Wave Pipelines", *Proceedings 4th Asynchronous Circuit Design Workshop (ACID)*, Grenoble, Feb. 2000.
- [Hauck00b] O. Hauck, A. Katoch, and S. A. Huss, "VLSI System Design Using Asynchronous Wave Pipelines: A 0.35 μm CMOS

1.5 GHz Elliptic Curve Public Key Cryptosystem Chip”, *Proceedings IEEE 6th International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC)*, pp. 188–197, Eilat, Apr. 2000.

[HeHu01] S. Hermanns and S. A. Huss, „Embedding of Asynchronous Wave Pipelines into Synchronous Data Processing“, 4th Sophia Antipolis forum on MicroElectronics, Nov. 2001

[Herm01] „Synchrones Einlesen von asynchronen gepulsten Signalen als Ausgabe einer Wave Pipeline“, Interner Bericht VIVA-1, Fachgebiet ISS, Technische Universität Darmstadt, 2001