Abstract

This work describes the systematic (semi-) automatic design of asynchronous circuits in a design environment for synchronous circuits. The methodology developed in this work allows to implement any behavioural model in stages of an asynchronous pipeline. Models can therefore be described with standard hardware description languages (VHDL, Verilog, EDIF) and are allowed to have storing elements. As a result a gate netlist is obtained which timing behaviour is validated by a pre-layout timing simulation. Controlling of each stage can be done with standardised or new defined control automata. DGC [45] which can generate hazard free circuits is used to synthesise the new defined control automata. Other synthesis steps and simulation are done with the common tool synopsys [94]. Integrating the asynchronous moduls into an synchronous design is possible without problems. Further an extensive catalogue of controllers for the four phase protocol was created. Additional interface modules are also part of the catalogue. These interface modules are for example connections to synchronous modules. The capacity of the software ASMOGEN as a part of this work is shown in a comparison between a synchronous implementation and an asynchronous one of a Reed Solomon decoder (symbolwidth is 12 bits; wordlength is 2720 symbols).

Kurzzusammenfassung

Diese Arbeit beschreibt den systematischen (semi-) automatischen Entwurf asynchroner Schaltungen in einer Entwurfsumgebung für synchrone Schaltungen. Die erarbeitete Methodik erlaubt es, beliebige Verhaltensmodelle in asynchrone Piplinestufen zu realisieren. Die Modelle können hierzu in einer standardmäßigen Hardware-Beschreibungssprache (VHDL, Verilog, EDIF) verfasst und nach Wunsch auch mit Speichern versehen sein. Als Ergebnis erhält man eine Gatternetzliste, die auf ihr Zeitverhalten mittels einer Pre-Layout-Simulation überprüft ist. Die Steuerung der einzelnen Stufen kann über standardisierte oder selbst vorgegeben Kontrollautomaten erfolgen. Zur Synthese der selbst vorgegeben Kontrollstrukturen wird die frei verfügbare Synthesesoftware DGC [45] verwendet, die hazardfreie Schaltungen erzeugen kann. Für die restlichen Syntheseschritte und Simulationen wird die Standardsoftware Synopsys [94] verwendet. Eine Einbindung der asynchronen Module in eine synchrone Einsatzumgebung ist ohne Probleme möglich. Im Rahmen dieser Arbeit ist ein umfangreicher Katalog an Kontrollstrukturen für das Vierphasenprotokoll entstanden. Zu diesem Katalog zählen auch weitere Schnittstellenmodule, wie z.B. für die Anbindung an synchrone Module. Die Leistungsfähigkeit der erstellten Software ASMOGEN wurde anhand des Vergleiches der asynchronen Implementierung mit der synchronen Implementierung eines Read Solomon Dekoders (Symbolbreite ist 12 Bits, Wortlänge ist 2720 Symbole) nachgewiesen.

1 Einführung

Die minimal mögliche Strukturgröße innerhalb einer integrierten Schaltung sinkt über die Zeit exponentiell. In [36] wird die Entwicklung von Mikroprozessoren von 1997 bis 1999 und eine Prognose bis 2002 gezeigt. Die Anzahl der Transistoren pro Mikroprozessoren nimmt stetig zu. Bis 2002 führt dieser Anstieg zu geschätzten 130 Millionen Transistoren auf einem einzigen Chip. Diese Entwicklung führt zu Problemen, da die Entwickler nicht in dem gleichen Maß ihre individuelle Produktivität steigern können. Hier müssen entsprechend neue Softwarewerkzeuge mit höherem Automatisierungsgrad sowie neue Methoden und Wege der Entwicklung helfen, diese Lücke der Entwicklerproduktivität zu der Komplexität der zu entwerfenden Bausteine zu verringern.

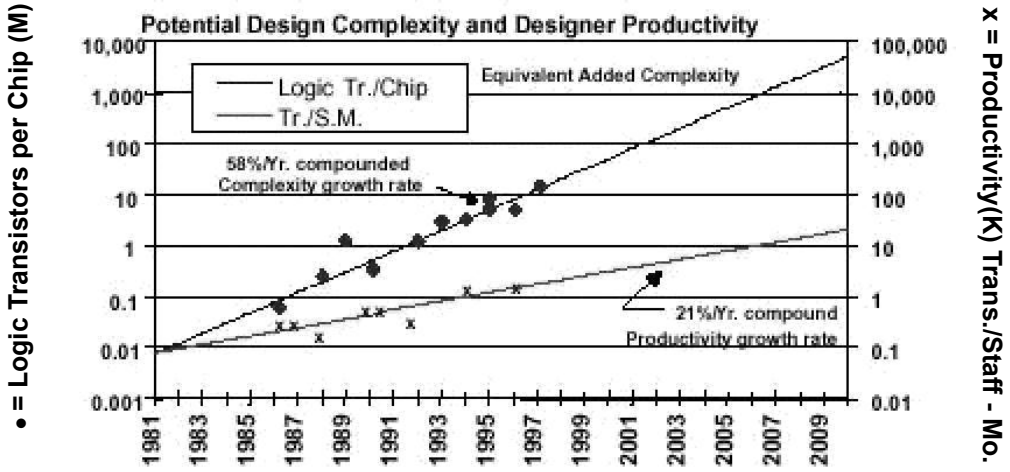

Bild 1.1: Lücke in der Entwurfsproduktivität (Figure5 ITRS)

Um den Sachverhalt und die Implikationen dieser Steigerung der Komplexität zu verdeutlichen, gibt es einen anschaulichen Vergleich mit Straßennetzen. Während die Komplexität einer integrierten Schaltung des Jahres 1963 in einer 25 µm CMOS-Technologie dem Straßennetz einer mittelgroßen Stadt mit ca. 4 km^2 Fläche entsprach, müsste man die Komplexität einer integrierten Schaltung des Jahres 1993 in einer 0,5 µm CMOS-Technologie mit einem Straßennetz vergleichen, welches mit ca. 9 Millionen km^2 schon die Größe Europas überbietet. Mit einer 0,13 µm CMOS-Technologie im Jahr 2001 umspannt das vergleichbare Straßennetz mit etwa 2,3 Milliarden km^2 schon ein Vielfaches der Erdoberfläche (~500 Millionen km^2).

Um den Vergleich zu Ende zu führen, muss man sich jetzt vorstellen, dass letzteres Straßensystem mit nur einem einzigen zentralen Signal gesteuert wird. Für ein Straßennetz erscheint dies unmöglich, in integrierten Schaltungen wird aber genau dieses gemacht. Nahezu immer steuert ein einziges globales Takt-Signal den gesamten Datenverkehr auf einem Baustein. Dies ist aus elektrischen Gründen auch hier nicht ohne erhebliche Probleme und Entwurfsaufwand zu erreichen.

Auch in [36] werden diese Probleme festgehalten und eine mögliche Lösung angegeben:

„*The fastest buffered interconnect for a long wire can be over 100x the switching time of an individual gate. Given expected global clock rates approaching 3 GHz, it will be difficult to achieve synchronous, on-chip operation without introducing multi-cycle latencies in the interconnect. System design must comprehend the timing issue more fully*“

„*The large currents being introduced with increasing power densities and lower voltages all lead to larger supply rail inductive noise. Synchronous systems worsen the problem by scheduling the power switch surges around regular time periods. Thus, supply rail design to reduce effects such as voltage drop or current surges are required early in the design process.*“

„*Process technology support for increased switching is not keeping up with the trend in clock rate increase. Thus, design techniques such as reducing the number of logic levels between clocked registers allow the current trend for clock rate increase. Careful consideration of single versus multi-cycle paths and circuits will become more important early in the design process. Specifically, locally synchronous but globally asynchronous design techniques will need to be supported by the tools.*“

Ein vorgeschlagener möglicher Lösungsweg aus den oben beschriebenen Problemen ist also die Nutzung asynchroner Schaltungskonzepte. Das heißt, dass nicht mehr ein einzelnes globales Signal den Datentransfer steuert, vielmehr wird es lokale beziehungsweise lokal erzeugte Steuersignale geben, die jeweils ein Modul steuern. Letztendlich ist der integrierte Baustein aus einzelnen Modulen aufgebaut, die ihrerseits in sich synchron arbeiten. Die Kommunikation zwischen den Modulen findet aber asynchron statt. Dabei kann die Größe der einzelnen Module unterschiedlich sein und auch die Module selbst können asynchron arbeiten.

1.1 Motivation für den Entwurf asynchroner Schaltungen

Der Entwurf asynchroner Schaltungen wird im Moment nicht in der Weise unterstützt, wie der Entwurf synchroner Schaltungen. Dies liegt zum einen an der großen Menge an Erfahrungen, die sich in den Softwarefirmen und Hardwareentwicklungsfirmen gesammelt haben. Zum anderen reichten bisher synchrone Schaltungen weitestgehend aus. Mit den zunehmenden Problemen im Entwurf synchroner Schaltungen wird aber gerade die Nutzung asynchroner Schaltungen interessanter.

Als Hauptursache der Probleme synchroner Schaltungen ist der Takt selbst zu betrachten. Da die Schaltungen immer größer werden, müssen immer größere Taktnetze erzeugt werden. Dieses bedingt ein immer größer werdendes Taktnetz, welches auszubalancieren ist. Das wieder bedeutet einen erhöhten Energieverbrauch und auch Platzbedarf. Neben den aufkommenden Designschwierigkeiten sind noch andere Aspekte eines globalen Taktes zu betrachten. Das sind

zum Beispiel die Elektromagnetische Verträglichkeit oder auch der Schutz der Informationen vor unerwünschtem Zugriff.

Im Einzelnen erhofft man sich durch den Wegfall eines globalen Taktes mehrere Vorteile.

1.1.1 Vereinfachung globaler Zeitaspekte

Im synchronen Entwurf ist üblicherweise eine bestimmte Taktrate vorgegeben, die durch die spätere Einsatzumgebung bedingt ist. Innerhalb der synchronen Schaltung muss man alle Pfade so trimmen, dass die Propagierungszeit durch diese kürzer als die Taktperiode minus der Setup-Zeit des nachfolgenden Speicherelements ist. Im asynchronen Entwurf interessieren hier nur die absoluten Zeiten („bis wann muss der Ausgang reagiert haben?“), so dass hier weniger Pfade zeitlich zu optimieren sind.

1.1.2 Bessere Schnittstellen zur Umwelt

Ein weiteres Problem der synchronen Schaltung stellt die Einsynchronisation externer (asynchroner) Signale dar. Eine Bedingung, die dabei einzuhalten ist, ist der sichere gegenseitige Ausschluss, d.h. es darf nicht gleichzeitig von außen (Daten) und innen (Takt) ein Zugriff auf die Einsynchronisierungsflipflops erfolgen (Einhaltung der Setup- und Hold- Zeiten).

Bei einer asynchronen Schaltung entfällt dieses Problem im Allgemeinen, auch wenn hier trotzdem Zeitbedingungen auftreten, die eingehalten werden müssen.

1.1.3 Geschwindigkeit

Die in [36] angesprochenen Probleme der Laufzeiten auf Leitungen werden mit zunehmender Größe der integrierten Bausteine immer dramatischer. Die Gatterverzögerungszeiten nehmen durch Verbesserungen in der Prozesstechnologie im größeren Maße ab, als dies für die Laufzeiten auf den Verbindungsleitungen der Fall ist. Da somit das Verhältnis Leitungslaufzeit zu Gatterverzögerungszeit signifikant steigt, muss den Leitungslaufzeiten beim Entwurf mehr Aufmerksamkeit gewidmet werden.

Vorrangiges Ziel sind kurze Verbindungen zwischen den Gattern einer Schaltung. Die für diesen Optimierungsschritt relevanten Platzierungs- und Verdrahtungswerkzeuge benutzen über die Jahrzehnte verbesserte und ausgereifte Algorithmen. Allerdings sind diese Algorithmen bei der Verkürzung der Leitungslänge globaler Signale nutzlos, da diese globalen Signale meist auf dem gesamten Baustein verfügbar sein müssen. Bei einer synchronen Schaltung ist es vor allem die Taktleitung mit den entsprechenden Signallaufzeiten, die den maximal möglichen Datendurchsatz beschränken.

Asynchrone Schaltungen werden nicht zentral über einen globalen Takt gesteuert, sondern dezentral über lokale Steuerelemente. Somit kommen asynchrone Schaltungen ohne einen globalen Takt aus. Das Fehlen globaler Signale in asynchronen Schaltungen macht den Datendurchsatz unabhängig von den Laufzeiten globaler Signale. Auf dieser Tatsache beruht das Potential asynchroner Schaltungen, eine größere durchschnittliche Performanz zu erreichen, als die entsprechenden synchronen Implementierungen.

1.1.4 Keine Clock Skew Probleme

Eine synchrone Schaltung basiert auf der „zeitgleichen“ Abarbeitung der Daten, zum Beispiel gesteuert durch eine steigende Flanke eines Taktsignals. Der hierfür verwendete Takt muss aber im gesamten Entwurf gleichzeitig aktiv sein. Der tatsächlich vorhandene Zeitunterschied des Auftretens dieser Taktflanke an unterschiedlichen Orten bezeichnet man mit *Clock Skew*, beziehungsweise *Taktversatz*. Bei der zunehmenden Fläche heutiger integrierter Schaltungen bei gleichzeitiger Miniaturisierung der Schaltelemente nehmen die relativen Leitungslängen und damit die Signallaufzeiten zu. Somit wird es auch immer schwieriger, die Bedingung der Gleichzeitigkeit zu erfüllen.

Während hier bei synchronen Schaltungen ein nicht unerheblicher Entwurfsaufwand entsteht, fällt dieser bei asynchronen Schaltungen weg, da kein globaler Takt zum Einsatz kommt. Die lokal erzeugten Takte und die für einzelne (kleine) Module benötigten (globalen) Takte kann man wesentlich einfacher implementieren.

1.1.5 Durchschnittliche Performanz

Mit der Entscheidung, alles mit einem globalen Takt zu steuern, steht beim synchronen Entwurf ein weiterer Nachteil in Verbindung. Die Taktfrequenz kann nicht höher sein, als es der längste Pfad der Schaltung zulässt. Erst wenn Daten durch diesen propagiert sind und am Ende dieses Pfades für die Zeit T_{setup} stabil anliegen, darf die nächste Taktflanke auftreten. Damit bestimmt die größte Propagierungszeit die Performanz der Schaltung.

Viele asynchrone Schaltungen besitzen daher einen Mechanismus, der feststellt, wann ein Datum vollständig verarbeitet ist und als Ergebnis in der nächsten Stufe weiterverarbeitet werden kann. Da die Propagierungszeit datenabhängig ist, zeigt die asynchrone Schaltung eine von den Daten abhängige Performanz. Es ergibt sich daher für asynchrone Schaltungen eine bessere durchschnittliche Performanz.

1.1.6 Geringere Abhängigkeit von physikalischen Eigenschaften

Die Laufzeiten durch einen Pfad variieren auf Grund unterschiedlicher Umgebungsparameter (Temperatur, Versorgungsspannung, ...). In einer synchronen Schaltung muss man immer die schlechtesten Parameterkombinationen zur Bestimmung der maximalen Taktfrequenz und damit der Performanz der Schaltung betrachten.

Im Gegensatz zum synchronen Entwurf macht dies im asynchronen Entwurf keine größeren Probleme. Durch die verwendeten Mechanismen zur Detektion der vollständigen Abarbeitung von Daten ist eine asynchrone Schaltung unempfindlicher gegen globale Parameterschwankungen. Sie arbeitet weiterhin korrekt, wobei sich die Performanz der asynchronen Schaltung entsprechend den Parameterwerten verändert.

1.1.7 Unabhängiger von Technologieänderungen

Ein Problem des synchronen Entwurfs ist die Anpassung an eine neue Technologie beziehungsweise an den Herstellungsprozess eines anderen Anbieters. Es ändern sich die Laufzeiten durch die verschiedenen Pfade und dementsprechend muss man die Schaltung den neuen Bedingungen anpassen. Unter Umständen ist entweder eine niedrigere Taktfrequenz zu wählen (unrealistisch), oder die nun neu entstandenen längsten Laufzeiten müssen gekürzt werden. Es ist auf jeden Fall ein Eingriff in eine bestehende Schaltung notwendig.

Auch hier hat eine asynchrone Schaltung mit Komplettierungsdetektionsmechanismus den Vorteil, weniger abhängig von solchen Änderungen der Technologie oder des Herstellungsprozesses zu sein. Es sind hier keine oder nur minimale Änderungen notwendig.

1.1.8 Sicherheit

In einer synchronen Schaltung wird die Datenverarbeitung gleichzeitig mit der aktiven Taktflanke gestartet. Das kann einerseits zu Störungen umliegender Schaltungen führen. Andererseits können damit auch Informationen über die Funktionsweise des Bausteins gewonnen werden. Bei einer asynchronen Schaltung lässt sich das Spektrum verschmieren, was das Gewinnen von Informationen aus dem Betrieb des Bausteins wesentlich erschwert.

1.1.9 Bessere Elektromagnetische Verträglichkeit

In synchronen Schaltungen ist die Datenverarbeitung, gesteuert durch den globalen Takt, schrittweise durchzuführen. Mit jeder aktiven Taktflanke startet in der gesamten Schaltung gleichzeitig eine neue Berechnung. Dies führt zu hohen Versorgungsstromspitzen, die bei einer asynchronen Schaltung vermeidbar sind. Hier hat man die Möglichkeit die Abarbeitung der Daten so zu steuern, dass keine Störspitzen auftreten.

1.1.10 Low Power

Bei einer synchronen Schaltung entsteht durch kontinuierliche Pegelwechsel auf der Taktleitung Verlustleistung, auch wenn sich der Rest der Schaltung in „Ruhe“ befindet. Somit wird in einer synchronen Schaltung immer dann unnötig Energie verbraucht, wenn diese Schaltung nichts zu tun hat.

Allerdings kann der Energieaufwand des Taktnetzwerkes auch in einer „aktiven“ Schaltung bis zu 50% des Gesamtverbrauchs der Schaltung ausmachen. Also selbst wenn die Schaltung Daten verarbeitet, ist das Taktnetz immer noch der Hauptverbraucher von Energie.

Asynchrone Schaltungen besitzen keinen globalen Takt. Damit fällt ein wichtiger Leistungsverbraucher weg. Hat eine asynchrone Schaltung nichts zu tun, so befindet sie sich vollständig in Ruhe und es wird kaum Energie verbraucht (vergleiche Kapitel 2.6.2). Andererseits benötigen Daten verarbeitende asynchrone Schaltungen oftmals auch einen selbstgenerierten Takt, der also nur dann aktiv ist, wenn neue Daten anliegen. Da dieser selbstgenerierte Takt aber nur lokal benötigt wird, ist dessen Taktnetz mit weniger Treibern versehen, als ein Taktnetz in einer synchronen Schaltung. So verbrauchen diese lokalen Taktnetze in der Summer weniger Energie als ein vergleichbarer globaler Takt in einer synchronen Schaltung.

Somit sind asynchrone Schaltungen eine attraktive Alternative für Low Power Anwendungen.

1.2 Einordnung dieser Arbeit

Ziel dieser Arbeit ist es, den Entwurf asynchroner Schaltungen in einen Standard- Entwurfsablauf einzubinden, der für den Entwurf synchroner Schaltungen ausgearbeitet ist. Der Vorteil dieser Vorgehensweise liegt darin, dass der Hardwareentwickler in der gleichen Entwurfs- und Softwareumgebung sein gesamtes System entwickeln kann. Somit ergibt sich die Möglichkeit, einzelne asynchrone Module in ein System einzubauen, um zum Beispiel den Verbrauch des verbrauchsstärksten Moduls zu senken. Damit können die zu erwartenden Vorteile einer asynchronen Schaltung genutzt und gezielt eingesetzt werden. Eine konsequente Anwendung des dargestellten Weges erlaubt den Entwurf eines asynchronen Systems mit Hilfe der Standard-Entwurfswerkzeuge für synchrone Schaltungen mit vergleichsweise geringen Änderungen.

Um zu gewährleisten, in einer Entwurfsumgebung bleiben zu können, werden die Architekturen synchroner Schaltungen angepasst und zusätzlich zu entwickelnde Steuerlogik interaktiv generiert. Der Entwurf asynchroner Module wird mit Hilfe einer standardisierten Hardware-

Beschreibungssprache (VHDL[1] [35]) und einer gängigen Simulations- sowie Synthesesoftware bewerkstelligt.

1.3 Übersicht

In Kapitel 1.1 wurden die Vorteile asynchroner Schaltungen dargelegt. Diese sind als zu erreichende Ziele anzusehen. Das heißt man kann nicht erwarten, eine Variante asynchroner Schaltungen beziehungsweise eine Methodik zu finden, die all diese Vorteile aufweist. Vielmehr geht es darum, einzelne Vorteile auszunutzen und überhaupt erst zu erzielen. Wie dies erfolgen kann, wird in dieser Arbeit gezeigt.

Die **Grundlagen** zum Thema **asynchroner Schaltungen** werden in Kapitel 2 erläutert. Hier werden Definitionen zur Asynchronität und Klassifizierungsmerkmale vorgestellt. Des Weiteren werden einzelne Aspekte aufgedeckt, die bei asynchronen Schaltungen zu beachten sind. Der globale Takt selbst wurde als größtes Hindernis synchroner Schaltungen in Kapitel 1.1 dargestellt. Das Taktnetz ist vor allem auch ein großer Energieverbraucher. Die **Entstehung der Verlustleistung** wird in Kapitel 2.6.2 beschrieben.

Für das Aufstellen einer effektiven Methode und einer entsprechenden Software, ist der Überblick über **existierende Werkzeuge** und **Implementierungsvarianten asynchroner Schaltungen** wichtig. Dieser wird in Kapitel 3 gegeben.

Der Haupteil der Arbeit ist in Kapitel 4 beschrieben. Kapitel 4.1 beschreibt die erarbeitete **Methodik** und die Arbeitsschritte der entwickelten Software. Der **Entwurf des Datenpfades** ist sehr eng an den Standardentwurfsablauf synchroner Schaltungen angelehnt. Der **Entwurf des Kontrollpfades** hingegen benötigt zusätzliche Schritte und erfolgt mit Hilfe einer speziellen Synthesesoftware (DGC [45]).

Die für die gewählte Implementierungsvarianten relevanten Schnittstellen werden in Kapitel 4.2 vorgestellt. Es wird eine systematische **Klassifizierung** möglicher (Standard-) **Kontrollstrukturen** und die einzelnen **Implementierungen** dieser Kontrollstrukturen vorgestellt. Zudem werden spezielle - aber häufig vorkommende - Schnittstellenmodule vorgestellt.

In Kapitel 5 werden zwei **Beispiele** für realisierte asynchrone Schaltungen gezeigt. Das erste Beispiel zeigt, dass die gewählte Methode nicht nur für rein datenverarbeitende Schaltungen einsetzbar ist. Bei dem Beispiel handelt es sich um einen fiktiven Controler für eine **digitale Fotokamera**. Das zweite Beispiel ist ein größeres Beispiel aus der Praxis: Ein **Read-Solomon Decoder** mit einer Symbolbreite von 12 Bit, 170 Symbolen pro Frame und der Fähigkeit bis zu 85 Fehler korrigieren zu können. Der Decoder ist für eine Anbindung an einen 40Gbit Daten-

1. **V**HSIC (Very High Speed Integrated Circuits) **H**ardware **D**escription **L**anguage