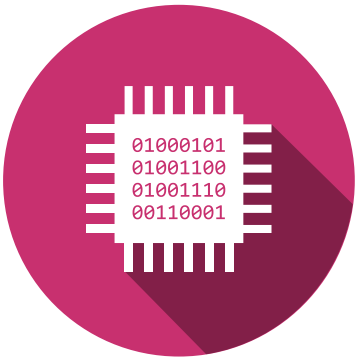
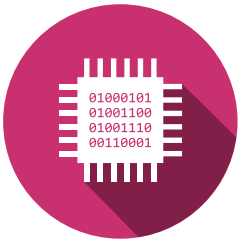


Methodologie für die Entwicklung von digitalen Schaltungen (MET)

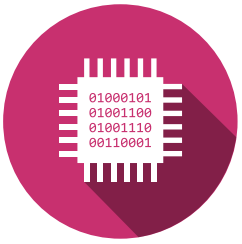
Vorlesung Digitales Design

Hes·so VALAIS WALLIS
Haute Ecole d'Ingénierie
Hochschule für Ingenieurwissenschaften

Orientierung: Systemtechnik (SYND)
Kurs: Digitales Design (DiD)
Verfasser: Christophe Bianchi, François Corthay, Pierre Pompili, Silvan Zahno
Datum: 25. August 2022
Version: v2.1

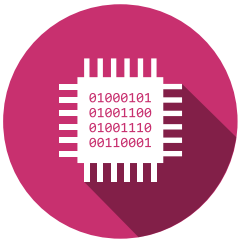


Inhaltsverzeichnis



1 Einführung

Dieses Dokument enthält die Richtlinien für die Entwicklung von digitalen Schaltungen, von deren Definition in Form eines Pflichtenhefts bis hin zu ihrer Validation im Rahmen eines umfassenden Systems.



2 Entwicklungsmodell: V-Diagramm

2.1 Prinzip des V-Diagramms

Von den zahlreichen existierenden Entwicklungsmodellen ist das V-Diagramm zweifellos dasjenige, das am meisten verwendet wird. Hauptmerkmale:

- Ziel jeder Etappe ist die Ausarbeitung eines Zwischenprodukts.
- Am Ende jeder Etappe wird eine Kontrolle zur frühzeitigen Aufdeckung und Elimination von Anomalien und Ungenauigkeiten durchgeführt. Der Übergang zur nächsten Etappe hängt vom Resultat dieser Kontrolle ab.
- Wenn ein Schritt zurück gemacht werden muss, sollte sich dieser auf die unmittelbar davor liegende Etappe beschränken. Es wird empfohlen, dass ein genehmigtes Produkt nur mittels einer formellen Änderungsprozedur verändert werden kann.

2.2 Anwendung des V-Diagramms bei der Entwicklung von digitalen Schaltungen

Die Abbildung 1 zeigt ein Modell eines V-Diagramms, angewandt auf den Entwicklungsprozess von digitalen Schaltungen.

Dieses Diagramm umfasst folgende Entwicklungsphasen:

- Die Spezifikationsphase umfasst alle vorgängigen Definitions- und Analyseaktivitäten (Machbarkeit).
- Die Entwurfsphase umfasst alle Architektur- und Designaktivitäten (Zerlegung in Blöcke).
- Die Validationsphase umfasst Teilschaltungs-, Funktions- und Leistungstests.
- Im Rahmen der Integrationsphase wird das Produkt in seiner Anwendungsumgebung validiert.

Diese Einteilung in Phasen verringert die Anzahl Sitzungen sowie den gegenseitigen Austausch von Dokumenten und wird von den Unternehmen sehr oft angewandt.

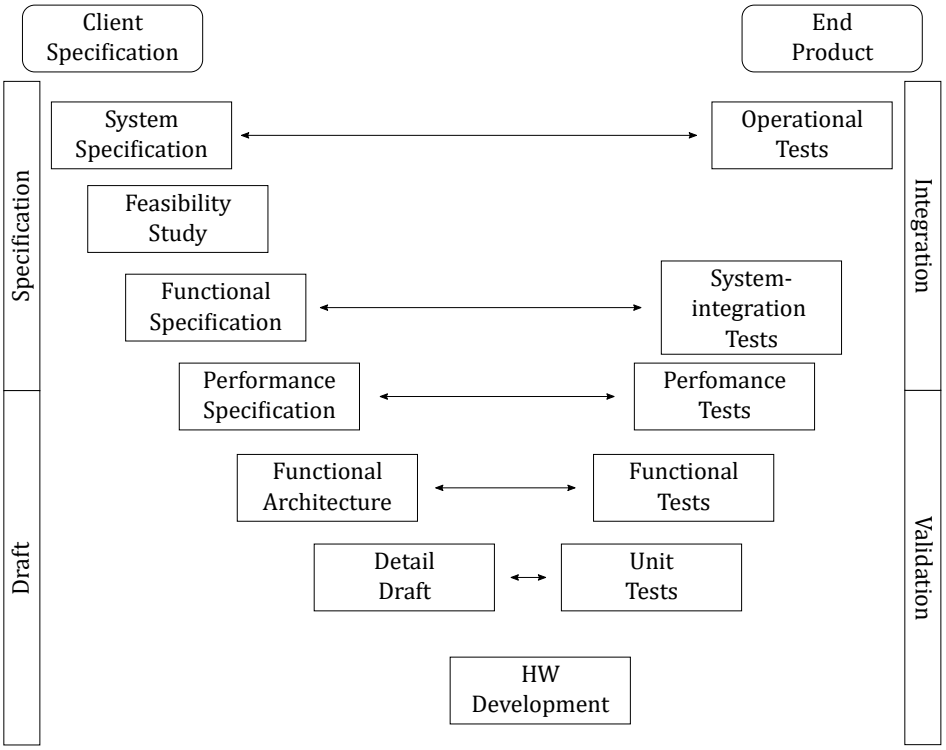


Abbildung 1: V-Diagramm

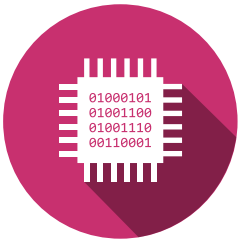
2.3 Dokumentation und Planung der Aktivitäten

Damit diese Technik erfolgreich ist, müssen folgende Elemente berücksichtigt werden:

- Vorgängige Zusammenstellung einer Dokumentation: Vor der Herstellung eines jeden Produkts muss eine Dokumentation erstellt werden.
- Gute Planung der zukünftigen Aktivitäten im Zusammenhang mit den vorgesehenen Studien und der Validation.

Ein grundlegendes Element des V-Diagramms sind die horizontalen Pfeile, die Standardvorgehen darstellen, die auf allen Entwicklungsebenen gelten: Die Phase im linken Teil muss die Test- und Validationsverfahren ausarbeiten, die in der Phase auf derselben Ebene im rechten Teil ausgeführt werden. Mittels dieses Verfahrens wird der klassische Fehler vermieden, das Ergebnis anhand dessen zu validieren, was entwickelt wurde, anstatt dessen, was entwickelt werden sollte. Es ermöglicht unter anderem:

- die Gesamtentwicklungskosten durch eine bessere Planung zu senken (weniger Zeitverluste),
- die eingesetzte Zeit auf die Spezifikations- und Entwicklungsaktivitäten (Vortests) zu übertragen, die bis zu 50% der gesamten Entwicklungsdauer darstellen.



3 Spezifikationsphase

Die Startphase eines Projekts ist eine delikate Phase, da sie einen Wissenstransfer zwischen dem Kunden und dem für die Realisierung des Produkts verantwortlichen Projektteam bedingt. Die Beteiligten haben nur wenig Zeit, sich kennen und verstehen zu lernen und haben in der Regel weder denselben kulturellen Hintergrund noch dieselben Erwartungen vom Projekt. Um eine ausreichende Homogenität zu erzielen, muss deshalb ein Dialog stattfinden. Nur wenn die Erwartungen des Kunden umfassend und eindeutig erfasst werden, können Qualität erzielt und das Projekt korrekt organisiert werden.

Zu Beginn des Projekts müssen die funktionalen und nicht funktionalen Erwartungen des Kunden deshalb klar definiert werden. Diese Etappe ist die Spezifikation der Bedürfnisse. Es geht dabei vor allem darum, den Dialog zwischen dem Kunden und den Entwicklern zu regeln, um sich eindeutig und umfassend darüber einig zu werden, wozu der Schaltkreis dienen soll.

3.1 Pflichtenheft

Es ist wichtig, zwischen dem Spezifikationsdokument und dem Pflichtenheft zu unterscheiden. Das Pflichtenheft wird ausschliesslich vom Kunden verfasst. Er identifiziert alle seine Bedürfnisse (Beweggründe), ohne dabei elektronische Lösungen zu nennen. Das Spezifikationsdokument hingegen legt fest, wozu der Schaltkreis dienen soll. Es beschreibt aus Sicht des Anwenders eine materielle Lösung für die im Pflichtenheft ausgedrückten Bedürfnisse.

Falls zu Beginn des Projekts kein Pflichtenheft vorhanden ist, bedeutet dies für die Entwickler in der Spezifikationsphase einen Mehraufwand, weil sie zusätzlich die Bedürfnisse definieren müssen. Ein Kunde beurteilt nämlich den Erfolg eines Projekts nicht anhand des genialen und zuverlässigen Schaltkreises, den er geliefert bekommt, sondern anhand der Berücksichtigung seiner Bedürfnisse bezüglich des Materials sowie des Mehrwerts, der für ihn entsteht.

3.2 Spezifikationsdokument

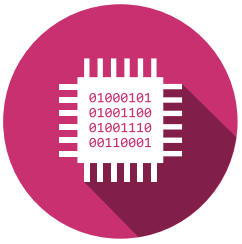
Das Spezifikationsdokument ist sicherlich das Dokument, dessen Erstellung am heikelsten ist und am längsten dauert. Es muss eine vollständige und unzweideutige Vision der gesamten Schaltung liefern. Folgende Punkte müssen dabei berücksichtigt werden:

- Allgemeine Spezifikationen: Name der Schaltung, elektrische Kenndaten (Strom, Ladung, Niveau usw.), Betriebsumgebung.
- Funktionale Spezifikationen: Ein-/Ausgänge, Schnittstellen mit anderen funktionellen Blöcken, Beschrieb der Funktionen.
- Leistungsspezifikationen: spezifische Ablaufdiagramme, Anwendungshäufigkeit, Verbrauch, Grösse.

Dieses Dokument muss die Sicht des Benutzers der Schaltung ausdrücken und auch für den Kunden verständlich sein, der die Schaltung benutzt.

3.3 Eingesetzte Mittel

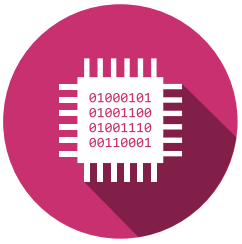
Der grösste Teil der Spezifikationsarbeit basiert auf dem Dialog zwischen dem Entwicklungsteam und allen Personen, die am Projekt beteiligt sind. Um effizient zu sein, müssen diese Gespräche vom Entwicklungsteam entsprechend organisiert und vorbereitet werden. Um den Zeitaufwand der



Kunden für solche Gespräche zu optimieren, können Sitzungstechniken angewandt werden.

Für die Validation der Spezifikation am Ende jeder Phase ist jeweils eine gemeinsame Schlussprüfung notwendig. Doch auch während der Phase selbst sollten mehrere Treffen organisiert werden, um zu verhindern, dass sich das Projekt in eine falsche Richtung bewegt. Nicht selten werden deshalb zwei oder drei Teildokumente zur Vorbereitung erstellt, bevor das definitive Dokument unterbreitet wird.

Alle in dieser Phase erstellten Dokumente müssen vom Kunden validiert werden, d.h. sie müssen klar und verständlich verfasst werden. Dies bedeutet nicht, dass keine Modellierungstechniken verwendet werden dürfen, sondern dass diese erklärt und für den Kunden verständlich gemacht werden müssen. Die Wahl der Techniken hängt somit stark von der Kultur des Kunden und der Art des Problems ab, das gelöst werden muss.



4 Entwurfsphase

Dieses Kapitel enthält einige Empfehlungen bezüglich der Vorgehensweise beim Entwurf einer digitalen Schaltung anhand der Spezifikationen.

4.1 Zerlegung nach Funktionen

In einem ersten Schritt ist es für den Ingenieur nützlich, die Schaltung anhand der Spezifikationen in verschiedene funktionelle Blöcke zu zerlegen. Jeder Block sollte die verschiedenen physikalischen Teile der Schaltung darstellen (Eigenfunktionen [„Glue Logic“, Auswahl, Zähler, Sequenziereinheit, Lesen der Daten, Filtern der Daten...] oder spezifische Schaltflächen [digitale Eingänge, digitale Ausgänge, **R**andom **A**ccess **M**emory (RAM), **F**irst **I**n **F**irst **O**ut (FIFO)...]).

Jeder Block kann je nach Komplexität wiederum in verschiedene Module zerlegt werden. Ein für die Sequenzierung der ASIC dienender Block kann somit in mehrere Zähler, Zustandsanzeigen oder Schaltglieder aufgeteilt werden.

Aus Gründen der Übersichtlichkeit kann jedes Modul in Untermodule aufgeteilt werden. Für eine globale logische Funktion scheinen drei bis fünf Ebenen angebracht, um sowohl die Schaltung zu vereinfachen (mehrere einfachere funktionelle Blöcke) als auch klare Schaltbilder zu erstellen (Anzahl Gatter auf einem Schaltbild).

4.2 Architekturdokument

Bevor der Entwurf beginnt, müssen alle Blöcke, Module und Untermodule anhand ihrer Eingänge, Ausgänge und allgemeinen Funktionsweise beschrieben oder bildlich dargestellt werden. Diese Aufteilung in funktionale Blöcke dient als Grundlage für die Ausarbeitung des Architekturdokuments der digitalen Schaltung.

Das Architekturdokument enthält die Zerlegung der digitalen Schaltung in funktionelle Blöcke:

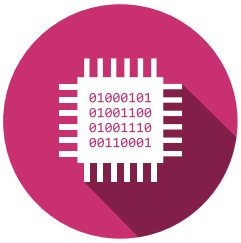
- Kurzbeschrieb der digitalen Schaltung und deren Umgebung
- Liste der Ein- und Ausgänge und Zuteilung der Pins
- Zerlegung der digitalen Schaltung in funktionelle Blöcke
- Schätzung der Grösse der digitalen Schaltung (Anzahl äquivalenter Gatter)
- Schätzung des Verbrauchs
- Liste der kritischen Funktionen Teststrategie, Überprüfbarkeit und Überwachbarkeit der funktionalen Blöcke und Zuteilung der Test-**I**nput **O**utput (I/O)

4.3 Detaillierter Entwurf

4.3.1 Werkzeug

Mit der auf einem **P**rogram **C**ounter (PC) installierten graphisches Interface haben Sie auf alle Werkzeuge Zugriff, die für das Zeichnen des Schaltbilds oder den Beschrieb in **V**ery High Speed Integrated Circuit **H**ardware **D**escription **L**anguage (VHDL) nötig sind.

Falls ein Schaltbild gezeichnet wird, wird die Library der verwendeten Komponenten in der Regel vom Hersteller des Schaltkreises geliefert.



4.3.2 Methodologie

Alle Schaltbilder müssen in derselben Library abgelegt werden. Die Namen der Schaltbilder müssen möglichst mit der ausgeführten Funktion oder dem Namen der Blöcke, denen sie zugehören, in Verbindung gebracht werden können (Abkürzungen).

Die Funktionen werden ausgehend von der tiefsten Ebene ausgeführt. Jede logische Funktion wird mit Hilfe des Tools Modelsim auf unabhängige Art und Weise simuliert.

Für den Zusammenbau der verschiedenen Module eines selben Blocks werden alle Module in Form von Symbolen bearbeitet. Diese Symbole werden anschliessend in einem neuen Schaltbild zusammengefasst, das seinerseits in Symbolform bearbeitet und in eine höhere logische Ebene übertragen wird.

Ein Block kann auch mittels VHDL-Code beschrieben werden. Wenn dieser Block in eine grössere Schaltung integriert werden soll, muss er in Form eines Symbols bearbeitet und in diese höhere Schaltebene übertragen werden. Es wird dasselbe Simulationstool benutzt.

In der Abbildung 2 ist ein Beispiel einer Zerlegung in Baumstruktur dargestellt.

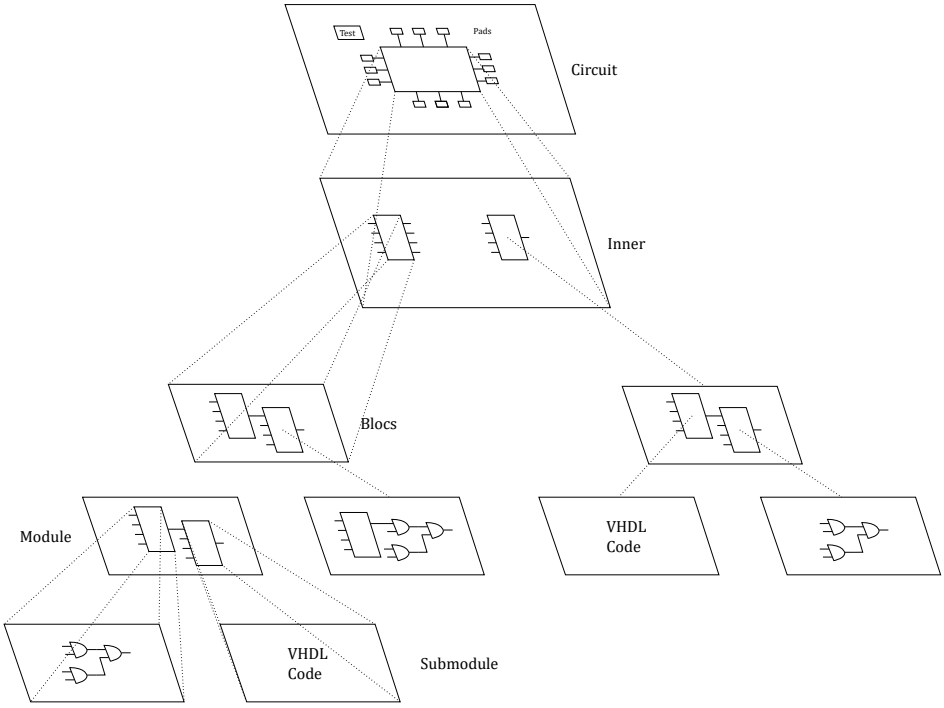
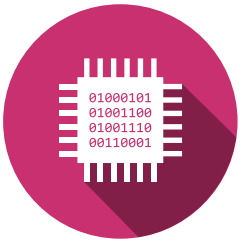


Abbildung 2: Zerteilung in Baumstruktur

4.3.3 Allgemeine Entwurfsregeln

Beim detaillierten Entwurf der Schaltung müssen folgende allgemeine Regeln berücksichtigt werden:

- Form: Ein Schaltbild wird von links nach rechts gelesen. Die Eingänge befinden sich in der Regel auf der linken und die Ausgänge auf der rechten Seite. Aus Gründen der Leserlichkeit ist es manchmal vorzuziehen, einen Ein- oder Ausgang in der Nähe der Gatter oder des Blocks zu belassen, die ihm zugeteilt wurden.
- Name der Signale



- Die Namen der Signale werden in Kleinbuchstaben geschrieben und bestehen ausschliesslich aus Buchstaben und Zahlen. Nur für Busse werden Spezialzeichen verwendet. Bei einem Namen, welches aus mehreren Wörtern entsteht, werden diese mit einem Bindestrich _ zusammengebunden.
- Ein aktiv tiefes Signal erkennt man am Postfix _*n*.

4.4 Synchrone oder asynchrone Systeme

4.4.1 Gatterverzögerungszeit (Gate Delay)

Beim Schaltungsentwurf muss die Gatterverzögerungszeit berücksichtigt werden.

Auf Grund der Durchlaufzeit tritt zwischen einer Veränderung am Eingang und der entsprechenden Anpassung am Ausgang oft eine Verzögerung auf. Diese hängt von der verwendeten Technologie ab und kann innerhalb einer selben Technologie von einem Gatter zum anderen stark variieren. Die Verzögerung hängt ausserdem von der Belastung des Gatters ab.

Auf Grund dieser Verzögerungen kann es vorkommen, dass sich der Ausgang eines Logikblocks während eines kurzen Moments in einem anderen Zustand befindet als der Eingang. Eine solche Zeitstörung entspricht der Situation, in der am Blockausgang zwei fast gleichzeitige Übergänge zwischen zwei identischen, aufeinanderfolgenden Zuständen auftreten.

Solche Zeitstörungen sind auf die Verzögerungszeiten zurückzuführen. Die Veränderung eines Eingangssignals kann den Ausgang auf zwei oder mehreren parallelen Wegen erreichen, die mit jeweils unterschiedlichen Verzögerungen verbunden sind.

4.4.2 Empfehlung: Von Gatterverzögerungszeiten unabhängige Entwurfe

Gatterverzögerungszeiten sind unzuverlässige Systemparameter. Deshalb wird empfohlen, Systeme möglichst so zu entwerfen, dass sie nicht von Gatterverzögerungszeiten betroffen sind. Dazu werden synchrone Logiken verwendet.

Ein synchrones logisches System ist ein System, in dem die Taktsignale aller Kippschaltungen mit demselben Befehlssignal verbunden sind.

4.4.3 Regeln bezüglich der sequentiellen Logik und der Gatterverzögerungen

Asynchrone logische Systeme sind schwierig zu entwickeln und noch schwieriger zu testen und zu beurteilen. Es wird deshalb empfohlen, sich möglichst an die folgenden Regeln zu halten:

4.4.3.1 1. Regel Jede sequentielle Logik muss synchron sein. Verwenden Sie deshalb möglichst einen einzigen externen Taktgeber, der nur auf einer Flanke aktiv ist. Die wichtigsten Elemente dieser sequentiellen Systeme sind das D- oder E-Flipflop.

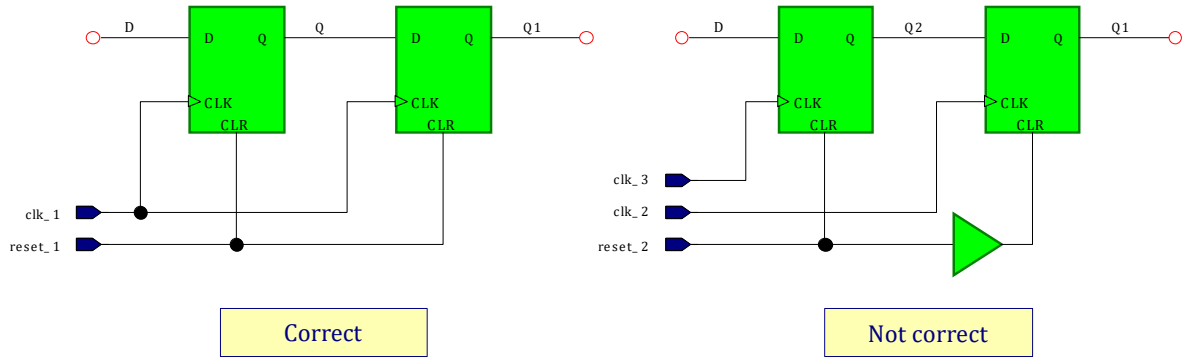


Abbildung 3: Regel 1

4.4.3.2 2. Regel Es ist verboten, die Durchlaufzeit der Logikelemente zu verwenden, um so Impulse zu erzeugen (unterschiedliche Durchlaufzeiten des ursprünglichen Signals und der Verzögerungskette).

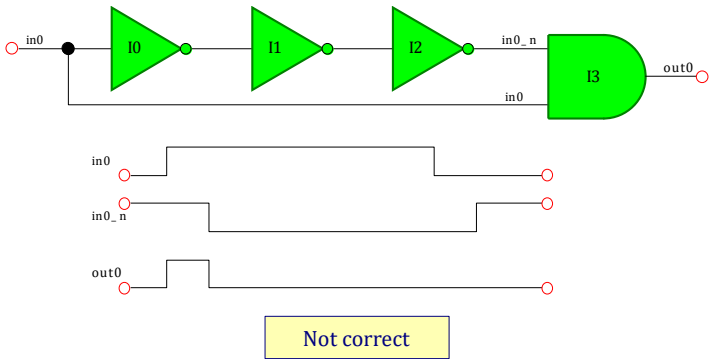


Abbildung 4: Regel 2

4.4.3.3 3. Regel Es ist vorzuziehen, die Taktsignale direkt auf das Flipflop zu leiten, statt diese mittels eines logischen Elements zu kontrollieren, um so jegliche Zeitstörung bei den Eingängen zu vermeiden. Es ist ausserdem nötig, dass die Signale bei den vom Taktgeber synchronisierten Eingängen die Setup- und Haltezeiten in Bezug auf den Taktgeber einhalten.

Bemerkungen: Wenn die Zahl der Gatter oder der Verbrauch möglichst niedrig gehalten werden sollen, kann sich das Einhalten der dritten Regel als unvorteilhaft erweisen.

Asynchrone Frequenzteiler benötigen viel weniger Material als synchrone Frequenzteiler. Wenn sie mit logischen Elementen zusammen benutzt werden, die bei einer höheren Frequenz funktionieren, müssen die Ausgänge der Teiler mit dem höheren Frequenzsignal neu synchronisiert werden.

Einige Ein- oder Ausgangsblöcke bedürfen einer asynchronen Funktionsweise, um rasch auf externe Bedingungen reagieren zu können. In diesem Fall muss der asynchrone Teil des Systems aufs Minimum begrenzt werden.

Es ist wichtig, für die asynchronen Systemteile das Zeitverhalten des Systems intensiv zu analysieren und Funktionsblöcke ohne Gatterverzögerungen zu konzipieren.

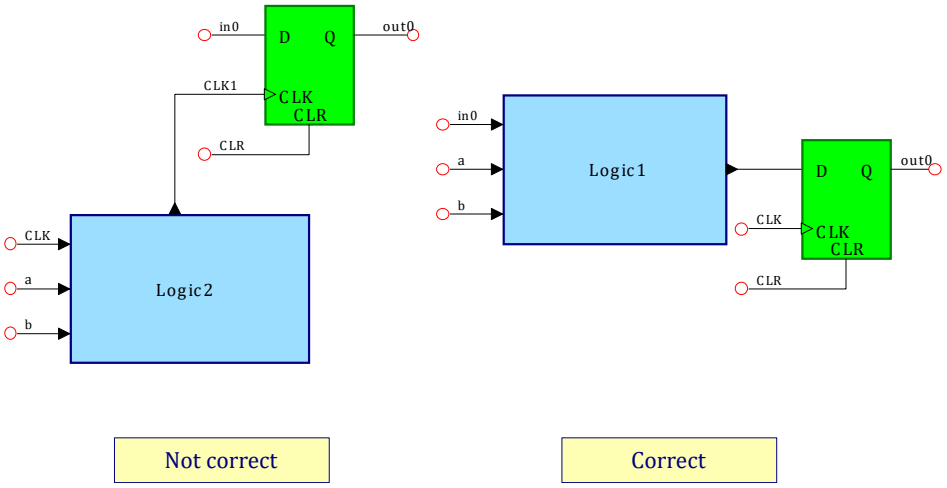


Abbildung 5: Regel 3

4.4.3.4 4. Regel An den asynchronen Eingängen der Flip-Flops dürfen keine Signale mit Zufallswerten eingespeist werden. Die Signale Set und Reset dürfen also nicht aus einer rein kombinatorischen Dekodierung abgeleitet werden.

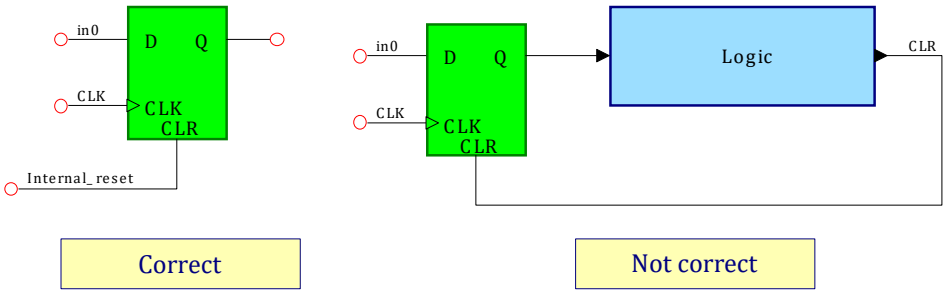


Abbildung 6: Regel 4

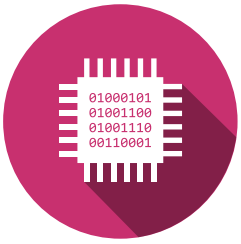
4.4.3.5 5. Regel Jede sequentielle Maschine muss nach dem Einschalten oder zu Beginn einer Simulation in einen bekannten Zustand versetzt werden können. Dazu müssen die asynchronen Set- und Reset-Eingänge der Flipflops verwendet werden. Grundsätzlich sollten diese Eingänge nicht zur Erfüllung der Funktionalität der Schaltung verwendet werden, sondern nur zur Sicherstellung der Testbarkeit.

4.4.3.6 6. Regel Die kürzeste Periode eines Taktgebers eines Synchronrechners wird wie folgt berechnet:

$$T_{min} \leq T_{ClkQ_{max}} + T_{QD_{max}} + T_{skew} - T_{setup_{max}} \tag{1}$$

wobei:

- T_{ClkQ} die Verzögerungszeit zwischen der Taktflanke und dem Flipflop-Ausgang Q ist,
- $T_{QD_{max}} die Verzgerung der lngsten Kette von Gatter nzwischen einem Ausgang Q einer sequentiellen Eingng der sequentiellen Logik ist,$
- T_{setup} die Setup-Zeit der sequentiellen Logik ist.



Bemerkungen: Um die Funktionsgeschwindigkeit einer Schaltung zu erhöhen, können Synchronisationsregister in die grossen Gatterketten eingefügt werden.

4.4.3.7 7. Regel Die Ein- und Ausgangssignale eines Systems müssen mit Hilfe von D-Flipflops synchronisiert werden. Mittels der nachstehenden Abbildung können die Zeitstörungen und metastabilen Zustände, die auftreten, wenn ein Signal von einem asynchronen zu einem synchronen System übergeht, filtriert werden.

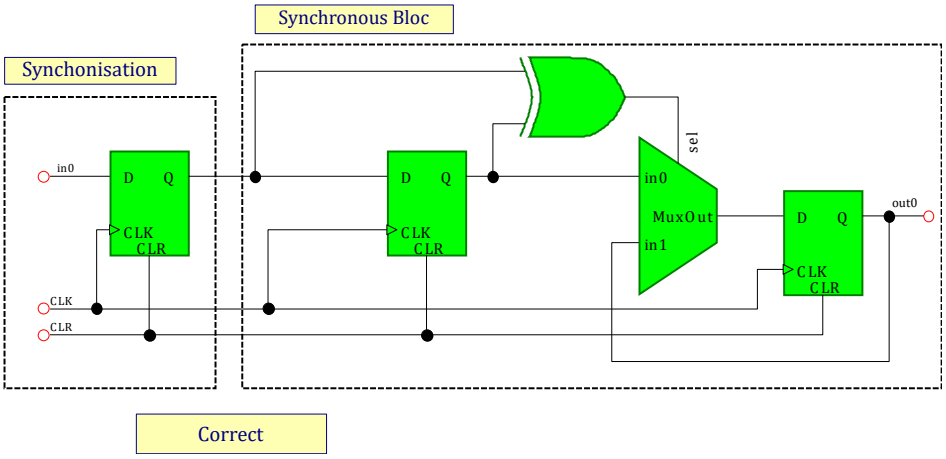


Abbildung 7: Regel 7

4.4.3.8 8. Regel Das Verschwinden des internen Initialisierungssignals der Schaltung muss synchron zum Taktgeber sein; doch sein Erscheinen ist asynchron.

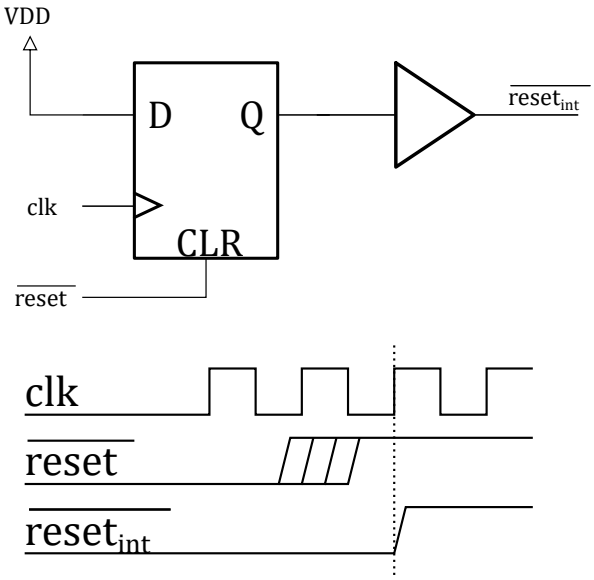


Abbildung 8: Regel 8

4.4.4 Regeln bezüglich das elektrische Verhalten der Logikelemente

4.4.4.1 9. Regel Die Ausgänge der Gatter nicht zu sehr belasten: Für grosse Signale den Fan-Out des Gatters, das sie erzeugt, sowie den Fan-In der Gatter, die dieses steuern muss, einschätzen.

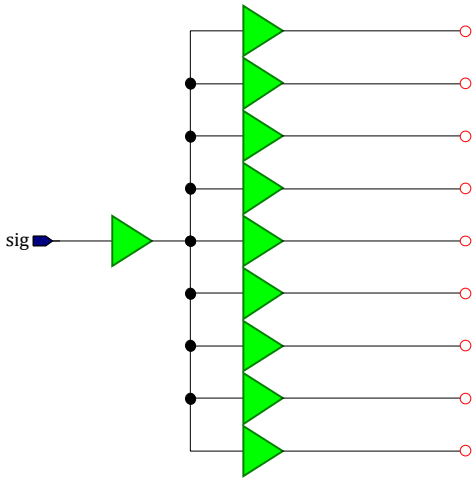
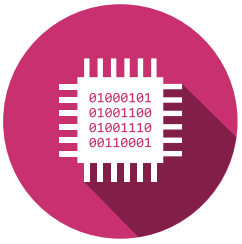


Abbildung 9: Regel 9

4.4.5 Schlussfolgerung

Die beste Lösung für die Herstellung eines komplexen logischen Systems besteht darin, eine synchrone Logik zu verwenden. Mit einer synchronen Logik können zwar nicht so schnelle Informationsbearbeitungszeiten wie mit einer asynchronen Logik erreicht werden, doch sie erlaubt eine grössere Flexibilität beim Entwurf und vor allem eine genauere Funktionsanalyse des Systems.



5 Verifikations- und Validationsphase

Über einen Punkt sind sich alle Studien einig: Ein Fehler kostet umso mehr, desto später er im Lebenszyklus entdeckt wird. Es ist deshalb sehr wichtig, alles daran zu setzen, um Fehler so früh wie möglich aufzuspüren. Im Idealfall sollte ein Fehler sofort nach Abschluss der Etappe, in der er aufgetreten ist, entdeckt und korrigiert werden. Das Aufspüren und Korrigieren von Fehlern wird Verifikation und Validation genannt.

5.1 Validation

Bei der Validation wird überprüft, ob das Produkt, das entwickelt wird, dem vom Kunden erwarteten Produkt entspricht („Stellen wir DAS korrekte Produkt her?“). Die Präsentation von Prototypen während der Spezifikationsphase, gemeinsame Prüfungen oder die Validation durch den Kunden sind Beispiele für Validationstechniken.

5.2 Verifikation

Bei der Verifikation wird sichergestellt, dass die Produkte am Ende einer Entwicklungsetappe den Produkten und Standards des Etappenbeginns entsprechen („Stellen wir ein korrektes Produkt her?“). Prüfungen, Verbesserungen und Inspektionen der Schaltbilder oder des Codes sind einige Beispiele für Verifikationstechniken. Bei der Verifikation ist die Anwesenheit des Kunden meist nicht erforderlich.

5.3 Validations- und Verifikationstechniken

Es existieren zahlreiche Verifikations- und Validationstechniken. Einige davon sind statisch, da sie sich auf Dokumente beziehen (vgl. Prüfung). Andere sind dynamisch, weil sie auf der Simulation eines Teils oder des gesamten VHDL-Codes basieren (vgl. Teilschaltungstest).

Nachdem die Vorgehensweise für die Entwicklung definiert wurde, muss deshalb für jede definierte Etappe folgende Frage gestellt werden: „Was können wir tun, um sicherzustellen, dass die Endprodukte dieser Etappe einerseits den Bedürfnissen des Kunden entsprechen (Validation) und andererseits in Bezug auf die Zielsetzungen der Etappe und die Produkte bei Etappenbeginn zutreffend und kohärent sind (Verifikation)?“.

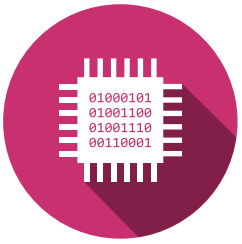
Alle geplanten Validations- und Verifikationsaktivitäten müssen bei der Projektplanung mit einbezogen werden.

Für kleinere Projekte muss für die **Verifikation** mindestens Folgendes vorgesehen werden:

- Simulationen der entwickelten VHDL- oder Schaltbildfunktionen
- Prüfung der Entwurfsdokumentation durch fachkompetente Auditoren
- Gegenseitiges Korrekturlesen des VHDL-Codes durch die verschiedenen Teammitglieder

Für die **Validation** muss mindestens Folgendes vorgesehen werden:

- Die durchgeführte/n Simulation/en für die Validation des Designs der gesamten Schaltung. Dazu muss anhand der Spezifikationen eine Konformitätsmatrix ausgearbeitet werden, welche die gesamten Punkte enthält, die geprüft werden müssen. Für jeden Punkt muss angegeben werden, mittels welcher Simulation und an welcher Stelle die Verifikation durchgeführt wurde. Auf der nächsten Seite ist ein Beispiel einer solchen Matrix abgebildet.

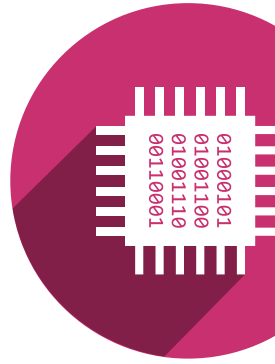


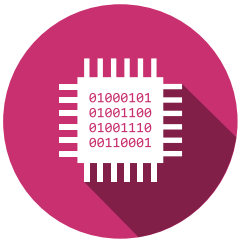
- Eine gemeinsame Prüfung der Spezifikationsdokumentation durch fachkompetente Auditoren sowie durch Vertreter jeder Kundenkategorie (zukünftige Anwender, Entscheidungsträger usw.).
- Interne (teaminterne) und externe (Validation durch den Kunden) Validationstests.

Weitere empfohlene Techniken: Herstellung von Prototypen in der Spezifikationsphase (Validation), Analyse des Codes mittels dedizierter Tools usw. Von den gesamten Validations- und Verifikationstechniken sind die Simulations- und Prüftechniken die effizientesten und die am meisten verwendetsten.

Identifikationsnummer der Funktionsanforderung	Beschrieb	Testbench & Simulation Setup	Zeitpunkt der Validation	Validationsmethode	Validationsstand	Beilagen Seite
(1)	(2)	(3)	(4)	(5)	(6)	(7)

- (1) Jede Spezifikation wird mittels einer einmaligen Nummer identifiziert: DSxx-YYXXX
  - DSxx ist die Nummer des Spezifikationsdokuments (DS für Design Specification)
  - YY ist die Nummer des Kapitels (1, 2, 3, ...,11,...)
  - XXX ist die Nummer der Funktionsanforderung (005, 010, 015, 020, 021, 030, 035, ...)
- (2) Schlüsselwörter der Funktionsanforderung, die validiert werden muss.
- (3) Identifikation der verwendeten Testbench sowie des Simulation Setups.
- (4) Zeitpunkt der Validation der Anforderung im Simulationsablauf.
- (5) Validationsmethode (Analyse, visuelle Prüfung, Druck, Ergebnisdatei, automatische Validation, ...)
- (6) Validationsstand (OK, Not OK, gemessener Wert)
- (7) Nummer der Beilage (bei Bedarf)





6 Integrationphase

In dieser Phase werden alle Etappen des rechten Teils des V-Diagramms vom detaillierten Entwurf bis zum Projektabschluss zusammengefasst. Das Entwicklungsteam muss diese Phase so organisieren, dass die Aufeinanderfolge Verifikation und Validation der Teilschaltungen - Integration - Systemvalidation eingehalten werden kann. Nur so können die Konformität und Qualität des Produkts gewährleistet werden.

Diese Phase verläuft umso einwandfreier, desto besser die vorgängigen Phasen durchgeführt wurden. Wenn in den Spezifikations- und Entwurfsphasen Verifikations-, Validations- und Integrationsmechanismen vorgesehen wurden, ist diese Sequenz eine Routinearbeit mit relativ geringem Zeitaufwand.

6.1 Resultate

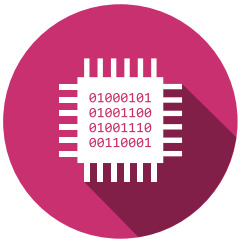
In dieser Phase werden nicht nur Berichte verfasst, sondern es müssen auch Quellen und Executables gefunden werden. Viele Elemente müssen deshalb auf Datenträgern geliefert werden. Folgende Elemente werden in Form von Dossiers geliefert:

- Ein detaillierter Entwurfsbericht mit einem detaillierten Beschrieb der Entwurfsaktivitäten, der Codierung und Verifikation, der Validationen der Teilschaltungen sowie der Integration und Endvalidation. Der Schwerpunkt muss dabei auf der Gewissenhaftigkeit der durchgeführten Validationen sowie auf den Elementen liegen, die es einem anderen Team ermöglichen, das Projekt zu einem späteren Zeitpunkt weiterzuführen.
- Benutzerhandbuch. Dabei handelt es sich um das Dokument, das dem Kunden (Endanwender) geliefert wird. Es muss die Installation, die Einstellungen und die Benutzung des Materials beschreiben. Dieses Dokument wird eigentlich im Laufe des Projekts verfasst, da es das Modell übernimmt, das während der allgemeinen Entwurfsphase erarbeitet wurde (das wiederum auf der funktionalen Beschreibung des Produkts basiert, die während der Spezifikationsphase erstellt wurde).

6.2 Verwendete Mittel

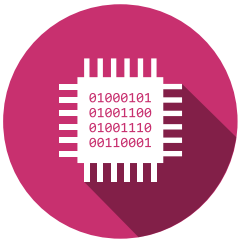
Während dieser Integrationsphase arbeitet das Team in der Regel vollzeitlich. Neben der Beherrschung der technischen Mittel treten in dieser Zeit oft zwei Arten von Problemen auf:

- Terminprobleme: Die im Rahmen der vorhergehenden Phasen akkumulierten Verzögerungen werden unweigerlich spürbar, sobald der Termin näherrückt. Die Teilnehmer geraten deshalb oft in Panik, statt sich auf die Qualität und die Organisation zu konzentrieren. Es gibt nur eine Lösung: VORAUSSICHT, VORAUSSICHT, VORAUSSICHT. Ob der Termin eingehalten wird, entscheidet sich schon während der Spezifikationsphase. Die Entwickler und insbesondere die Projektleiter müssen vor allem in der Anfangsphase sehr vorsichtig sein (die zwei Wochen, die am Ende fehlen, werden oft schon zu Beginn verloren). Durch ein gewissenhaftes Projektmonitoring können Katastrophen zudem vermieden werden, indem Zielsetzungen rechtzeitig neu definiert werden.
- Die Verwaltung der Arbeit der Gruppe: Viele Überlegungen der vorhergehenden Phasen können und/oder müssen gemeinsam in der Gruppe durchgeführt werden. Diese Phase jedoch zeichnet sich durch viele individuelle Arbeiten aus. Die Arbeit kann nur erledigt werden, wenn sie parallel dazu verwaltet wird. Doch die einfache Addition individueller Arbeiten führt nicht automatisch zu deren Summe. Dazu braucht es Organisation. Deshalb



muss eine Struktur für die Aufteilung der Aufgaben auf die Teammitglieder und die für die Synchronisation benötigte Zeit aufgestellt werden. Je umfangreicher die Gruppe, desto präziser muss die Organisation sein und desto länger dauert die Synchronisation. Diese Zeit muss bei der Planung ausdrücklich berücksichtigt werden. Für eine 5- bis 6-köpfige Gruppe, die vollzeitlich an einem Projekt arbeitet, scheinen eine tägliche kurze Lagebesprechung (Kaffeepause) sowie eine strukturierte wöchentliche Sitzung eine realistische Lösung zu sein.

Die Beherrschung der Technik ist ebenfalls ein wichtiges Element: Jeder Entwickler muss alle im Rahmen des Projekts verwendeten Tools beherrschen können (Entwicklungs-, Dokumentations-, Projektmanagementtools). Die Integrationsphase ist keine Lernphase. Wenn neue Techniken benutzt wurden, wie dies bei fast allen Projekten der Fall ist, müssen parallel zur Entwicklung der einzelnen Phasen Lernphasen stattfinden. Wird ein Tool nicht beherrscht, ist es in den meisten Fällen besser, von dessen Benutzung abzusehen und vielleicht weniger spektakuläre, aber dafür bekannte Techniken anzuwenden.



Akronyme

FIFO **F**irst **I**n **F**irst **O**ut. 7

I/O **I**nput **O**utput. 7

PC **P**rogram **C**ounter. 7

RAM **R**andom **A**ccess **M**emory. 7

VHDL **V**ery High Speed Integrated Circuit **H**ardware **D**escription **L**anguage. 7, 8