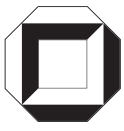
Andreas Vörg

Automatisierte Qualifizierung und Auslieferung wiederverwendbarer Komponenten



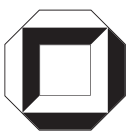
universitätsverlag karlsruhe

Andreas Vörg

Automatisierte Qualifizierung und Auslieferung wiederverwendbarer Komponenten

Automatisierte Qualifizierung und Auslieferung wiederverwendbarer Komponenten

von
Andreas Vörg



universitätsverlag karlsruhe

Dissertation, Eberhard-Karls-Universität Tübingen,
Fakultät für Informations- und Kognitionswissenschaften, 2005
Dekan: Prof. Dr. Michael Diehl
1. Berichterstatter: Prof. Dr. Wolfgang Rosenstiel
2. Berichterstatter: Prof. Dr. Dietmar Müller (Technische Universität Chemnitz)

Impressum

Universitätsverlag Karlsruhe
c/o Universitätsbibliothek
Straße am Forum 2
D-76131 Karlsruhe
www.uvka.de

Universitätsverlag Karlsruhe 2005
Print on Demand

ISBN 3-937300-84-8

Danksagung

Diese Arbeit entstand während meiner Tätigkeiten als wissenschaftlicher Mitarbeiter im Forschungsbereich „Systementwurf in der Mikroelektronik“ am FZI Forschungszentrum Informatik an der Universität Karlsruhe.

An dieser Stelle möchte ich Allen danken, die sich in dieser Zeit auf vielfältige Weise an der Entstehung dieser Arbeit beteiligt haben.

Mein besonderer Dank gilt Herrn Prof. Dr. Wolfgang Rosenstiel für die Betreuung und Unterstützung während meiner Arbeit, für seine wertvollen Hinweise und die Übernahme des Gutachtens. Ebenso danke ich Herrn Prof. Dr. Dietmar Müller für die Erstellung des Gutachtens.

Danken möchte ich auch allen meinen Kollegen für die Diskussionen, Hinweise, Vorschläge und Korrekturen. Dies alles hat zum Gelingen dieser Arbeit beigetragen.

Ohne die finanzielle Unterstützung des Projekts „Intellectual Property Qualifizierung“ (IPQ - 01M3048) und des europäischen MEDEA+ Projekts „Tools and Methods for Intellectual Property“ (ToolIP - A511) durch das Bundesministerium für Bildung und Forschung (BMBF) wäre meine Arbeit nicht möglich gewesen. Ich bedanke mich recht herzlich dafür.

Ein ganz besonderer Dank gilt meiner Frau für ihr Verständnis und ihre besondere Unterstützung in diesem Lebensabschnitt. Schließlich danke ich meinen Eltern, die mich stets bestärkt und unterstützt und mir so meinen Weg ermöglicht haben.

Hannover im Juni 2005

Andreas Vörg

Inhaltsverzeichnis

Kurzfassung

Aktuelle und zukünftige Chipentwürfe werden aus wiederverwendbaren Basisbausteinen, so genannten IP-Modulen, aufgebaut. Diese IP-Module müssen definierte Qualitätskriterien erfüllen, um den Chipentwurf zu beschleunigen, das Integrationsrisiko aufseiten des IP-Nutzers und den Aufwand für Nacharbeit aufseiten des IP-Anbieters zu reduzieren. Die Übereinstimmung des IP-Modulentwurfs mit diesen Kriterien muss automatisch geprüft werden, da durch das exponentielle Komplexitätswachstum manuelle Prüfungen nicht mehr durchführbar sind. Qualitätsmängel und die Bindung von Ressourcen, die durch einen konventionellen Austauschprozess nach der Qualifizierung durch den IP-Anbieter entstehen, können durch einen automatisierten, qualitätserhaltenden IP-Austauschprozess verhindert beziehungsweise wesentlich reduziert werden.

Die vorliegende Arbeit stellt eine automatisierte Qualifizierungs- und Auslieferungsmethode anhand applikationsunspezifischer Standardqualitätskriterien für digitale Soft-IP-Module vor. Die Automatisierung basiert auf einem einheitlichen Qualifizierungs- und Austauschformat. Durch die vorgestellten Methoden wird der gesamte Wiederverwendungsprozess vorhersehbarer und robuster. Einerseits bleiben qualifizierte Merkmale erhalten und andererseits dürfen mehr als 80 % Zeit- und Ressourceneinsparungen erwartet werden.

Schlüsselwörter: Intellectual Property; IP; Virtual Component; VC; VSIA; QIP; Quality IP; Auslieferung; Export; Import; Qualität; Qualifizierung; Automatisierung; Austauschformat; Qualitätsbewertung

1 Einleitung

Die Mikroelektronik steht an der Schwelle zur Nanoelektronik. Erste Produkte in 90-nm-Technologie, wie beispielsweise AMDs Opteron-Prozessor, der seit dem Frühjahr 2004 in Dresden gefertigt wird, dokumentieren den Übergang zur Nanoelektronik. SEMATECH prognostiziert in der „International Technology Roadmap for Semiconductors" (ITRS) im Jahr 2007 den 65-nm-Technologieknoten zu erreichen und 2010 den 45-nm-Knoten [54]. Aktuellere Quellen kündigen erste Produkte in 65-nm-Technologie sogar schon für 2006 an. So soll beispielsweise die 65-Nanometer-Fertigung von AMD64-Prozessoren 2006 in Dresden beginnen [78].

Dadurch wird einmal mehr der ungebrochen anhaltende Trend in der Halbleiterfertigung deutlich gemacht. Mit geringen Schwankungen stehen alle 18 Monate neue Fertigungstechnologien bereit, die elektronische Schaltungen mit sinkender Strukturgröße auf wachsender Chipfläche und Waferfläche ermöglichen. Bereits 1965 hat Gordon Moore diesen Trend vorausgesagt, der auch als „Mooresches Gesetz" bekannt ist. Seitdem folgt die Halbleiterforschung und -industrie diesem Gesetz ([26], [27]). Das Mooresche Gesetz zählt zu den wenigen Dingen, die sich in diesem von Veränderungen geprägten Umfeld nicht verändert hat. Es hat immer noch seine Gültigkeit.

In der Vergangenheit haben technologische Möglichkeiten Halbleiterprodukte in ihrer Realisierbarkeit begrenzt. Inzwischen ist die Entwurfsproduktivität das begrenzende Glied in der Herstellungskette. Laut einer SEMATECH-Studie wächst die Entwurfsleistung mit lediglich 21 %. Ihr steht eine Zunahme der technologisch realisierbaren Komplexität von ca. 58 % gegenüber [28]. Die Komplexität ist durch die eingangs beschriebenen Fortschritte in der Fertigungstechnologie bedingt. Die Lücke zwischen Entwurfs- und Komplexitätswachstum wird Produktivitätslücke (engl. productivity gap) genannt. Sie zu schließen ist die Herausforderung an die Forschung in der elektronischen Entwurfsautomatisierung (engl. Electronic Design Automation, EDA).

Die exponentiell wachsende Zahl an integrierbaren Transistoren (Integrationsdichte) ermöglicht den Entwurf kompletter elektronischer Systeme auf einem einzigen Chip. Solche Systeme werden auch System-auf-einem-Chip (engl. System-on-a-Chip, SoC) genannt. Aufgrund der geschilderten Wachstumsverhältnisse ist es gegenwärtig eine Herausforderung, die geplante Produkteinführungszeit (engl. Time-to-Market) bei immer kürzer werdenden Entwicklungszyklen einzuhalten. In der Zukunft wird es sogar unmöglich sein, einen Entwurf erfolgreich abzuschließen, wenn nicht neue Entwurfsmethoden angewendet werden [29].

1.1 Ausgangssituation

Aus diesem Grund wurde in den 90er Jahren die Methode der Wiederverwendung zur Steigerung der Entwurfsproduktivität eingeführt. In [57] wird sogar eine Halbierung der Entwurfskosten durch Wiederverwendung

prognostiziert. Das Neue an dieser Art der Wiederverwendung ist die Abstraktionsebene, auf der Funktionalität gekapselt und wieder verwendet wird. Seit jeher war Abstraktion ein erfolgreiches Mittel zur Beherrschung der wachsenden Komplexität. Transistoren wurden durch logische Gatter abstrahiert und logische Gatter durch arithmetische Funktionen (Addierer, Multiplizierer, Multiplexer usw.). Mit jedem Abstraktionsschritt konnte die Erzeugung der darunter liegenden (weniger abstrakten) Ansichten zunehmend EDA-Werkzeugen überlassen werden. Dadurch konnte jeweils eine Steigerung der Entwurfsproduktivität erreicht werden.

Die in der vorgelegten Arbeit behandelte Wiederverwendungsmethode schlägt die Erzeugung elektronischer Modulentwürfe (Prozessoren, Mikrokontroller, Standardschnittstellen (PCI, USB, …) usw.) vor, um diese als Grundbausteine in mehreren unterschiedlichen SoC-Entwürfen wieder zu verwenden [44]. Wiederverwendbare Module werden auch Intellectual Property (IP) genannt. Durch die Vermeidung des Neuentwurfs und einer kontinuierlichen Verbesserung evtl. enthaltener Entwurfsfehler reduziert diese Methode die Komplexität und damit das Entwurfsrisiko. Darüber hinaus unterstützt sie die Einhaltung der geplanten Produkteinführungszeit. Ein SoC kann sehr viel schneller aus bereits verifizierten IP-Modulen aufgebaut werden, als es in einer kompletten Neuentwicklung möglich ist. Im Vergleich zu einem einmaligen Auftragsprojekt für ein Funktionsmodul erhöhen wiederverwendbare IP-Module darüber hinaus den Kapitalertrag (engl. Return on Investment, ROT). Der Grund dafür liegt in der Wiederverwendbarkeit des IP-Entwurfs in verschiedenen Projekten innerhalb der eigenen Firma aber auch durch einen externen IP-Nutzer.

Abbildung 1 zeigt, dass, neben den Vorteilen des IP getriebenen SoC-Entwurfs, Wiederverwendung sogar absolut notwendig ist, um ein SoC-Projekt erfolgreich zu beenden [29]. In Abbildung 1 geht die ITRS von einer 42 prozentigen Verbesserung der Entwurfsproduktivität aus. Die ITRS bezeichnet dieses Wachstum selbst als unzureichend. Ihre Prognose geht davon aus, dass ein erfolgreicher SoC-Entwurf innerhalb eines Zeitrahmens von zehn Personenjahren fertig gestellt sein muss, dass so viel wie möglich neue Logik im SoC-Entwurf integriert werden soll und dass wiederverwendbare Logik doppelt so schnell integriert werden kann wie neue Logik. Die Ressourcen für Speichergenerierung werden wegen der Verwendung von Speichergeneratoren vernachlässigt. Unter diesen Umständen ist ein erfolgreicher SoC-Entwurf im Jahr 2010 nur möglich, wenn über 80 % der verfügbaren Chipfläche von Speicher belegt sind (grau hinterlegter Bereich in Abbildung 1 auf Seite 3).

Die Entwurfsproduktivität muss um 100 % verbessert werden, um auch in Zukunft erfolgreiche SoC-Projekte durchführen zu können. Die Voraussetzung für eine solche Verbesserung der Entwurfsproduktivität ist der Einsatz eines hohen Anteils (zwischen sechs und 82 %) wiederverwendbarer Logik. Dieser Anteil ist notwendig, wenn neue Logik maximiert werden und weniger als die Hälfte des SoC von Speicher belegt sein soll. Abbildung 2 auf Seite 3 zeigt, dass dadurch erfolgreiche SoC-Projekte bis ins Jahr 2016 ermöglicht werden.

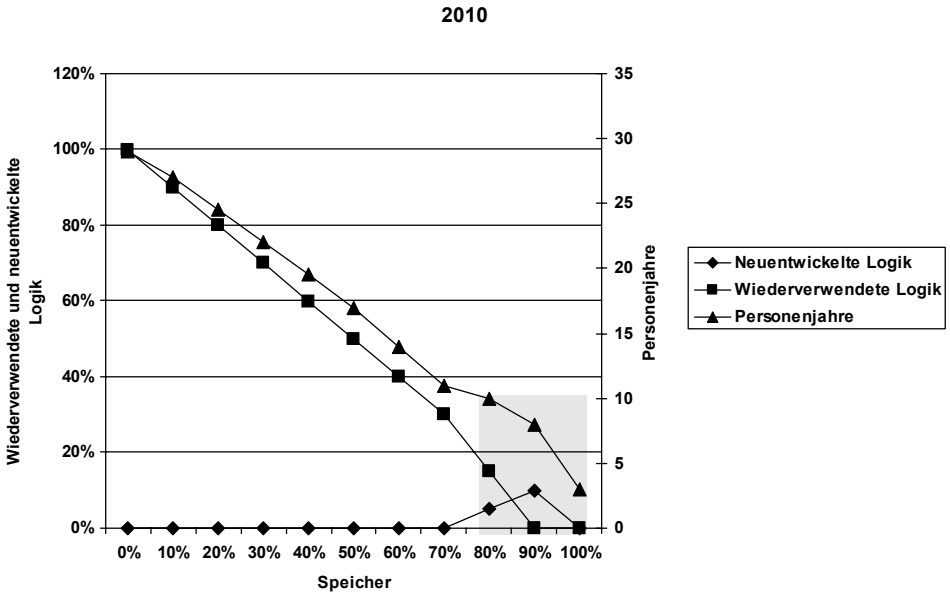


Abbildung 1: Ungenügende 42 prozentige Verbesserung der Entwurfsproduktivität für SoC im Jahre 2010 nach ITRS 2001

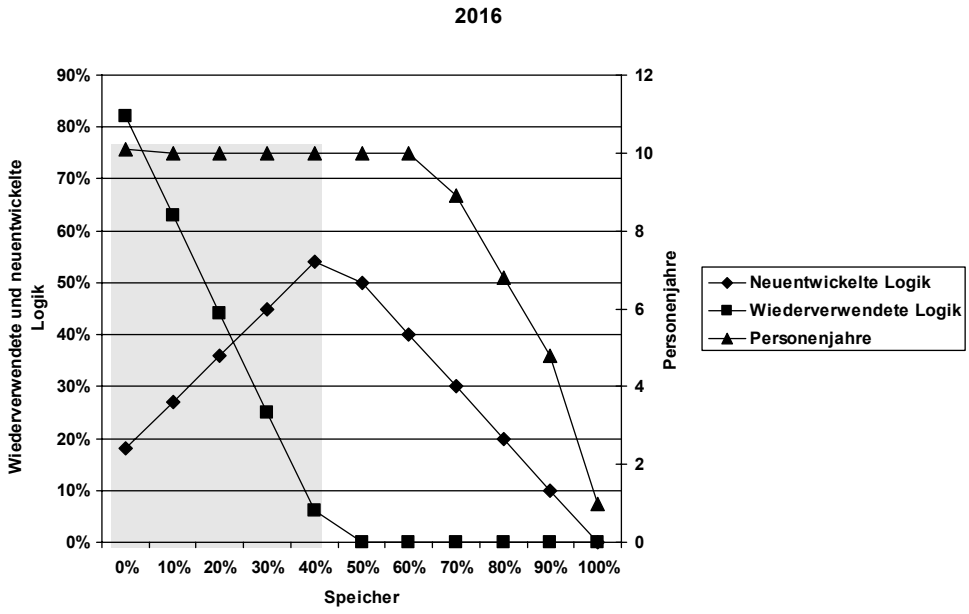


Abbildung 2: Ausreichende 100 prozentige Verbesserung der Entwurfsproduktivität für SoC im Jahre 2016 nach ITRS 2001

IP-Module werden von IP-Anbietern unabhängig vom Anwendungskontext entwickelt, um ihren Wiederverwendungsgrad zu erhöhen. Sie werden dann in einer möglichst großen Zahl von applikationsspezifischen Projekten wieder verwendet. In Deutschland und Europa haben sich Kernkompetenzen für Automotive-, Telekommunikation- und zunehmend auch Multimediaapplikationen etabliert. Seit der Einführung des IP-basierten

Entwurfs hat sich ein wachsender Markt für IP-Module entwickelt. Es werden beispielsweise Prozessoren, Mikrokontroller, digitale Signalprozessoren, Peripheriekomponenten, Komprimierungs- und Dekomprimierungskomponenten, kryptografische und multimediale Verschlüsselungs- und Entschlüsselungskomponenten angeboten und wieder verwendet. Die Wiederverwendbarkeit muss sowohl durch eine sorgfältige Marktanalyse vor der Entwicklung eines IP-Moduls als auch durch die Übereinstimmung des Entwurfs mit Wiederverwendbarkeitskriterien während und bei Abschluss der Entwicklung eines IP-Moduls sichergestellt werden.

Häufig werden IP-Module nicht nur intern (in der eigenen Firma zwischen Abteilungen), sondern auch mit externen (über Firmengrenzen hinweg) IP-Nutzern ausgetauscht. Dabei hat sich gezeigt, dass insbesondere die externe Wiederverwendung nicht die gewünschte Beschleunigung beim Entwurf komplexer Systeme gebracht hat. Die Analyse der Wiederverwendungsprozesse in Forschung und Industrie hat gezeigt, dass die Schnittstelle zwischen IP-Anbieter und IP-Nutzer nur unzureichend definiert ist. Unter Schnittstelle ist die Qualität des IP-Moduls zu verstehen. Eine ausreichend definierte, aber proprietäre Schnittstellenvereinbarung kann den Nutzen bei weitem übersteigen [32] und der Aufwand zur Einhaltung der geforderten Qualität kann mit enormen Kosten verbunden sein [77]. Insbesondere muss die Extraktion aus der Entwicklungsumgebung des IP-Anbieters beziehungsweise die Integration in die Entwicklungsumgebungen des IP-Nutzers unterstützt werden. Dabei spielt die Gewährleistung der Qualität eine enorme Rolle.

1.2 Ziel der Arbeit

Das Ziel der vorliegenden Arbeit ist es, die Schnittstelle zwischen IP-Anbieter und IP-Nutzer überprüfbar zu gestalten und dadurch die Erhaltung standardisierter Qualitätsmerkmale aus dem Entwurfsablauf des IP-Anbieters in den SoC-Integrationsprozess des IP-Nutzers zu garantieren. Anhand quantitativer Ergebnisse wird belegt werden, dass die Nutzung der vorgeschlagenen Methoden eine Verbesserung der Wiederverwendbarkeit und eine Verringerung des Integrationsrisikos zur Folge hat. Dieses Ziel wird mit automatisierten Qualifizierungs- und Auslieferungsmethoden anhand von Standardqualitätskriterien für digitale Soft-IP-Module erreicht. Die Automatisierung basiert auf einem einheitlichen Qualifizierungs- und Austauschformat. Durch die vorgestellten Methoden wird der gesamte Wiederverwendungsprozess vorhersehbarer und robuster. Einerseits bleiben qualifizierte Merkmale erhalten und andererseits dürfen mehr als 80 % Zeit- und Ressourceneinsparungen erwartet werden.

Es wird gezeigt, dass die Einhaltung von Qualitätsmerkmalen das Integrationsrisiko verringert und den Wiederverwendungsprozess beschleunigt. Das kann aber nur erreicht werden, wenn die Einhaltung dieser Merkmale mit vertretbarem Aufwand sowohl vom IP-Anbieter als auch vom IP-Nutzer überprüft werden kann. Des Weiteren muss die Einhaltung der Merkmale auch nach der IP-Übermittlung vom IP-Anbieter an den IP-Nutzer sichergestellt sein. Dazu wird eine Austauschmethode erforscht, durch die

die Erhaltung dieser Merkmale bis in den SoC-Entwurf gewährleistet wird. Nur so wird es möglich sein, die von der ITRS prognostizierte Zahl an IP-Modulen bereitzustellen und diese in kalkulierbarer Zeit mit vorhersehbarem Risiko zu integrieren.

1.3 Aufbau der Arbeit

Im folgenden Kapitel 2 werden die Grundlagen für die vorliegende Arbeit gelegt. Benötigte Begriffe werden definiert und die relevanten Entwurfsmethoden vorgestellt.

Themenverwandte Arbeiten, Werkzeuge und Organisationen, die an der IP-Qualifizierung und Standardisierung beteiligt sind, werden in Kapitel 3 vorgestellt, um den aktuellen Stand der Technik aufzuzeigen.

Der Hauptteil dieser Arbeit wird in den Kapiteln 4 und 5 dargelegt. Es wird die IP-Qualifizierungsmethode zur automatischen Überprüfung von Standardqualitätskriterien beschrieben und wie derartig qualifizierte IP-Module ohne Qualitätsverlust automatisiert an einen IP-Anbieter übertragen und in den Chipentwurfsablauf integriert werden können.

Das einheitliche Qualifizierungs- und Austauschformat sowie die Implementierung der vorgestellten Methoden wird in Kapitel 6 beschrieben.

In Kapitel 7 werden die ermittelten Ergebnisse diskutiert, um die Tauglichkeit der vorgestellten Methoden zu verdeutlichen. Mit einer Zusammenfassung und einem kurzen Ausblick auf mögliche Anknüpfungspunkte und Anwendungsmöglichkeiten schließt Kapitel 8.

2 Grundlagen

In den nachfolgenden Kapiteln wird ausführlich auf die Qualifizierung und den Austausch von IP-Modulen eingegangen. Doch zuvor werden in diesem Kapitel die zum Verständnis benötigten Grundlagen und Definitionen dargestellt. Verwendete Begriffe werden erläutert und es wird darauf eingegangen, in welchem Umfeld IP-Module verwendet werden und auf welche IP-Module die vorliegende Arbeit angewendet werden kann.

Die in der vorliegenden Arbeit verwendeten Konventionen werden in Abschnitt 2.1 erläutert.

Ausgehend von den Bedingungen unter denen elektronische Schaltungen entworfen werden, wird in Abschnitt 2.2 auf die Möglichkeit des SoC-Entwurfs eingegangen.

Die Zusammenhänge zwischen IP-basierter und plattformbasierter Entwurfsmethodik werden in Abschnitt 2.3 dargelegt. Anschließend werden die aus der Literatur bekannten IP-Härtegrade in die gewohnten Abstraktionsebenen in Abschnitt 2.4 eingeordnet.

Schließlich definiert Abschnitt 2.5 die verwendeten Qualitätsbegriffe.

2.1 Konventionen

Im Verlauf dieser Arbeit werden die folgenden Konventionen verwendet.

Konvention	Steht für
<Variablenname> oder $Variablenname	Variablennamen sind in spitzen Klammern eingeschlossen oder es ist ein $-Zeichen vorangestellt. Des Weiteren gelten die Konventionen zur Darstellung von Variablen des jeweils dargestellten Datenformats. Beispielsweise werden in XML Tags in spitzen Klammern codiert. In diesem Fall hat die spitze Klammer also nicht die hier erläuterte Sonderbedeutung.
`Quellcode`	Quellcode ist in der Schriftart Courier New gesetzt.
Dateinamen	Verzeichnis- und Dateinamen sind in der Schriftart Courier New und kursiv gesetzt.
Wichtige Begriffe	Wichtige Begriffe werden bei ihrer ersten Verwendung und Erklärung durch Fettdruck hervorgehoben.

Tabelle 1: Verwendete Konventionen

Verwendete Akronyme und Abkürzungen werden im Anhang auf Seite 125 erläutert.

2.2 Entwurf elektronischer Schaltungen

Die Entwicklung der Entwurfsmethoden für elektronische Schaltungen wird im Wesentlichen durch die exponentiell wachsende Integrationsdichte getrieben.

2.2.1 Integrationsdichte

Die Integrationsdichte bezeichnet die Anzahl Transistoren eines Mikrochips pro Flächeneinheit. Gemäß dem Mooreschen Gesetz verdoppelt sich die Integrationsdichte etwa alle 18 Monate, was seit über 35 Jahren mit kleinen Schwankungen zutrifft.

So enthielt 1999 Intels erster Pentium-III mit 500 MHz Taktfrequenz noch einen externen L2-Cache und bestand aus etwa 9,5 Millionen Transistoren. Nur etwa 1½ Jahre später arbeitete der Pentium 4 bereits mit 1,4 GHz Taktfrequenz, einer 20-stufige Pipeline, wurde in 130-nm-Technologie gefertigt und bestand aus etwa 42 Millionen Transistoren.

Das Wachstum der Integrationsdichte ist durch die Verkleinerung der physikalischen Strukturen auf dem Chip bedingt. Derzeit überschreitet die Mikroelektronik die Schwelle zur Nanoelektronik mit ersten Produkten in 90-nm-Technologie. Als Beispiel sei AMDs Opteron-Prozessor genannt, der seit dem Frühjahr 2004 in Dresden gefertigt wird. Die International Technology Roadmap for Semiconductors (ITRS) rechnet 2007 damit den 65-nm-Technologieknoten zu erreichen und 2010 den 45-nm-Knoten [54].

Diese Beispiele zeigen, dass in absehbarerer Zeit nicht mit einer Abflachung oder gar Konvergenz des exponentiellen Integrationsdichtewachstums zu rechnen ist. Die fortschreitende Verkleinerung der physikalischen Strukturen (Leiterbahnen, Transistoren usw.) ermöglicht die Fertigung eines kompletten Systems-auf-einem-Chip (engl. System-on-a-Chip, SoC).

2.2.2 System-auf-einem-Chip

In der Vergangenheit musste die Funktionalität eines Systems auf mehrere Chips verteilt werden. Diese Chips wurden anschließend auf Leiterplatten zu einem System integriert. Aber schon heute ist aufgrund der wachsenden Integrationsdichte die Funktionalität eines gesamten Systems auf einem einzigen Chip integrierbar. Mit Erreichen des 45-nm-Knotens werden fast eine Milliarde Transistoren auf einem einzigen Chip zur Verfügung stehen. Die hieraus resultierenden Parameter, welche die Systemfunktion beeinflussen, können beim Entwurf unmöglich einzeln berücksichtigt werden. Andererseits soll das Potenzial eines Chips ausgeschöpft werden. Das bedeutet, die Entwurfskomplexität muss beherrscht werden. Systementwickler könnten andernfalls ungenutzte Chipfläche nur durch automatische Generierung homogener Strukturen, wie z. B. mit Speichermatrizen, füllen. Aus diesem Grund sind verschiedene Entwurfsmethoden mit fortschreitender Zunahme der Entwurfskomplexität erforscht und eingesetzt worden.

2.3 Entwurfsmethoden

Um beim Entwurf nicht alle Parameter berücksichtigen zu müssen, wurden Entwurfsmethoden entwickelt, die mit fortschreitender Integrationsdichte zunehmend von den physikalischen Transistorparametern abstrahieren. Dies ist die einzige Möglichkeit, um die fortschreitende Komplexität zu beherrschen. Wurden früher Transistoren durch logische Gatter (AND, NAND, OR, NOR, …) und später Gatter durch arithmetische Funktionen (Addierer, Multiplizierer, Multiplexer, …) abstrahiert, so abstrahieren heute Funktionsblöcke (IP-Module) komplette Prozessoren (CPU, DSP), Mikrokontroller (MCU) usw. Diese IP-Module werden beim IP-basierten Entwurf als Basisbausteine für den Systemaufbau verwendet.

2.3.1 IP-basierter Entwurf

Hardware, die häufig benötigte Aufgaben erfüllt, wird von spezialisierten Anbietern (IP-Anbieter) als wiederverwendbarer Funktionsblock (IP-Modul) entwickelt. So werden beispielsweise im Multimediabereich Audio und Video Codecs, für Ver- und Entschlüsselungsaufgaben, Kryptografieblöcke oder Standardschnittstellen (z. B. PCI, OCP-IP [60]) für die Verbindung der Blöcke über On-Chip Busse (z. B. AMBA) angeboten. Solche IP-Module werden beim IP-basierten Entwurf als Basisbausteine für die SoC-Entwicklung von einem IP-Nutzer verwendet.

Ein SoC besteht aus mindestens einem Prozessor, häufig aber aus einer Kombination von CPU, DSP und MCU, Speicher und Peripheriehardware. Spezielle Hard- und Software zur Erfüllung der Systemfunktionalität wird zusätzlich in Form von IP-Modulen in den Chipentwurf integriert.

IP-Module (engl. Intellectual Property Module) oder kurz IPs werden häufig auch als virtuelle Komponenten (engl. Virtual Components, VC) bezeichnet. Der Begriff „Virtual Component™" ist eine geschützte Bezeichnung der Virtual Socket Interface Alliance für Komponenten, welche die Virtual Socket Interface™ Spezifikation erfüllen [56]. Diese Spezifikation muss noch als visionär bezeichnet werden. Sie beschreibt eine Standardschnittstelle für virtuelle Komponenten und Sockel, die unabhängig von der gekapselten Funktion ist. Vorlagen für Systementwürfe (Plattformen) stellen die virtuellen Sockel bereit, in die dann unabhängig von dieser Vorlage entworfene, beliebige virtuelle Komponenten eingesetzt werden können. Dadurch soll eine einfache Austauschbarkeit und Wiederverwendbarkeit sichergestellt werden. In der Praxis werden die beiden Begriffe IP und VC jedoch gleichwertig benutzt. Im Laufe dieser Arbeit wird in der Regel von IP-Modulen gesprochen, da es der geläufigere Begriff ist. Soll die Einhaltung bestimmter Qualitätskriterien betont werden, wird der Begriff qualifiziertes IP verwendet. Diese Begriffe werden in Abschnitt 2.5 definiert.

Ein gebräuchliches Charakterisierungsmerkmal für IP-Module ist ihre „Härte" beziehungsweise der Grad, mit dem sie auf einen bestimmten Fertigungsprozess abgestimmt sind. Dabei spiegelt der Härtegrad die Abstraktionsebene wieder, auf der die Integrationsdaten des IP-Moduls beschrieben sind. Die üblichen Abstraktionsebenen werden in Abschnitt

2.4 erklärt. Anerkannt sind die Härtegarde der VSIA in [56]. Sie werden im Folgenden erläutert.

2.3.1.1 Soft-IP

Soft-IP wird als synthetisierbare HDL-Beschreibung ausgeliefert und hat dadurch den Vorteil am flexibelsten zu sein und den Nachteil, dass seine zeitliche Leistungsfähigkeit, der Flächenbedarf und die elektrische Leistungsaufnahme nur bedingt vorhersehbar sind. Die Notwendigkeit, den RTL-Quellcode an den IP-Nutzer ausliefern zu müssen, erhöht das Missbrauchrisiko des geistigen Eigentums für Soft-IP-Module.

2.3.1.2 Firm-IP

Firm-IP ist strukturell und topologisch hinsichtlich Leistungsfähigkeit und Flächenbedarf durch Platzierung und Verwendung einer allgemeinen Technologiebibliothek optimiert. Der Detaillierungsgrad reicht von der Platzierung der RTL-Blöcke über relativ platzierte Datenpfade und parametrisierte Generatoren, bis zu einer vollständig platzierten Netzliste. Häufig wird eine Kombination dieser Ansätze angewandt, um das Entwurfsziel zu erreichen. Firm-IP stellt einen Kompromiss zwischen Soft- und Hard-IP dar. Es ist flexibler und portierbarer als Hard-IP, aber vorhersehbarer in der Leistungsfähigkeit und dem Flächenbedarf als Soft-IP. Der Auslieferungsumfang eines Firm-IP-Moduls enthält eine Kombination aus synthetisierbarem RTL-Quellcode, Referenztechnologiebibliothek, detailliertem Platzierungsplan und eine vollständige oder in Teilen vorhandene Netzliste. Wird eine vollständige Netzliste ausgeliefert, so wird erwartet, dass Testlogik eingefügt wurde und Testlisten zum Lieferumfang gehören. Firm-IP ist nicht verdrahtet. Das Missbrauchsrisiko entspricht dem von Soft-IP, wenn RTL-Quellcode zum Lieferumfang gehört.

2.3.1.3 Hard-IP

Hard-IP ist hinsichtlich Leistungsaufnahme, Flächenbedarf und/oder zeitlicher Leistungsfähigkeit optimiert und bereits auf eine bestimmte Technologie abgebildet. Beispiele enthalten vollständig platzierte und verdrahtete Netzlisten, die für eine bestimmte Technologiebibliothek optimiert sind, ein kundenspezifisches, physikalisches Layout oder eine Kombination dieser Optionen. Hard-IP ist spezifisch auf den Technologieprozess abgestimmt und wird im Allgemeinen im GDSII-Format ausgeliefert.

Die Performanz von Hard-IP ist am besten vorhersehbar, aber durch seine Prozessabhängigkeit ist es die am wenigsten flexible IP-Härte. Hard-IP benötigen zumindest ein Verhaltensmodell auf hoher Abstraktionsebene, eine Testliste, vollständige physikalische Modelle und Zeitverhaltensmodelle zusammen mit den GDSII-Daten. Die Möglichkeiten Hard-IP rechtlich zu schützen sind sehr viel klarer durch den Copyrightschutz geregelt und es muss kein RTL-Quellcode ausgeliefert werden. Bedingt durch die enormen Fortschritte bei der Chipherstellung, bedeutet Wiederverwen-

dung von Hard-IP in der Regel, seine GDSII-Darstellung auf eine neue Prozesstechnologie abzubilden.

Die Form des Hard-IP eignet sich am besten zur Auslieferung und Integration analoger und performanzoptimierter Entwürfe. In diesen Fällen ist die EDA-Unterstützung noch unzureichend. Ein optimales Integrationsergebnis kann nur durch die Einbeziehung der unteren Abstraktionsebenen bis hin zur physikalischen Ebene bei der Entwicklung des IP-Moduls erzielt werden. Die Auslieferung von Hard-IP wird daher mittel- bis langfristig als Zwischenlösung angesehen, bis ausreichende EDA-Unterstützung auch für diese Entwürfe verfügbar ist. Die Notwendigkeit, die Entwurfsleistung zu erhöhen, erfordert IP-Module auf hoher Abstraktionsebene zu entwerfen und wieder zu verwenden (Soft-IP). Um mit Soft-IP in die Hard-IP vorbehaltenen Domänen vorzudringen, muss jedoch die Vorhersehbarkeit von Soft-IP gesteigert werden und damit das Integrationsrisiko für den IP-Nutzer verringert werden. Für Soft-IP muss folglich der größte Aufwand getrieben werden, um seine endgültige Ausprägung im SoC vorhersehbar zu machen und so das Integrationsrisiko zu verringern. Die vorliegende Arbeit konzentriert sich daher auf digitales Soft-IP. Abbildung 3 zeigt eine Einordnung der IP-Härtegrade in die Abstraktionsebenen.

Eine Erweiterung des IP-basierten Entwurfs ist der plattformbasierte Entwurf. Beide Entwurfsmethoden sind eng miteinander verbunden.

2.3.2 Plattformbasierter Entwurf

Der plattformbasierte Entwurf berücksichtigt die Tatsache, dass ein neuer Chip in einer Produktfamilie oder einem Applikationsbereich nur wenig neue Funktionalität bereitstellt, sodass die meisten Eigenschaften des Vorgängers beziehungsweise des Applikationsbereichs übernommen werden. Als Beispiel sei die Entwicklung von Digitalkameras und Mobiltelefonen genannt. Bei einem Mobiltelefon etwa bleibt die Funktionalität des Vorgängermodells jeweils erhalten und wird um Zusatzfunktionen wie MP3-Player, UKW-Radio, Digitalkamera erweitert.

Eine Entwurfsplattform besteht aus den grundlegenden Komponenten (Prozessor, Speicher, On-Chip Bus, …), die zur Erfüllung der Systemfunktionalität notwendig sind [55]. Zur Erweiterung oder Differenzierung der Funktionalität innerhalb der Produktfamilien oder zu Produkten der Mitbewerber können zusätzliche IP-Module eingebaut werden beziehungsweise Plattformkomponenten ausgetauscht werden [71]. Bei der Entwicklung einer Plattform wird nach einem Kompromiss gesucht, um einerseits eine Vorlage für möglichst viele Entwürfe bereitzustellen und andererseits die Freiheitsgrade beim Entwurf der auf der Plattform basierenden Produkte zugunsten einer Entwicklungsbeschleunigung einzuschränken.

Je nach adressierter Entwurfsunterstützung unterscheidet man verschiedene Plattformtypen ([58], [59]). Tabelle 2 gibt einen Überblick über die gängigsten Typen und nennt Beispiele.

Plattform	Anbieter	Plattformtyp
AMBA bus architecture [62]	ARM	KZP
Excalibur [61]	Altera	PP
Nexperia [65]	Philips Semiconductors	AP
PrimeXsys [63]	ARM	AP
StarCore [64]	freescale	PZP

Tabelle 2: Übersicht verfügbarer Plattformen

Kommunikationszentrische Plattformen (KZP) geben lediglich die Kommunikationsinfrastruktur mit Peripherie vor und schränken auf diese Weise den Entwurfsaufwand ein. Ein Vertreter dieser Kategorie ist ARMs AMBA Bus.

Programmierbare Plattformen (PP) eignen sich besonders zur Entwicklung von Anwendungen, in denen Rekonfigurierbarkeit verlangt wird. Das Beispiel einer Altera Excalibur Plattform zeigt, dass solche Plattformen in der Regel aus einem rekonfigurierbaren Prozessor und programmierbarer Logik bestehen.

Applikationsplattformen (AP) wie Philips Nexperia oder ARMs PrimeXsys unterstützen komplette Applikationsbereiche. So unterstützt beispielsweise ARMs PrimeXsys in erster Linie Networking- und Automotive-Anwendungen und Nexperia Multimedia- beziehungsweise Kommunikationsanwendungen. Bedingt durch den großen Einsatzbereich ist der Entwurfsaufwand basierend auf Applikationsplattformen sehr hoch.

Prozessorzentrische Plattformen (PZP) geben den Prozessor plus Peripherie, sowie Softwaretreiber und grundlegende Anwendungsroutinen vor und schränken damit den Entwurfsaufwand ein. Ein Beispiel für diesen Plattformtyp ist Freescales StarCore Plattform.

Aufgrund der zu Beginn erwähnten Produktdifferenzierung werden in die Plattformen weitere Hardware-/Software-IP-Module (HW/SW-IP-Module) integriert. Dabei muss die Kompatibilität der IP-Module mit der Plattform und den bereits integrierten IP-Modulen gewährleistet werden. Zu diesem Zweck bieten die Plattformanbieter ein begleitendes Kompatibilitätsprogramm. Durch umfangreiche Kontrollen seitens des Plattformanbieters wird angestrebt, Inkompatibilitäten auszuschließen. Hierbei müssen IP-Module unterschiedlicher Art (HW/SW), unterschiedlicher Härtegrade und unterschiedlicher Qualitätsstandards größtenteils manuell überprüft werden.

2.4 Abstraktionsebenen

Die Beschreibung komplexer Systeme wird durch die Darstellung auf unterschiedlichen Abstraktionsebenen erleichtert. Dadurch kann der Betrachter einer solchen Beschreibung sich auf die für die jeweilige Ebene interessanten Details konzentrieren. Überflüssige Details werden ignoriert. Die Einordnung in Abstraktionsebenen erfolgt im Hardware-Bereich nach dem Y-Diagramm [66]. In der folgenden Abbildung 3 wurden die Abstraktionsebenen aus dem Y-Diagramm übernommen und die vorgestellten Begriffe Plattform und IP-Härtegrade eingeordnet. Je tiefer liegend eine

Ebene ist, desto mehr Details sind in der Beschreibung enthalten, und umso komplexer ist die Beschreibung zu verstehen. Höher liegende Ebenen abstrahieren von diesen Details und helfen somit die Komplexität zu beherrschen. Ermöglicht wird diese Betrachtungsweise durch verfügbare EDA-Werkzeuge die automatisch tiefer liegende Beschreibungen aus den abstrakteren generieren können (Synthese).

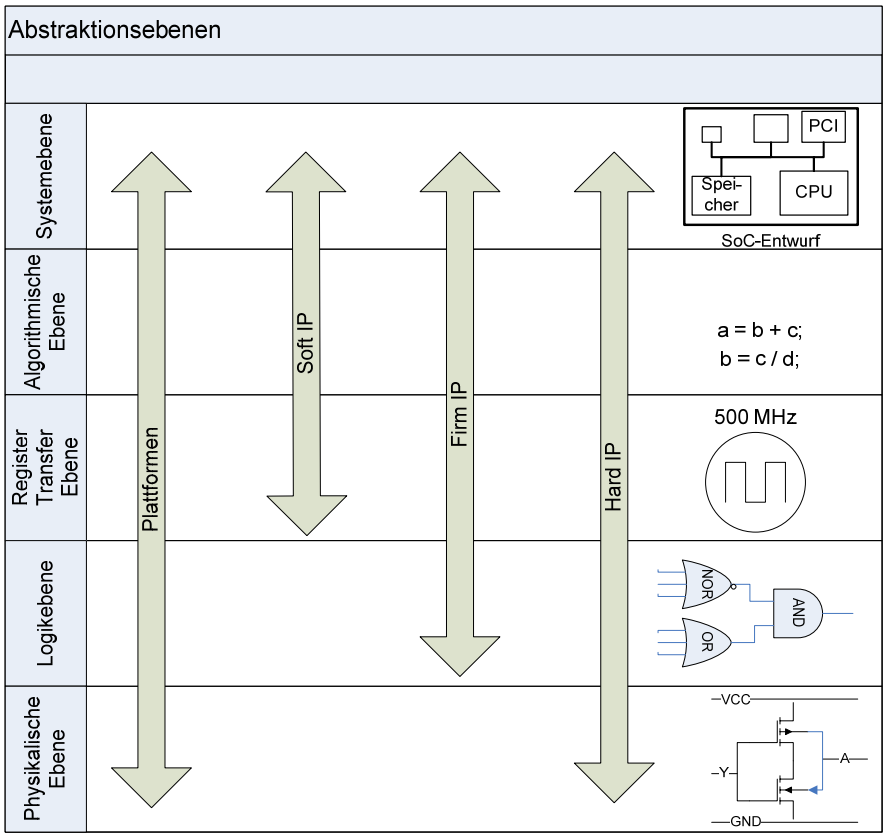


Abbildung 3: Abstraktionsebenen

2.4.1 Systemebene

Auf Systemebene, der abstraktesten Ebene, ist es ausreichend die Funktionalität des Systems zu beschreiben. Es ist nicht erforderlich festzulegen, welche Teile in Hardware gefertigt oder als Software programmiert werden sollen, dies ist Teil der Hardware/Software-Partitionierung. Diese Ebene eignet sich zur Architekturexploration mit abstrakten IP-Modulbeschreibungen. So können geeignet spezifizierte, abstrakte Modelle zur Performanzanalyse verwendet werden, um sich für ein bestimmtes IP-Modul zu entscheiden. Dafür erforderlich ist eine abstrakte Beschreibung des IP-Moduls mit geeigneten Charakterisierungseigenschaften. Die Überprüfung, ob ein solches Modell zum Lieferumfang eines IP-Moduls gehört, ist Teil der automatischen IP-Qualifizierung, die in Kapitel 4 beschrieben wird. Eine Exploration auf dieser Ebene ist die Grundlage, um festzulegen welche Teile auf bereits vorhandene IP-Module abgebildet werden können und welche Teile neu entwickelt werden müssen. Plattformen werden ausgehend von der Systemebene bis zur physikalischen Ebene beschrieben. Dabei ist es nicht erforderlich, dass

auf jeder Ebene eine explizite Beschreibung der Plattform vorliegt. Es müssen nur die Ebenen beschrieben sein, auf denen das System validiert werden soll beziehungsweise Funktionalität integriert werden soll. Teilweise ist es ausreichend, wenn Konzepte zur Einbindung von IP-Modulen einer anderen Abstraktionsebene genutzt werden können.

2.4.2 Algorithmische Ebene

Die algorithmische Ebene erlaubt eine zeitfreie Beschreibung der Funktionalität. Komplexe algorithmische Ausdrücke, Prozeduren, Prozesse und Variablen beschreiben das Verhalten des Entwurfs. Die Struktursicht auf die algorithmische Ebene zeigt Hardware-Module, die nebenläufig interagieren.

2.4.3 Register-Transfer-Ebene

Auf Register-Transfer-Ebene (RT-Ebene) wird das Zeitverhalten einer Schaltung in Taktzyklen betrachtet. Innerhalb eines Taktzyklus muss ein Signal (Datentransfer) ausgehend von einem Register das Zielregister erreicht haben. Daher wird diese Ebene RT-Ebene genannt. Es wird zwischen sequenziellen und kombinatorischen Verhalten unterschieden. Zwischen Registern verhält sich die Schaltung kombinatorisch, d. h. es sind keine speichernden Elemente (Register) zwischen Start- und Zielregister im Signalpfad enthalten. Demgegenüber enthält ein sequenzieller Signalpfad speichernde Elemente.

Soft-IP-Module sind auf RT-Ebene beschrieben. Mit zusätzlichen Zeitinformationen wie Taktdauer und Zeitverhalten der Beschreibungselemente lassen sich auf RT-Ebene bereits qualitative Aussagen über das Zeitverhalten (z. B. synchron, asynchron, multicycle path) machen. Auf Qualitätsprüfungen auf RT-Ebene geht Kapitel 4 detailliert ein.

2.4.4 Logikebene

Auf Logikebene wird ein Schaltungsentwurf sehr implementierungsnah, aber noch unabhängig von der Zieltechnologie, durch Boolsche Gleichungen und Automaten beschrieben. Mit Firm-IP-Modulen macht man sich die bessere Vorhersehbarkeit der implementierungsnahen Darstellung zunutze, ohne die Flexibilität in Bezug auf die Zieltechnologie aufzugeben.

2.4.5 Physikalische Ebene

Diese Ebene beschreibt Schaltungsentwürfe anhand ihres elektrischen Potenzial- und Stromverlaufs mit Differenzialgleichungen. Schaltungskomponenten wie Transistoren, Dioden, Widerstände und Kapazitäten sind durch die Zieltechnologie vorgegeben. Die physikalische Ebene ist die typische Beschreibungsform für Hard-IP-Module. Sie ist aufgrund der zieltechnologienahen Beschreibung die vorhersehbarste Abstraktionsebene in Bezug auf das Verhalten der realen, gefertigten Schaltung. Das bevorzugte Austauschformat auf dieser Ebene ist das GDSII-Format.

Es wurde bereits beschrieben, dass sich die vorliegende Arbeit auf digitale Soft-IP-Module konzentriert. Der IP-Anbieter stellt in diesem Fall Integrationsdaten auf RT-Ebene bereit, die in Bezug auf die Zieltechnologie noch zu abstrakt formuliert sind, um performanzrelevante Aspekte vorherzusehen. Dies erschwert einem IP-Nutzer die Entscheidung für ein IP-Modul, da das mit dem IP-Modul verbundene Integrationsrisiko nur unzureichend abgeschätzt werden kann. Die Aufgabe der IP-Qualifizierung ist es unter anderem, diese Vorhersehbarkeit zu erhöhen. Doch bevor auf die Durchführung der IP-Qualifizierung eingegangen werden kann, muss geklärt werden, was unter Soft-IP-Qualität zu verstehen ist.

2.5 Qualität

Qualität ist nach ISO-9000 definiert als: „Vermögen einer Gesamtheit inhärenter Merkmale eines Produkts, eines Systems oder eines Prozesses zur Erfüllung von Kundenforderungen und anderen interessierten Parteien“ [3].

Diese abstrakte Definition muss zunächst für die Anwendung auf IP-Module interpretiert werden. Dabei ergeben sich drei wesentliche Anforderungen:

1. Es müssen Qualitätskriterien definiert werden, welche die Qualität und in diesem Sinne die Wiederverwendbarkeit und Vorhersehbarkeit eines IP-Moduls ausdrücken können.
2. Qualität muss messbar sein. Eine Qualitätsmessung bedeutet, den Grad der Übereinstimmung mit den definierten Qualitätskriterien zu bestimmen.
3. Der Übereinstimmungsgrad muss bewertet werden, um entscheiden zu können, ob ein IP-Modul von guter oder schlechter Qualität ist.

Zusätzlich muss definiert sein, was qualifiziert werden muss. Bei der Entwicklung eines IP-Moduls beim IP-Anbieter werden die so genannten Entwurfsdaten generiert.

Entwurfsdaten: Daten (HDL-Beschreibungen, Synthese- und Simulationsumgebung, Dokumentation, Qualifizierungsberichte usw.) eines IP-Moduls, die beim IP-Anbieter entstehen. Dazu zählen alle Daten, die für den Entwurf benötigt werden beziehungsweise beim Entwurf und während der Verarbeitung der Entwurfsdaten entstehende Daten (z. B. Protokolldateien der Synthese und Simulation).

Da nicht alle Entwurfsdaten zur Integration benötigt werden, sind die Entwurfsdaten von den Integrationsdaten zu unterscheiden.

Integrationsdaten: Die Integrationsdaten eines IP-Moduls sind in der Regel eine echte Untermenge der Entwurfsdaten. Der Umfang der Integrationsdaten entspricht den Daten, die vom IP-Anbieter an den IP-Nutzer ausgeliefert werden. Sie sind entweder durch Defacto-Standards ([15], [22]) festgelegt oder werden individuell bei Vertragsabschluss zwischen IP-Anbieter und IP-Nutzer vereinbart.

Wichtige Begriffe im Zusammenhang mit der Qualität eines IP-Moduls sind die Vollständigkeit, die Integrität und die Konsistenz eines IP-Moduls.

Vollständig ist ein IP-Modul, wenn alle durch einen Standard definierten oder vertraglich vereinbarten Dateien zum Umfang der Integrationsdaten gehören. Zu beachten ist, dass der Umfang der Integrationsdaten nicht statisch ist, sondern sich aus den jeweiligen Vertragsvereinbarungen und Standards ergibt. Die **Integrität** eines IP-Moduls ist gewährleistet, wenn alle internen Dateiabhängigkeiten eines vollständigen IP-Moduls aufgelöst werden können. Abhängigkeiten zu externen Dateien dürfen nur bestehen, wenn diese ausnahmslos in jeder relevanten Entwicklungsumgebung als verfügbar vorausgesetzt werden können (z. B. VHDL `ieee` und `std` Bibliotheken). Sind dateiformatübergreifend alle Verweise der Dateiinhalte eindeutig, vollständig, und ohne unnötige Verweise erfüllt, ist die **Konsistenz** eines IP-Moduls gewährleistet. Beispielsweise verweisen Objektnamen in Syntheseskripten beispielsweise auf `Entity`-, `Port`-, `Signal`namen usw. des HDL-Quellcodes. Werden während der Entwicklung eines IP-Moduls beispielsweise `Port`namen nur im Quellcode geändert, wird das IP-Modul inkonsistent, da das Syntheseskript noch einen Verweis auf den alten `Port`namen enthält. Eine automatische Qualitätsmessung kann in diesem Fall nur die Inkonsistenz feststellen, nicht aber entscheiden, ob eine Inkonsistenz aufgrund eines unnötigen Verweises (Quellcode-Redundanz) oder aufgrund einer Objektnamensänderung (Quellcode-Änderung) vorliegt.

Teilweise ist es erforderlich zusätzliche Daten für den IP-Nutzer zu generieren, um beispielsweise einen automatischen Import in den Entwurfsablauf des Nutzers zu ermöglichen. Diese Thematik wird in Kapitel 5 ausführlich diskutiert.

Mit diesen Voraussetzungen kann der Begriff der IP-Qualifizierung definiert werden.

IP-Qualifizierung: Die Messung und Bewertung des Übereinstimmungsgrads eines IP-Moduls mit definierten Qualitätskriterien basierend auf den Entwurfs- beziehungsweise Integrationsdaten.

2.5.1 Qualitätskriterien

Viele Bücher und wissenschaftliche Konferenzbeiträge beschäftigen sich mit Qualitätskriterien, um elektronische Entwürfe oder den Entwurfsprozess zu verbessern. Die adressierten Teilbereiche des Entwurfs lassen sich zu den folgenden Themen zusammenfassen: Verbesserung der Codierung (engl. Coding Style) [68], unter anderem in Bezug auf Synthetisierbarkeit oder Simulierbarkeit mit unterschiedlichen Werkzeugen, Entwurf für die Fertigung (engl. Design for Manufacturability, DFM) [67] und Entwurf für Testbarkeit (engl. Design for Test, DFT) ([69], [70]).

IP-Module müssen diese Anforderungen ebenfalls erfüllen und zusätzlich portierbar und wiederverwendbar sein. Aus diesem Grund können aus den genannten Themenkomplexen Qualitätskriterien für IP-Module abgeleitet werden. Viele Firmen haben daraus firmeninterne Richtlinien entwickelt (z. B. STMicroelectronics „Blue Book“). Die Einhaltung der firmeninternen Qualitätsanforderungen führt zwar zu verbesserter Qualität und damit zu besserer Wiederverwendbarkeit, aber die Anforderungen unterscheiden sich von Firma zu Firma. Da die firmeninternen Qualitätskriterien meist unveröffentlicht bleiben, sind dem IP-Nutzer die erfüllten Qualitätskriterien nicht bekannt. Die firmeninternen Qualitätskriterien eignen sich daher nicht zum Qualitätsvergleich von IP-Modulen unterschiedlicher Hersteller. Werden beispielsweise unterschiedliche Codierrichtlinien der Hardware-Beschreibungssprache (engl. Hardware Description Language, kurz HDL) eingehalten und der IP-Anbieter hält die durchgängige Verwendung der `downto`-Anweisung und der IP-Nutzer die durchgängige Verwendung der `to`-Anweisung für qualitativ hochwertig, ist es offensichtlich, dass dadurch eine Quelle für Fehlverbindungen zwischen Bussen solcher unterschiedlich qualifizierter IP-Module gegeben ist.

Um solche Missverständnisse zu vermeiden, ist das Vorhandensein von standardisierten oder zumindest veröffentlichten Kriteriensammlungen, die die Gesamtqualität eines IP-Moduls berücksichtigen, unerlässlich. Mit zu den bekanntesten, veröffentlichten Sammlungen solcher Qualitätskriterien gehören das Reuse Methodology Manual (RMM) [4] und die Kriteriensammlung im Quality IP (QIP) Programm der Virtual Socket Interface Alliance (VSIA) [7].

Auf der Grundlage definierter Qualitätskriterien kann die Qualität eines IP-Moduls gemessen werden. Eine darüber hinausgehende Standardisierung der Kriterien trägt zur Transparenz des Qualitätsbegriffs bei und fördert die Entwicklung automatischer Qualitätsmessmethoden.

2.5.2 IP-Qualitätsmessung

Bei der Qualitätsmessung wird der Übereinstimmungsgrad eines IP-Moduls mit definierten Qualitätskriterien gemessen. Bei einer an die Qualitätsbewertung angepassten Messung hängt das Messergebnis von der Formulierung des Kriteriums und der vorgesehenen Quantifizierung ab. **Quantifizierung** bezeichnet den Detaillierungsgrad eines Qualitätsmessergebnisses bezüglich eines Qualitätskriteriums. Häufig wird darauf geachtet, dass innerhalb einer Qualitätsbewertung eine möglichst einheitliche und einfache Quantifizierung gewählt wird. So waren in OpenMORE lediglich „ja“ und „nein“ Antworten möglich. In QIP gibt es zusätzlich die Möglichkeit Fragen mit „immer“, „häufig“ oder „nie“ zu beantworten. Welchen Detaillierungsgrad eine Antwort haben kann, ist in der Regel durch die Qualitätsbewertung vorgegeben.

Für Qualitätsmessungen können kommerzielle Werkzeuge eingesetzt werden. Synthesewerkzeuge, Simulatoren, Linter und Code-Coverage-Werkzeuge gehören zu den bekanntesten Vertretern. Ein **Linter** (engl. Lint = dt. Fussel) oder **Rule-Checker** ist ein Programmierwerkzeug, das

lexikalische und semantische Quellcodeanalysen durchführt und dadurch Codierrichtlinien überprüfen kann. Eine typische Aufgabe für die Lint-Analysen ist es, Variablen zu finden, die vor ihrer Nutzung nicht gesetzt wurden oder die in einem anderen Zusammenhang verwendet werden als ihr Datentyp es zulässt usw. Daher sind Linter auf ein bestimmtes Datenformat (z. B. C, C++) festgelegt. Heute sind viele typische Lint-Analysen Teil eines optimierenden Compilers. Lint-Analysen prüfen aber auch Fälle, die von einem Compiler nicht geprüft werden, wie z. B. die Konsistenz zwischen Modulen und die Compiler-Portabilität. Die ersten Linter waren für C-Code verfügbar. Heute gibt es auch Linter für weitere Programmiersprachen und HDLs. Für eine umfassende Qualifizierung reichen diese Werkzeuge jedoch nicht aus, da nur einzelne Qualitätsbereiche unabhängig davon wie sie zur Gesamtqualität des IP-Moduls beitragen, adressiert werden. Mit den genannten Werkzeugen können **applikationsunspezifische** Standardkriterien automatisch geprüft werden. Applikationsspezifische Übereinstimmung, beispielsweise der Implementierung eines Busprotokolls mit der Spezifikation, kann von diesen Werkzeugen nicht geleistet werden. Häufig werden daher Qualitätsmessungen noch manuell, beispielsweise während Code-Überprüfungen, durchgeführt.

Qualitätsmessungen sind in der Regel an die durchzuführende Qualitätsbewertung angepasst.

2.5.3 IP-Qualitätsbewertung

Je mehr Qualitätskriterien von einem IP-Modul erfüllt werden, desto höherwertig ist seine Qualität. Aufgrund der Tatsache, dass die Qualitätskriterien mit Bezug auf die Wiederverwendbarkeit formuliert sind, ist mit zunehmender Qualität eine Verringerung des Integrationsrisikos verbunden. Messbare Qualitätskriterien sind daher unabdingbar für die Abschätzung des Integrationsrisikos.

Eine Qualitätsbewertung komprimiert die Vielzahl an Qualitätskriterien in eine überschaubare Anzahl an Qualitätsindikatoren und ermöglicht daher einen effektiven Qualitätsvergleich. IP-Entwicklern zeigt eine Qualitätsbewertung die Abweichung vom intern oder standardisiert festgelegten Qualifizierungsziels an. Dieses Abweichungsergebnis ermöglicht eine Abschätzung des Aufwands, der bis zur Erreichung des Qualifizierungsziels notwendig ist. Es werden die Kriterien hervorgehoben auf deren Einhaltung die Entwickler sich konzentrieren müssen, um das IP-Modul wiederverwendbarer zu machen. Viele IP-Anbieter haben dafür interne Qualitätsrichtlinien entwickelt [43]. Darüber hinaus ist eine Standardisierung einer Qualitätsbewertung für die Ausweitung des IP-Geschäfts erforderlich, da sie IP-Anbietern hilft den Kundenanforderungen zu entsprechen und für IP-Nutzer der Qualitätsbegriff dadurch transparent wird.

Die Virtual Component (VC) Quality Development Working Group (DWG) der VSIA hat einen solchen Standard entwickelt. Folgt man der Definition, dass IP-Qualität als „messbare Übereinstimmung mit festgelegten Quali-

tätskriterien“ definiert ist, muss eine Qualitätsmetrik aus den folgenden Elementen bestehen:

- eine Menge Qualitätskriterien C_i,
- eine Bewertung (Wertemenge) Q_i für jedes Kriterium und
- ein Bewertungsschema, um einen Gesamtqualitätswert des IP-Moduls zu berechnen.

Die Beurteilung eines IP-Moduls umfasst das Folgende:

- für jedes Qualitätskriterium muss ein Qualitätswert $q_i \in Q_i$ bestimmt und zugewiesen werden und
- Berechnung der Gesamtqualität q entsprechend dem Bewertungsschema.

Eine frühe und bekannte Qualitätsbewertung war OpenMORE [5]. OpenMORE basiert auf dem Reuse Methodology Manual (RMM) [4] und weist jedem Kriterium C_i ein Gewicht w_i zu:

- w_i = 2, falls C_i eine RMM-Richtlinie ist und
- w_i = 10, falls C_i eine RMM-Regel ist.

Die Bewertung ist:

- 0, falls das Kriterium nie erfüllt ist,
- ½, falls das Kriterium manchmal erfüllt ist und
- 1, falls das Kriterium immer erfüllt ist.

Die Gesamtqualität wird als gewichtete Summe berechnet: $q = \sum_i w_i \cdot q_i$

Als Ergebnis wird ein durchschnittlicher Qualitätswert berechnet; qualitätsschwache Kriterien können durch hohe Qualität in anderen Bereichen kompensiert werden.

Im Gegensatz dazu folgten der VSIA Virtual Component Transfer (VCT) Ansatz (Anhang D in [15]) und der nie veröffentlichte Quality Spreadsheet (QSS) Ansatz der Philosophie, dass immer das schwächste Glied (Qualitätskriterium) die Kette (IP-Modul) zum Reißen bringt. Daher berechneten diese Ansätze notwendigerweise das Minimum der einzelnen q_i. Eine solche Bewertung ist sehr streng, trägt aber weder zur Unterscheidung zwischen IP-Modulen ähnlicher Qualität bei, noch erlaubt sie dem IP-Entwickler sukzessive Qualitätsverbesserungen anzuzeigen.

Für die QIP-Bewertung wurde ein kombinierter Ansatz gewählt ([7], [8]). Die primäre Bewertung wird als gewichtete Summe mit Standardgewichten durchgeführt. Dies führt zu einem einheitlichen, vergleichbaren Bewertungsschema zwischen funktional gleichwertigen IP-Modulen. Durch eine anwendungs- beziehungsweise nutzerspezifische Auswahl der definierten Gewichte für jedes Kriterium kann eine individuelle Bewertung durchgeführt werden. Die primäre QIP-Bewertung wird durch eine „schwächste-Glied“-Bewertung ergänzt. Dieses sekundäre Bewertungsschema signalisiert zusätzlich, ob problematische Kriterien (Pflichtkriterien

(engl. imperatives) beziehungsweise Regeln (engl. rules)) verletzt sind. Neben den problematisch eingestuften Kriterien existieren Richtlinien (engl. guidelines), deren Einhaltung ein Indiz für gute Entwurfspraktiken ist. Die Einhaltung von optional gewichteten Kriterien erhöht die Qualität. Optionale Kriterien können die Gesamtqualität nicht verschlechtern, ihre Bewertung verhält sich im Verletzungsfall neutral. Falls heilbare Kriterien erfüllt sind, verschlechtern sie die Gesamtqualität. Ein heilbares Kriterium wird in QIP immer von einer untergeordneten, neutralisierenden Gegenmaßnahme begleitet. Ist die Gegenmaßnahme im IP-Modul umgesetzt, neutralisiert sie die Qualitätsverschlechterung. Eine Gegenmaßnahme kann nur bewertet werden, falls das übergeordnete heilbare Kriterium zutrifft. Eine Übersicht über die QIP-Gewichtungen ist in Tabelle 3 zusammengestellt.

Pflicht (engl. imperative)	Verletzungen in dieser Kategorie führen zwangsläufig zu unzuverlässigem Verhalten des IP-Moduls.
Regel (engl. rule)	Regelverletzungen können zu erheblichen und unvorhersehbaren Kosten während der Integration des IP-Moduls führen.
Richtlinie (engl. guideline)	Die Einhaltung von Richtlinien führt zur generellen Verbesserung der Integrationsfähigkeit und der Wartbarkeit des IP-Moduls. Die Erfüllung von Richtlinien wird als Indiz angesehen, dass allgemein als gut bewertete Entwurfspraktiken bei der Entwicklung des IP-Moduls beachtet wurden.
Optional	Die Einhaltung optional gewichteter Kriterien führt zu einer Verbesserung der Qualität, ansonsten verhält sich ihre Bewertung neutral in Bezug auf die Gesamtqualität.
Heilbar (engl. mitigable)	Erfüllt ein IP-Modul ein heilbar gewichtetes Kriterium, führt das zu einer Verschlechterung der Gesamtqualität. Die Auswirkung auf das Qualitätsergebnis kann durch die Einhaltung geeigneter Gegenmaßnahmen neutralisiert werden. QIP definiert zu jedem heilbaren Kriterium eine Gegenmaßnahme.

Tabelle 3: QIP-Gewichtungen

Die Gesamtbewertung eignet sich, um verschiedene IP-Module qualitativ zu klassifizieren. Vor einer Kaufentscheidung sollte der IP-Nutzer die einzelnen Bewertungen der Qualitätskriterien betrachten, um einen detaillierten Einblick in die IP-Qualität und das potentielle Risiko zu erhalten. Evtl. sollte der IP-Nutzer eine Veränderung der Standardgewichtung in Betracht ziehen, um spezifische Bedürfnisse in die Bewertung einfließen zu lassen.

Hinsichtlich ihrer Qualität bewertete IP-Module werden als **qualifizierte IP-Module** bezeichnet. Wurde das IP-Modul nur anhand einer Teilmenge der Kriterien bewertet, spricht man von einem **teilqualifizierten IP-Modul**. Die Daten fertig gestellter IP-Module, die zur Wiederverwendung freigegeben wurden, werden in einer **Wiederverwendungsdatenbank** (engl. IP repository) gespeichert. Eine solche Datenbank pflegen sowohl IP-Anbieter, als auch IP-Nutzer. IP-Anbieter exportieren IP-Module aus der Datenbank, falls ein IP-Nutzer ein IP-Modul einkauft. IP-Nutzer verwenden eine solche Datenbank, um extern eingekaufte IP-Module für die interne Wiederverwendung freizugeben und um bereits intern vorhandene IP-Module in einem Chipentwurf wieder zu verwenden. Beim IP-Anbieter

spricht man vom **IP-Export**, wenn ein IP-Modul zur Auslieferung an den IP-Nutzer aufbereitet wird. Dazu wird auf die Entwurfsdaten in der Wiederverwendungsdatenbank des IP-Anbieters zugegriffen. Nach der Übernahme der Integrationsdaten des exportierten IP-Moduls durch den IP-Nutzer spricht man vom **IP-Import**, wenn das IP-Modul für den Entwurfsablauf des IP-Nutzers aufbereitet wird und in der Wiederverwendungsdatenbank des IP-Nutzers gespeichert wird. Zum Auffinden geeigneter IP-Module bei der **IP-Suche** (engl. IP retrieval) in einer Wiederverwendungsdatenbank, müssen die IP-Module charakterisiert werden. Die **Charakterisierungsdaten** werden zusammen mit den strukturierten Integrationsdaten in der Wiederverwendungsdatenbank gespeichert. Eine **IP-Struktur** ist eine definierte, hierarchische Verzeichnisstruktur, in der die Entwurfs- beziehungsweise Integrationsdaten eines IP-Moduls unter festgelegten Dateinamen eingeordnet sind. Innerhalb einer Firma ist meist eine einheitliche IP-Struktur festgelegt. Über Firmengrenzen hinweg unterscheiden sich IP-Strukturen in der Regel. Die IP-Struktur ist an den Entwurfsablauf angepasst.

Wird ein IP-Modul in einen Entwurfsablauf mit anderer IP-Struktur importiert, muss es umstrukturiert werden. Eine **Umstrukturierung** ist eine Veränderung der IP-Struktur aufgrund von Dateinamensänderungen und Änderungen der hierarchischen Verzeichnisstruktur.

Unter **Anpassung** wird der Prozess der Anpassung der Dateiinhalte nach einer Umstrukturierung verstanden. Das Ziel der Anpassung ist die Ausführbarkeit (z. B. der Skripte) und die Integrität der Daten des IP-Moduls wiederherzustellen.

Ein **IP-Format** entspricht einer IP-Struktur, die um zusätzliche, semantische Informationen ergänzt wird. Es bedarf zusätzlicher Werkzeuge, diese Semantik zu interpretieren. Darüber hinaus werden weitere Informationen in einem IP-Format gespeichert, die beispielsweise die Funktion (charakterisierende Daten), die Qualität des IP-Moduls beziehungsweise den Entwurfsablauf dokumentieren.

Der **Entwurfsablauf** definiert das Vorgehen ausgehend von der Spezifikation einer elektronischen Schaltung über die Implementierung und Verfeinerung bis zur Hardwarerealisierung in einer bestimmten Zieltechnologie. Teilaufgaben im Entwurfsablauf werden von Werkzeugen unterstützt. Diese Werkzeuge sind in den Entwurfsablauf integriert und auf die IP-Struktur abgestimmt. Das bedeutet, dass die Werkzeuge die Eingabedaten an festgelegten Stellen der IP-Struktur erwarten und ihre Ausgabedaten an ebenso festgelegten Stellen unter bestimmten Dateinamen ablegen.

Die üblichen **Zieltechnologien** des integrierten Schaltungsentwurfs sind ASICs (engl. Application Specific Integrated Circuit) oder FPGAs (engl. Field Programmable Gate Arrays). Während ASICs speziell für eine Anwendung hergestellt werden, handelt es sich bei FPGAs um universal vorgefertigte Bausteine, die vom Kunden im Anwendungsfeld, also vor Ort, programmiert werden können. FPGAs werden häufig verwendet, um IP-Module kostengünstig auf Hardware abzubilden und in einer

Umgebung testen zu können. Eine funktionierende Hardwareabbildung eines IP-Moduls, beispielsweise auf einem FPGA, ist ein wichtiges qualitätssteigerndes Kriterium, auch dann, wenn das IP-Modul bei weiteren Wiederverwendungen auf ASICs abgebildet werden soll.

Um die Daten eines IP-Moduls im Entwurfsablauf verarbeiten zu können, werden Konfigurationsdateien für die integrierten Werkzeuge benötigt. Eine **Konfigurationsdatei** enthält spezifische Einstellungen eines Werkzeugs, die zum Erreichen der gewünschten Ausgabe (z. B. Qualifizierungsergebnis, synthetisierte Netzliste usw.) unbedingt notwendig sind. Solche Einstellungen müssen vom IP-Anbieter an den IP-Nutzer ausgeliefert werden, falls die Ausgabe beim IP-Nutzer erzeugt werden muss (z. B. Synthese) oder nachvollziehbar sein soll (z. B. Qualifizierung). Konfigurationsdateien unterscheiden sich von den Installationsdateien.

Eine **Installationsdatei** ist in der Regel eine ausführbare Skriptdatei, die die Umgebung für ein bestimmtes Werkzeug erzeugt. Diese Umgebung ist unbedingt erforderlich, um das Werkzeug ausführen zu können. Bestandteile einer solchen Umgebung sind beispielsweise eine Umgebungsvariable, die auf das Installationsverzeichnis des Werkzeugs verweist, ein Verweis auf den Lizenzserver, das Setzen der Pfadvariable usw. Installationsdateien sind eng mit dem Entwurfsablauf verbunden und gehören daher nicht zum Auslieferungsumfang.

Eine Übersicht über den Wiederverwendungsablauf ist in Abbildung 8 auf Seite 49 grafisch aufbereitet.

Nachdem die grundlegenden Begriffe zum Verständnis der weiteren Arbeit gelegt wurden und geklärt wurde, dass sich die im Folgenden zu erläuternden Methoden auf digitale Soft-IP-Module beziehen, wird im nächsten Kapitel der Stand der Technik für die Qualifizierung und den Austausch von IP-Modulen aufgezeigt.

3 Stand der Technik

Dieses Kapitel gibt einen Überblick über aktuelle Ansätze, Werkzeuge und Organisationen, die mit den Teilgebieten dieser Arbeit, der IP-Qualifizierung und dem IP-Austausch, in Beziehung stehen.

Ausgehend von existierenden Qualitätskriterien für IP-Module und deren Bewertung in Abschnitt 3.1 werden aktuelle Standardisierungsbemühungen für IP-Qualitätskriterien in Abschnitt 3.2 untersucht. Daran schließt sich in Abschnitt 3.3 eine Diskussion über die Notwendigkeit der Automatisierung von Qualitätsmessungen zur Überprüfung von Qualitätskriterien an. In Abschnitt 3.4 werden schließlich verfügbare Qualifizierungswerkzeuge besprochen, die für automatische Qualitätsmessungen verwendet werden können.

Im Anschluss an eine erfolgreiche Qualifizierung werden IP-Module in Wiederverwendungsdatenbanken gespeichert. Aus diesen Datenbanken werden IP-Module für die Wiederverwendung ausgeliefert. Die Eignung dieser Datenbankansätze zur Qualitätserhaltung beziehungsweise Anpassung an neue Entwicklungsumgebungen werden in Abschnitt 3.5 diskutiert. Ansätze der objektorientierten und parametrischen Anpassung von IP-Modulen werden in Abschnitt 3.6 besprochen und in Abschnitt 3.7 die Arbeiten zur Unterstützung der IP-Auslieferung. Hinsichtlich ihrer Tauglichkeit zur Übermittlung der Integrationsdaten wurden Formate zur Verarbeitung elektronischer Systementwürfe in Abschnitt 3.8 untersucht. Mit einer Zusammenfassung der offenen Probleme in Abschnitt 3.9 schließt dieses Kapitel.

3.1 Qualitätskriterien und -bewertung für IP-Module

Das Reuse Methodology Manual (RMM) [4] ist eine umfangreiche Qualitätskriteriensammlung für IP-Module. Die Veröffentlichung des RMM und das Befolgen der vorgeschlagenen Regeln beziehungsweise Richtlinien sind für die IP-Geschäftswelt von Vorteil. Für den IP-Anbieter gibt das RMM Qualitätsziele vor, die zur Verbesserung der Wiederverwendbarkeit beitragen. IP-Nutzer profitieren von einem transparenten, durch das RMM definierten Qualitätsbegriff.

OpenMORE [5] ist eine auf dem RMM basierende Qualitätsbewertung. Aufgrund der Bewertung kann ein qualitativ hochwertiges IP-Modul von einem minderwertigen IP-Modul unterschieden werden. Die Akzeptanz und Umsetzung von RMM und OpenMORE in einigen Firmen hat zu einer Verbesserung des IP-Qualitätsniveaus geführt. Allerdings verhindert der manuelle Aufwand angesichts unzureichender, automatischer Qualitätsmessmethoden, zusammen mit der Tatsache, dass RMM und OpenMORE keine Industriestandards sind, eine weitere Akzeptanz.

Trotz des hohen Bekanntheitsgrads des Reuse Methodology Manual (RMM) [4] und OpenMORE [5] wird für die Akzeptanz einer Kriteriensammlung und Qualitätsbewertung durch alle IP-Anbieter und IP-Nutzer ein Industriestandard benötigt, um mit IP-Modulen das gewünschte Pro-

duktivitätswachstum zu erreichen. Die Standardisierung einer einheitlichen Qualitätsmetrik und einheitlicher Qualitätskriterien sind wesentliche Voraussetzungen dafür, dass der Begriff IP-Qualität eine sinnvolle Bedeutung erhält.

3.2 Standardisierung der Qualitätskriterien

Die Virtual Socket Interface Alliance (VSIA) hat einen solchen Standard erarbeitet [6]. Die Aufgabe der VSIA ist es, „die Produktivität der SoC-Entwicklung durch die Festlegung offener Standards und Spezifikationen, die die Software- und Hardware-VC-Integration aus mehreren Quellen unterstützen, grundlegend zu verbessern“. An der Verbesserung der IP-Qualität arbeitet die Quality Development Working Group (QDWG) [9] beziehungsweise der neu geschaffene Quality Pillar.

Im Juni 2004 ist die erste offizielle Version 1.11 der VSIA-Qualitätsbewertung (Quality IP, kurz QIP) kostenfrei für VSIA-Mitglieder freigegeben worden [7]. Mitglieder können sich von der VSIA-Webseite im Mitgliederbereich [42] das QIP-Arbeitsblatt im Microsoft®-Excel-Format mit dem dazugehörenden Benutzerhandbuch herunterladen. Im Herbst 2004 wurde der Beitritt zur VSIA und damit die Nutzung der QIP-Metrik vereinfacht. VSIA-Mitglieder dürfen die QIP-Bewertungsergebnisse ihrer IP-Module auch Nicht-Mitgliedern kostenfrei zur Verfügung stellen.

Der Standard umfasst derzeit Qualitätskriterien für digitales Soft-IP, digitales Verifikations-IP und eine Bewertung der IP-Reife sowie des IP-Anbieters. QIP ersetzt damit die bekannte OpenMORE Qualitätsbewertung. OpenMORE wurde der VSIA gestiftet [23]. Der Veröffentlichung von QIP ist ein Beta-Test mit großer Industriebeteiligung vorausgegangen. Daneben wurden weitere Firmenbeiträge interner Qualifizierungskriterien untersucht und in QIP integriert.

Der Autor hat innerhalb der VSIA-QDWG bis zum aktuellen Stand aktiv an der Standardisierung mitgearbeitet. QIP ist eine Sammlung von Qualitätskriterien für digitale IP-Module und einer darauf basierenden Metrik. QIP ermöglicht die Bewertung der IP-Qualität und wird derzeit durch die Kategorien digitales Hard-IP, Analog-IP und Mixed-Signal-IP, durch eine Kombination der Kriterien von digitalem Soft- und Analog-IP, ergänzt.

Innerhalb der Kategorien deckt QIP jeweils zwei Sichtweisen ab, die Sicht des Nutzers und die Sicht des Anbieters auf ein IP-Modul. Ein Nutzer hat die Möglichkeit, das IP-Modul vor einer Kaufentscheidung zu bewerten. In diesem Fall ist nur eine eingeschränkte Bewertung möglich, da er zu diesem Zeitpunkt noch keine vollständige Einsicht in das IP-Modul hat. Die Bewertung wird im Wesentlichen auf den Charakterisierungsdaten basieren, die ihm vom IP-Anbieter zur Verfügung gestellt werden. Eine weitere Anwendung von QIP für den IP-Nutzer ist, dem IP-Anbieter eine Rückmeldung bezüglich seiner Erfahrungen bei der IP-Integration zu geben.

Da in den QIP-Kriterien die Nutzeranforderungen an IP-Module manifestiert sind, kann ein IP-Anbieter die Einhaltung der Qualitätsanforderungen

überprüfen. Es liegt also im eigenen Interesse des IP-Anbieters, ein IP-Modul vor der Auslieferung an den IP-Nutzer mit QIP zu bewerten.

In der Kategorie IP-Reife- und IP-Anbieterbewertung müssen vom IP-Anbieter Fragen wie beispielsweise: „Wurde das IP-Modul bereits von einem Referenzkunden in Silizium gefertigt oder in einen FPGA abgebildet?“ beantwortet werden. Die Anbieterbewertung bewertet die Fähigkeiten und Möglichkeiten des IP-Anbieters, qualitativ hochwertige IP-Module zu entwickeln und Support bereitzustellen. Dies wird beispielsweise anhand der Frage: „Ist der technische Support während normaler Geschäftszeiten erreichbar?“, ermittelt.

IP-Nutzer- und Anbietersicht sind in QIP folgendermaßen gegliedert:

- Nutzersicht
  - Dokumentationsqualität:
    Unter diesem Gliederungspunkt werden die Qualität des Integrationshandbuchs, des digitalen Hardware-Referenzhandbuchs und der Versionsanmerkungen ermittelt.
  - Integrationsunterstützung
    Hierunter wird die Qualität der Konfigurierbarkeit und Parametrisierung, Build-Umgebung, Portierbarkeit, Erweiterbarkeit, Systemebenenmodellierung, API-Schnittstelle, Hardware-Schnittstelle, Syntheseunterstützung, Produktionstest (JTAG, ATPG usw.), Blockebenen-Selbsttest, Verifikationsumgebung und SoC-Verifikationsunterstützung bewertet.

Die Anbietersicht gliedert sich wie folgt:

- Entwurfs- und Verifikationsqualität
  - Entwurfsqualität – interne Entwurfsdokumentation:
    Die Qualität der internen Entwurfsdokumentation wird anhand von Produktbrief, Projektplan und der Dokumente über die Systemanforderungen, Entwurfsspezifikation, Verifikationsspezifikation und Qualitätsabschlussprüfung sowie der Prüfliste für den IP-Modul Speicher bewertet.
  - Entwurfsqualität – Entwurfsdetails:
    In diesem Abschnitt muss der IP-Entwickler über die Systemtechnik, Entwurf für die Synthese, eingebettete Speicher, Taktdomänen, Rücksetzrichtlinien, Code-Kommentierung, Codierrichtlinien und Skripte Auskunft geben.
  - Entwurf für den Test, Fertigungstest und Fertigung sind weitere wichtige Details, die durch die Unterabschnitte Verifikationsqualität, Code-Coverage, Datentransfer, Terminierung, Konfiguration, Datengenerierung, Protokollprüfung, Behandlung der Verifikationsrücksetzung, Verifikationskomponenten und -umgebung und Prozessprüfliste ergänzt werden.

QIP ist eine veröffentlichte, verfügbare Kriteriensammlung und ermöglicht dadurch eine für IP-Anbieter und IP-Nutzer transparente Qualitätsbewer-

tung. Die Akzeptanz eines Qualitätsstandards hängt aber stark davon ab, wie aufwendig die Durchführung einer Qualitätsmessung ist.

3.3 Automatisierung der Qualitätsmessungen

Ein Nachteil, der dem IP-basierten Entwurf anhaftet, ist ein zwei- bis dreimal höherer Entwicklungsaufwand, der darin besteht, den Entwurf des IP-Moduls wiederverwendbar zu machen [11]. Zum erhöhten Entwicklungsaufwand zählt auch das Aufzeigen der Übereinstimmung des IP-Entwurfs mit Qualitätskriterien. Dieser erhöhte Aufwand kann durch automatische Qualitätsmessungen reduziert werden.

Bisher ist die Qualitätsmessung sowohl für IP-Anbieter als auch IP-Nutzer nur unzureichend gelöst. Häufig bedeutet eine IP-Qualifizierung aufseiten des IP-Anbieters, Papiersammlungen von Qualitätskriterien einzuhalten. Gehören Qualitätsmessungen zum festen Bestandteil der Firmenpolitik, müssen sie dennoch oft manuell durchgeführt werden. Zeitintensive und fehleranfällige Code-Überprüfungen sind die Folge. Eine Qualifizierungsmethode, die die Entwicklung des IP-Moduls begleitet, ist meist nicht installiert. Auch für den IP-Nutzer gibt es nur unzureichende Unterstützung bei der Auswahl des „richtigen“ IP-Moduls. Sehr häufig kann eine Qualitätsmessung aus Zeit- und Komplexitätsgründen und in Ermangelung automatisierter Prozesse nicht durchgeführt werden.

Daher ist die Qualität des IP-Moduls nur so gut wie das Vertrauen, das in die Fähigkeiten des IP-Anbieters gesetzt wird. Die Folge von nicht effizient durchführbaren Qualitätsmessungen können Integrationsprobleme sein, die im Vorfeld hätten vermieden werden können. Solche Integrationsprobleme führen zumindest zu Verzögerungen im Projektplan und schlimmstenfalls zum Scheitern des Projekts, falls aufwendige Nacharbeiten durch den IP-Anbieter während des Integrationsprozesses notwendig werden. Das führt zwangsläufig zu einer nachhaltigen Schädigung der Geschäftsbeziehungen, da keine objektiven und effizienten Qualitätsmessmethoden verfügbar sind, die eine erneute Kaufentscheidung rechtfertigen könnten.

3.4 Qualifizierungswerkzeuge

Neben den veröffentlichten Hinweisen, welche Kriterien qualitativ hochwertige und damit wiederverwendbare IP-Module erfüllen müssen, gibt es auch Werkzeuge, wie beispielsweise Rule-Checker (Linter) ([12], [45], [46]), Code-Coverage- ([13], [47]) und Synthese-Werkzeuge [14], die für einzelne Qualitätsmessungen eingesetzt werden können. Mit diesen Werkzeugen können allerdings nur Teilbereiche der Qualifizierung durchgeführt werden. Es gibt kein umfassendes Konzept für eine Gesamtbeurteilung der Qualität eines IP-Moduls, beispielsweise nach VSIA-QIP, wonach alle Einzelmessungen in eine Gesamtbewertung einfließen. Abhilfe kann ein automatisierter Qualifizierungsablauf schaffen, der in den Entwurfsablauf integriert werden muss. Als Beispiel sei auf die Verifikation verwiesen. Heute ist es selbstverständlich, eine entwurfsbegleitende Verifikationsmethodik installiert zu haben.

3.5 Wiederverwendungsdatenbanken

IP-Wiederverwendungsdatenbanken werden in [30]-[33] vorgeschlagen. Die zur Wiederverwendung bereitstehenden IP-Module werden in einer Wiederverwendungsdatenbank gespeichert. Auf effiziente Suchalgorithmen zum Auffinden geeigneter IP-Module in der Wiederverwendungsdatenbank konzentriert sich [34]. Während andere Suchmethoden nur auf manuell aufbereiteten Charakterisierungsdaten basieren, erlaubt dieser Ansatz eine geschützte Fernsimulation eines IP-Modells. Dadurch werden detaillierte Prüfungen eines IP-Moduls vor einer Kaufentscheidung ermöglicht. Allerdings kann ein Simulationsmodell nicht alle, für eine problemlose Integration notwendigen Qualitätskriterien berücksichtigen. Zudem muss ein eigenständiges Modellformat gepflegt werden.

Die Autoren von [30] konzentrieren sich auf die Integration einer IP-Wiederverwendungsdatenbank in den Firmenentwurfsablauf für den effizienten Zugriff auf IP-Module durch unterschiedliche Abteilungen innerhalb der Firma. Dieser Ansatz unterstützt die Verwaltung und Verbreitung wiederverwendbarer IP-Module.

Das Reuse Management System (RMS) adressiert die IP-Suche. Es wird in [30] und [31] vorgestellt. Die Stärke von RMS ist die Kombination von Taxonomie-, Schlüsselwort-, Attribut- und Ähnlichkeitssuchalgorithmen. RMS unterstützt den IP-Nutzer bei der Suche nach verfügbaren IP-Modulen auf der Basis von funktionalen Kriterien. Im Gegensatz dazu beschränken sich viele Ansätze bei der Suchfunktionalität auf eine Schlüsselwortsuche in definierten Kategorien. Diese Suchmethode ist aber nicht erfolgreich, falls keine exakt passenden IP-Module vorhanden sind. In diesem Fall ist es wünschenswert auch ähnliche, anpassbare IP-Module zu finden. Fortschrittlichere Methoden werden noch erforscht. Einige Vorschläge zur Ähnlichkeitssuche gibt es in ([19], [20], [33]).

IP-Suchdienste verwenden für die Suche IP-Charakterisierungsdaten. Diese Daten müssen manuell gepflegt werden, während ein IP-Modul in die Wiederverwendungsdatenbank aufgenommen wird. Dazu ist meist Expertenwissen über das IP-Modul notwendig.

Aufgrund fehlender Qualitätsattribute in der Charakterisierungsdatenbasis ist eine Risikoabschätzung bei der IP-Suche nicht möglich. Zudem bieten Wiederverwendungsdatenbanken nur geringe bis keine Unterstützung bei der Integration der Entwurfsdaten in den SoC-Entwurfsablauf.

3.6 Anpassung von IP-Modulen

Die Autoren von [35] stellen eine Erweiterung von RMS um ein objektorientiertes VHDL-Modell (objective VHDL) vor. Damit konzentriert sich dieser Ansatz auf den Wunsch, vorhandene, aber nicht exakt passende IP-Module durch funktionale Änderungen an neue Wiederverwendungsmöglichkeiten anzupassen.

Eine weitere Anpassungsmöglichkeit für IP-Module an den Systementwurf ist das Setzen vorgegebener Parameter. Um ein IP-Modul möglichst allgemein wiederverwendbar zu gestalten, werden funktionale Parameter,

wie beispielsweise Parität, Datenwortlänge und Paritätsbitposition in den Entwurfsdaten variabel gehalten. Bevor ein solches IP-Modul in den Systementwurf integriert werden kann, müssen diese Parameter durch konkrete Werte ersetzt werden. Dafür gibt es bereits vereinzelt Werkzeuge, die das korrekte Setzen der Parameter im gültigen Wertebereich unterstützen und dabei die Abhängigkeiten zwischen den Parametern beachten ([36], [24]).

Diese Ansätze ermöglichen zwar eine funktionale Anpassung des HDL-Entwurfs, berücksichtigt aber nicht die Integrationsdaten (Synthese-, Verifikations-, Dokumentationsdaten usw.) und deren Anpassung an eine neue Integrationsumgebung.

Des Weiteren wurde untersucht, ob für die Format- beziehungsweise Modellanpassungen während des IP-Imports auf Beiträgen aus dem Forschungsgebiet der Datenbanktechnik, genauer dem Model-Management, aufgebaut werden kann. Untersuchungen (z. B. Kapitel 5, [51] und [49]) haben gezeigt, dass IP-Formate in Datenbankmodellen abgebildet werden können.

In [52] werden Operatoren für die semantische Abbildung zwischen Modellen vorgestellt. Es liegt nahe, die semantischen Abbildungen auf die unterschiedlichen Formate zwischen IP-Anbieter und IP-Nutzer zu übertragen. Neu ist in [52] die Anwendung auf komplette Modelle und die Adressierung der semantischen Heterogenität (= Mehrdeutigkeit). Allerdings zielt der Beitrag auf große Datenbestände in heterogenen Umgebungen. Dabei verändern sich die Datenmodelle häufig und befinden sich über ihre Umgebungsgrenze hinweg in einer kontinuierlichen Abhängigkeit voneinander. Ursprungs- und Zielmodell sind beim Model-Management immer bekannt. Häufig müssen große Datenmengen von einem Modell ins andere überführt werden.

Das ist bei der Anpassung von IP-Strukturen nicht der Fall. Der IP-Anbieter weiß in der Regel nicht, welche Struktur das IP-Modul beim IP-Nutzer hat. Zudem ist die Häufigkeit sowohl der Modelländerungen als auch der wiederholten Datenübermittlung nicht gegeben. Für zukünftige Untersuchungen der Abbildungseindeutigkeit des Ursprungsformats über das Zwischenformat in das Zielformat sollten die Ergebnisse aus diesem Forschungsbereich einbezogen werden. Die vorliegende Arbeit beschäftigt sich mit den Fragen, welche Inhalte des Formats sich verändern, wie diese Inhalte automatisch extrahiert werden können und wie eine automatische Anpassung der Inhalte erfolgen kann.

3.7 IP-Auslieferung

Eine Anpassung der Integrationsdaten, d. h. eine Anpassung an einen neuen Entwurfsablauf und die Anpassung der IP-Struktur, ist bei der Auslieferung des IP-Moduls vom IP-Anbieter zum IP-Nutzer und beim Import der Daten in den dortigen Entwurfsablauf notwendig. Die Arbeit in [33] beschreibt die Definition von Paketen. Diese bieten die Möglichkeit, voneinander abhängige Dateien in einem Paket zu kapseln. Dadurch werden unvollständige IP-Auslieferungen vermieden.

Jedoch handelt es sich dabei um einen statischen Ansatz. Pakete müssen im Vorfeld definiert werden. Eine automatische Entwurfsablaufintegration ist nicht vorgesehen.

Eine weitere Möglichkeit Daten zwischen Entwurfsabläufen effizient zu übertragen, ist die Verwendung einheitlicher Formate.

3.8 IP-Formate

Für die Verarbeitung elektronischer Entwürfe gibt es verschiedene Formate, in denen Entwurfsdaten und teilweise sogar Integrationsdaten gekapselt sind. Diese Formate sollen die Entwicklung elektronischer Systeme verbessern. Die für den IP-Austausch relevanten Formate werden im Folgenden diskutiert.

3.8.1 OpenAccess-Format

OpenAccess [49] ist eine spezifizierte Schnittstelle (API) zur Entwurfsdatenbasis für EDA-Werkzeuge. Dadurch entfallen Konvertierungen zwischen EDA-Formaten und mehrfache Datenhaltung, um Entwürfe mit unterschiedlichen Werkzeugen zu bearbeiten.

Für die Nutzung als IP-Austauschformat ist es allerdings Voraussetzung, dass sowohl IP-Anbieter als auch IP-Nutzer ihre Entwurfsabläufe komplett auf OpenAccess umstellen und alle benötigten Werkzeuge im Entwurfsablauf über eine OpenAccess Schnittstelle verfügen.

Zudem konzentriert sich OpenAccess hauptsächlich auf den „Backendbereich“, also die Abstraktionsebenen unterhalb der digitalen Soft-IP-Entwicklung und des IP-Austauschs. Eine Ausnahme ist die von Hewlett-Packard und Cadence Design Systems entwickelte Lese- und Schreibschnittstelle für das Verilog-Format [50], die es erlaubt Netzlisteninformationen im OpenAccess-Format auszutauschen.

Besser geeignet ist ein Format, das den Werkzeugen die Daten in gewohnter Weise, d. h. als Dateisystemstruktur bereitstellt und die Daten automatisch während des Austauschs an die neue Entwicklungsumgebung anpasst. Ein Verzicht, auf eine von nicht allen EDA-Werkzeugen bereitgestellte API, gestaltet den IP-Austausch und -Import in einen neuen Entwurfsablauf für IP-Anbieter und IP-Nutzer transparent.

3.8.2 Platform Express Format

Die Entwicklungsumgebung Platform Express von Mentor [72] unterstützt den plattformbasierten Entwurf. Systemarchitekturen können evaluiert, Simulationsumgebungen erstellt und eine HDL-Beschreibung des Systems erzeugt werden. Dafür muss ein IP-Anbieter eine XML-Beschreibung seines IP-Moduls bereitstellen, sodass ein IP-Nutzer anhand dieser Beschreibung das IP-Modul in Platform Express importieren kann. Die XML-Beschreibung zusammen mit den begleitenden Daten werden ein Platform Express Objekt genannt. Solche Objekte werden in eine Platform Express Bibliothek importiert. Über eine grafische Eingabe können Systemarchitekten IP-Module aus der Bibliothek in einen SoC-Entwurf einbinden.

Das Konzept der Konfiguratoren erlaubt es einem IP-Anbieter, Konfigurationsschritte festzulegen, die beim Einbinden seines IP-Moduls in einen Systementwurf durchgeführt werden müssen. Mithilfe von Generatoren werden unter anderem die HDL-Systembeschreibung und eine Systemdokumentation generiert. Das Systemmodell kann in Platform Express mit ModelSim [73] und Incisive [74] simuliert werden. Mit Seamless [75] kann die HW/SW-Schnittstelle verifiziert werden.

Trotz der Möglichkeiten, die Platform Express bietet, kann es keinen kompletten Entwurfsablauf bis zur Erzeugung der GDSII-Daten ersetzen. Es werden nur wenige Werkzeuge eines Entwurfablaufs unterstützt. Der Nutzen des Platform Express Objektformats ist daher zum einen auf Entwicklungsabläufe, in denen Platform Express verwendet wird und zum anderen lediglich auf einen Teil des Entwurfsablaufs beschränkt. Hinzu kommt, dass die Erzeugung der Platform Express Objektdaten aus einem vorhandenen IP-Modul, das in firmenspezifischer Struktur vorliegt, nicht vollständig unterstützt wird.

3.8.3 SPIRIT-Format

Das SPIRIT-Consortium [51] entwickelt ein XML-Format zur Spezifikation von IP-Modulen. Diese Spezifikation stellt für mehrere Abstraktionsebenen spezifische Metadaten von IP-Modulen für EDA-Werkzeuge bereit, welche die Integration, Konfiguration, den Austausch mit Im- und Export sowie die Validierung von IP-Modulen im Systemkontext ermöglichen. Die erste Version der SPIRIT-Spezifikation wurde Ende 2004 veröffentlicht. In dieser ersten Version wird ein lose gekoppelter Werkzeugablauf favorisiert. Das bedeutet, dass SPIRIT konforme Werkzeuge für die Kommunikation mit anderen Werkzeugen im SPIRIT-Entwurfsablauf das SPIRIT-Format als Eingabe akzeptieren müssen und als Ausgabe erzeugen müssen. In einer zweiten Version soll dann eine engere Kommunikation der Werkzeuge durch eine spezifische API eingeführt werden. Durch die Beteiligung von Mentor im SPIRIT-Consortium sind einige Konzepte des Platform Express Objektformats in das SPIRIT-Format eingeflossen.

Bisher ist nicht bekannt, inwiefern Qualitätsmessungen auf dem SPIRIT-Format aufsetzen können beziehungsweise deren Ergebnisse das SPIRIT-Format erweitern können. Wichtig ist in einem solchen Fall, dass diese Zusatzinformationen die auf dem SPIRIT-Format basierende API zu den EDA-Werkzeugen nicht in ihrer Funktion beeinträchtigen. Nachteilig ist, dass das SPIRIT-Format zunächst eine lose Schnittstelle, später eine API benötigt, welche von den verwendeten EDA-Werkzeugen im Werkzeugablauf bereitgestellt werden muss. Dadurch wird die Integration in aktuelle, unmodifizierte Entwurfsabläufe nicht unterstützt.

3.8.4 Advanced Library Format

Beim Entwurf elektronischer Schaltungen in Nanometertechnologie gelten Modellierungsannahmen nicht mehr, die bisher Gültigkeit hatten. Es hat sich herausgestellt, dass technologiespezifische Parameter bereits auf

höheren Abstraktionsebenen beachtet werden müssen. Das Advanced Library Format (ALF) [48] ist ein einheitliches Format, das funktionale, elektrische und physikalische Performanzwerte einer Technologiebibliothek beschreibt. EDA-Werkzeuge, die dieses Format unterstützen, können die enthaltenen Daten zur Schaltungsoptimierung verwenden. Das ALF ist seit 2003 ein IEEE-Standard (P1603). Bisher gibt es aber nur wenige unterstützende EDA-Werkzeuge.

Das ALF eignet sich nicht als IP-Integrationsformat, da es nur charakterisierende Daten enthält. Interessant ist es jedoch, um die Vorhersehbarkeit von IP-Modulen zu erhöhen, indem es technologiespezifische Parameter bereitstellt.

3.9 Zusammenfassung der offenen Probleme

Durch QIP gibt es ein einheitliches Verständnis, was unter wiederverwendbarem beziehungsweise einem qualitativ hochwertigen IP-Modul zu verstehen ist. Der Qualitätsbegriff ist durch applikationsunspezifische Qualitätskriterien und einer darauf basierenden Bewertung sinnvoll definiert. Allerdings wird eine effiziente, automatische Qualitätsmessung, die zu einer Verringerung des Integrationsrisikos beiträgt, nur unzureichend durch einzelne Werkzeuge unterstützt. Es wurde bereits darauf hingewiesen, dass durch mangelhafte Qualitätsmessmethoden Geschäftsbeziehungen nachhaltig geschädigt werden können.

Für IP-Module, die für die Wiederverwendung entwickelt werden, also definierte Qualitätskriterien erfüllen müssen, besteht immer ein erhöhter Entwicklungsaufwand. In dieser Arbeit wird gezeigt, dass sich trotz dieses erhöhten Aufwands die Entwicklung wiederverwendbarer IP-Module für den IP-Anbieter lohnt. Profitabel wird der Wiederverwendungsprozess durch den Einsatz einer effizienten Qualifizierungsmethode im Entwurfsablauf.

Für die Erhaltung der Qualität und einen automatisierten Import bei der Übertragung von IP-Modulen vom IP-Anbieter zum IP-Nutzer muss ein geeignetes IP-Format gewählt werden. IP-Wiederverwendungsdatenbanken basieren häufig auf Versionskontrollsystemen, z. B. CVS, RCS, Synchronicity, Clearcase oder einem proprietären Format, wie das RMS-Format. Auf die Datenformate OpenAccess, das SPIRIT-Format, das Mentor Graphics Platform Express Objektformat und das Advanced Library Format wurde ebenfalls eingegangen. Für den Zugriff auf die Integrationsdaten, die in diesen Formaten vorliegen, ist immer eine spezifische Schnittstelle (engl. Application Programmable Interface, API) notwendig. Solche Schnittstellen werden von den wenigsten Werkzeugen innerhalb eines Entwurfsablaufs unterstützt oder sind zumindest nicht bei allen potentiellen IP-Nutzern verfügbar. Die Formate sind untereinander nicht kompatibel. Aus Gründen der Kompatibilität mit vorhandenen Entwurfsabläufen schlägt diese Arbeit vor, die Integrationsdaten eines IP-Moduls als reine Datei-/Verzeichnisstruktur mit Zusatzinformationen auszuliefern. In den Zusatzinformationen werden automatisch generierte

Daten zur Qualitätserhaltung und für die automatisierte Integration in den Entwurfsablauf des IP-Nutzers übertragen.

Manche Ansätze, wie beispielsweise objective VHDL oder die vorgestellten Ansätze zur Parametrisierung, bieten zwar Anpassungsmöglichkeiten der HDL-Entwurfsdaten, dabei wird allerdings nicht der gesamte Umfang der Integrationsdaten betrachtet.

Um die Vollständigkeit einer IP-Auslieferung zu gewährleisten, wird in dieser Arbeit der Ansatz verfolgt, eine Vollständigkeitskontrolle basierend auf Abhängigkeiten der tatsächlich zusammengestellten Auslieferung zu realisieren. Dabei werden die Daten bezüglich ihrer Abhängigkeiten analysiert. Für die Integration in die Entwicklungsumgebung beim IP-Nutzer wird der Ansatz einer semantischen Abbildung des Anbieterformats in das Nutzerformat untersucht.

Kurz zusammengefasst werden in der vorliegenden Arbeit die folgenden Punkte bearbeitet:

- Beiträge zur Standardisierung von applikationsunspezifischen Qualitätskriterien und -bewertung in den Abschnitten 3.2 und 7.1
- Applikationsunspezifische Automatisierung von Qualitätsmessungen und Qualitätsbewertung in Abschnitt 4
- Überprüfbarkeit der applikationsunspezifischen IP-Qualität durch IP-Anbieter und IP-Nutzer in Kapitel 5
- Erhaltung der Qualität bei der IP-Auslieferung in den Abschnitten 5.5.1-5.5.4
- Automatisierte Integration in den Entwurfsablauf in Abschnitt 5.5.4
- Integration der Qualitätsmessmethoden in bestehende Entwurfsabläufe in Kapitel 6
- Automatische Erzeugung eines Übertragungsformats in Abschnitt 6.3.4.3

4 IP-Qualifizierung

In diesem Kapitel wird die vom Autor entwickelte Qualifizierungsmethode beschrieben. Dafür hat der Autor zunächst eine Automatisierbarkeitsuntersuchung der Qualitätsmessungen in Abschnitt 4.1 angestellt. Basierend auf diesem Ergebnis hat der Autor in Abschnitt 4.2, Phasen im Entwurfsablauf und Wiederverwendungsprozess identifiziert. In diesen Phasen werden Qualitätsmessungen durchgeführt, um eine nachhaltige Verbesserung der Entwurfsqualität zu erzielen. Vom Autor entwickelte Qualitätsmessmethoden werden in den einzelnen Phasen zur automatisierten Überprüfung der jeweils relevanten Qualitätskriterien eingesetzt. Sie werden in Abschnitt 4.3 besprochen.

Im nächsten Abschnitt wird die Automatisierbarkeit von Qualitätsmessungen am Beispiel der VSIA Quality IP-Metrik untersucht.

4.1 Automatisierbare Kriterienprüfungen

Manuelle Qualitätsmessungen sind aufgrund ihrer Komplexität und des benötigten Zeitaufwandes für aktuelle und erst recht für zukünftige Entwürfe nicht mehr durchführbar. Qualitätskriterien müssen automatisch geprüft werden. Aufgrund dieser Tatsache hat der Autor den VSIA-QIP-Standard (Version 1.11) hinsichtlich der Automatisierbarkeit der Kriterien für digitale Soft-IP-Module untersucht. Das Ergebnis der Untersuchung ist in Abbildung 4 grafisch aufbereitet

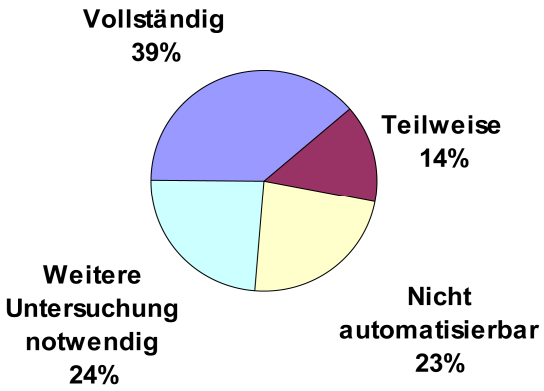


Abbildung 4: Automatisierbare VSIA-QIP-Kriterienprüfungen

Dabei wurden die Kriterien in eine der vier Kategorien eingeordnet:

1. **Vollständig automatisierbar** bedeutet, dass bereits Werkzeuge beziehungsweise Techniken verfügbar sind, mit denen dieses Kriterium vollständig überprüft werden kann.
2. **Teilweise automatisierbar** bedeutet, dass zumindest ein Teil dieses Kriteriums durch verfügbare beziehungsweise zu entwickelnde Werkzeuge überprüft werden kann. Für die Dokumentation beschränkt sich der überprüfbare Anteil der Kriterien, wenn auch nicht immer, so doch häufig auf die Verfügbarkeit wie z. B. aufgrund des Vorhandenseins bestimmter Dateien.

3. **Nicht automatisierbar** bedeutet, dass zum momentanen Stand der Technik keine Möglichkeit besteht, dieses Kriterium automatisch zu überprüfen.
4. **Weitere Untersuchung notwendig** besagt, dass für eine endgültige Zuordnung in eine der ersten drei Kategorien eine weitere Untersuchung des Kriteriums notwendig ist.

Diese Untersuchung wurde nicht mit dem Ziel der Vollständigkeit durchgeführt, sondern um die hohe Automatisierbarkeit der QIP-Qualifizierung darzustellen. Das Untersuchungsergebnis zeigt, dass mehr als 50 % der Kriterien zumindest (teil-)automatisiert geprüft werden können. Bereits dieses Ergebnis verdeutlicht das hohe Automatisierungspotenzial von QIP. Daher wurden die Kriterien, die bisher nicht in eine der ersten drei Kategorien eingeordnet sind, nicht weiter untersucht. Das Ergebnis bestätigt auch, dass der zusätzliche Aufwand für die Qualifizierung eines IP-Moduls gering gehalten werden kann. Nur durch möglichst geringen Qualifizierungsaufwand und durch eine Reduzierung des Integrationsrisikos wird sich ein Qualitätsstandard behaupten. Dass die Einhaltung von Qualitätskriterien auch zu einer erheblichen Reduktion des Integrationsaufwands führt, ist ein Ergebnis dieser Arbeit und wird in Kapitel 7 beschrieben.

Zur effizienten Durchführung der Qualifizierung müssen die automatisierbaren Qualitätskriterien als Qualitätsmessmethoden formuliert werden. Eine **Qualitätsmessmethode** zeichnet sich dadurch aus, dass sie die komplette oder partielle Übereinstimmung eines IP-Moduls mit einem Qualitätskriterium automatisch überprüfen kann.

Qualitätsmessmethoden müssen die Entwicklung und Wiederverwendung eines IP-Moduls in verschiedenen Phasen des Lebenszykluses begleiten. Diese Phasen werden im folgenden Abschnitt identifiziert und in einen IP-Qualifizierungsablauf integriert. Der Qualifizierungsablauf gibt eine Richtlinie vor, in welcher Phase welche Messmethoden anzuwenden sind, um einerseits den Qualifizierungsaufwand so gering wie möglich zu halten (Vermeidung von Mehrfachqualifizierung) und andererseits so früh wie möglich im Entwurfsablauf Qualität messbar zu machen und somit die Entwickler zu schulen. Dadurch wird eine nachhaltige Verbesserung, nicht nur der IP-Modul-Qualität selbst, sondern auch der Entwicklungsqualität erreicht.

4.2 Entwurfsablauf und Wiederverwendungsprozess

In einem weiteren Schritt wurden der Entwurfsablauf und der Wiederverwendungsprozess analysiert, um geeignete Ansatzsatzpunkte für die Qualifizierung zu finden. Dabei muss zwischen Qualifizierungen, die aufseiten des IP-Anbieters und aufseiten des IP-Nutzers erforderlich sind, unterschieden werden. Eine übergreifende Qualifizierung ist im Falle eines externen IP-Nutzers nicht möglich.

Drei wesentliche Entwurfsschritte konnten während der Entwicklung eines IP-Moduls beim IP-Anbieter identifiziert werden. Zu den Schritten zählen die **Umsetzung der Funktionalität in HDL-Quellcode**, die **Verifikation**

der Funktionalität und die **Synthese auf eine Beispieltechnologie**. Auch wenn Soft-IP-Module technologieunabhängig, also unsynthetisiert an den IP-Nutzer ausgeliefert werden, müssen dennoch die erforderlichen Syntheseskripte entwickelt werden. Eine Synthese zur Gewährleistung der Synthetisierbarkeit muss Teil der Qualifizierung sein.

Alle diese Entwurfsschritte sollten von einer Qualifizierungsphase begleitet werden. Im Folgenden wird von der **entwurfsbegleitenden Qualifizierungsphase** gesprochen (Abbildung 6 auf Seite 37, ❶). Es kann sehr aufwendig werden, die erforderliche Qualität nach Abschluss des Entwurfs in das IP-Modul „hineinzuprüfen“. In Kapitel 7 wird anhand konkreter Zahlen dargestellt, welcher Aufwand notwendig ist, um ein bereits ohne Berücksichtigung von Wiederverwendbarkeitskriterien entwickeltes Modul im Nachhinein zu einem wiederverwendbaren, qualifizierten IP-Modul aufzuwerten.

Zusätzlich muss nach der Fertigstellung eines IP-Moduls eine **abschließende Qualifizierung** (Abbildung 6 auf Seite 37, ❷) durch eine unabhängige Instanz durchgeführt werden. Diese Instanz kann Teil der Entwicklungsfirma sein, sollte aber unabhängig vom Entwicklungsteam existieren. Typischerweise übernimmt diese Aufgabe ein Qualifizierungsingenieur oder ein Qualifizierungsteam bei umfangreicheren Projekten. In der abschließenden Qualifizierungsphase können neben Standardqualifizierungsverfahren auch kundenspezifische Qualifizierungen durchgeführt werden.

Im Anschluss an die erfolgreich abgeschlossene Qualifizierung werden in der **Zertifizierungsphase** (Abbildung 6 auf Seite 37, ❸) die der durchgeführten Qualifizierung entsprechenden Zertifikate und Qualifizierungsberichte erzeugt. Als Grundlage dienen die während der Qualifizierung erzeugten Messergebnisse.

Weder **abschließende Qualifizierung** noch **Zertifizierung** müssen für Folgeauslieferungen wiederholt werden. Vorraussetzung dafür ist, dass im Anschluss keine Änderungen, z. B. aufgrund von Fehlerkorrekturen, durchgeführt wurden oder eine zusätzliche kundenspezifische Qualifizierung oder Zertifizierung erforderlich sind.

In der anschließenden **IP-Exportphase** (Abbildung 6 auf Seite 37, ❹) wird ein IP-Modul für die Auslieferung an einen IP-Nutzer vorbereitet. Es müssen die zu übermittelnden Integrationsdaten ausgewählt werden und qualitätssichernde Maßnahmen für den bevorstehenden IP-Import aufseiten des IP-Nutzers ergriffen werden.

Auf der Seite des IP-Nutzers lassen sich zwei Qualifizierungsphasen identifizieren. Während der **Eingangskontrolle** (Abbildung 6 auf Seite 37, ❺) möchte sich der IP-Nutzer von der Qualität des IP-Moduls überzeugen. Ist diese Phase erfolgreich abgeschlossen, schließt sich in der zweiten Phase die Aufbereitung des erworbenen IP-Moduls an den eigenen Entwurfsablauf an. Diese Phase wird **IP-Importphase** (Abbildung 6 auf Seite 37, ❻) genannt.

Die einzelnen Qualifizierungsphasen beim IP-Anbieter sind in Abbildung 6 auf Seite 37, ❶-❹ noch einmal grafisch in einem Ablaufplan dargestellt. Die beiden Qualifizierungsphasen aufseiten des IP-Nutzers sind in Abbildung 6 auf Seite 37, ❺-❻ abgebildet. Die verwendeten Symbole werden in der Legende in Abbildung 5 erläutert.

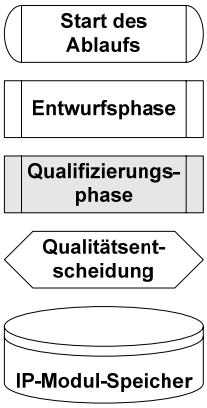


Abbildung 5: Legende zu Abbildung 6

Es wurden sechs Qualifizierungsphasen identifiziert:

1. Entwurfsbegleitende Qualifizierung
2. Abschließende Qualifizierung
3. Zertifizierung
4. IP-Export
5. Eingangskontrolle
6. IP-Import

Die ersten drei Phasen werden im weiteren Verlauf dieses Abschnitts detailliert besprochen. Die Phasen des IP-Exports, der Eingangskontrolle und des Imports werden in Kapitel 5 diskutiert.

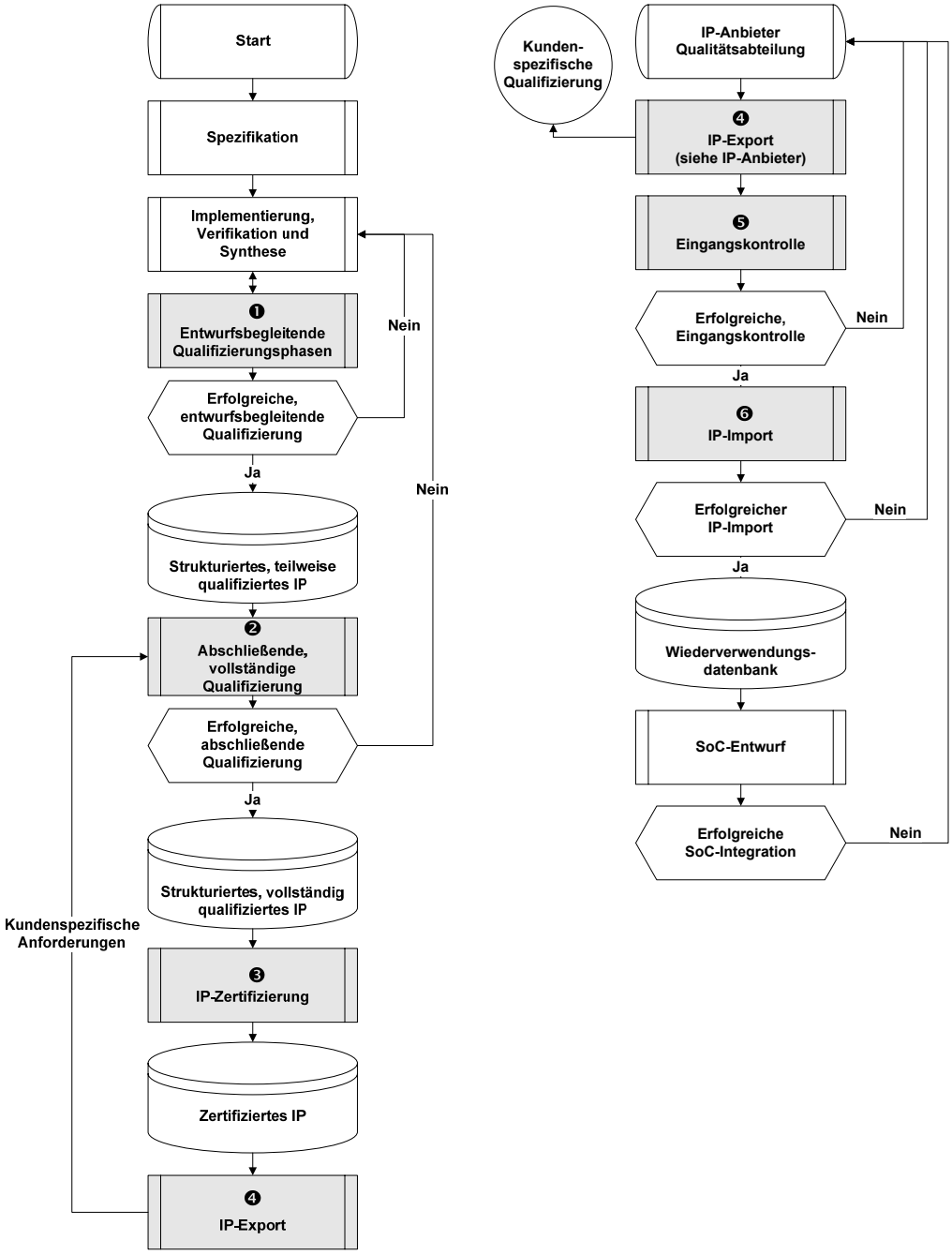


Abbildung 6: Qualifizierungsablauf beim IP-Anbieter (links) und IP-Auslieferung sowie IP-Nutzer (rechts)

4.2.1 Entwurfsbegleitende Qualifizierungsphase

Schon während der Entwurfsphase müssen die Entwickler Qualitätskriterien beachten und einhalten, da ein nachträgliches „Hineinprüfen“ von Qualität nur sehr aufwendig möglich ist. Dadurch werden die Entwickler in einen Lernprozess integriert, der die Qualität des Entwurfablaufs nachhaltig verbessert. Während der frühen Entwurfsphasen sollen die Entwick-

ler nur die Einhaltung der für ihre aktuelle Arbeit relevanten Qualitätskriterien prüfen. Deshalb werden nur bestimmte Qualitätsmessmethoden ausgeführt, die an die spezifischen Aufgaben in dieser Phase angepasst sind. Hierfür werden spezielle Qualitätsmessmethoden zusammengefasst, die sie bei folgenden Aufgaben unterstützen:

1. Implementierung
   Während der Implementierung wird die Übereinstimmung des Quellcodes mit internen Richtlinien und Standardrichtlinien untersucht.
2. Verifikation
   Verifikationsingenieure müssen durch Messmethoden hinsichtlich der Verifikationsabdeckung unterstützt werden.
3. Synthese
   Bei der Entwicklung von Soft-IP-Modulen wird eine Synthese als Qualifizierungsmaßnahme durchgeführt, um Synthesefehler zu vermeiden und die Übereinstimmung mit dem spezifizierten Zeitverhalten für eine Beispieltechnologie sicherzustellen.

Diese Aufteilung erlaubt eine effiziente Qualifizierung und eine größere Akzeptanz bei den Entwicklern.

Der Entwickler ist angewiesen, die von den Messmethoden angezeigten Qualitätsmängel zu korrigieren beziehungsweise Falschmeldungen mit einer Begründung zu dokumentieren. Dadurch wird ausgeschlossen, dass in nachfolgenden Qualifizierungsphasen unnötigerweise eine Qualifizierungsiteration ausgelöst wird. Der Begriff der Qualifizierungsiteration wird im nächsten Abschnitt erläutert. Spätestens vor Beginn der nächsten Entwurfsphase müssen vorhandene Qualitätsmängel beseitigt werden beziehungsweise dokumentiert sein.

Der Gewinn durch die Qualifizierungsmethode muss den Entwicklern plausibel gemacht werden. Nur so werden sie den zusätzlichen Aufwand betreiben und die Qualifizierungsmethode akzeptieren. Der Nutzen kann durch eine Reduzierung des Nacharbeitsaufwands während der Integration des IP-Moduls in einen SoC-Entwurf oder durch eine Verbesserung der Zufriedenheit bei den IP-Nutzern gezeigt werden. Die Möglichkeit, selbstständig Messungen durchführen und die Qualitätsverbesserung durch Korrektur des Quellcodes beeinflussen zu können, fördert zusätzlich die Akzeptanz bei den Entwicklern.

4.2.2 Abschließende Qualifizierungsphase

Am Ende einer IP-Modul-Entwicklung steht die abschließende Qualifizierung. Die bisherigen Qualifizierungen wurden von den Entwicklern selbst durchgeführt. Im Unterschied dazu wird die abschließende Qualifizierung von einem bisher nicht eingebundenen, unabhängigen Qualifizierungsingenieur oder Qualifizierungsteam durchgeführt. Kein IP-Modul darf ohne diese letzte Qualitätsfreigabe ausgeliefert werden. In dieser Phase wird das IP-Modul, wie bei einer Auslieferung an einen externen IP-Nutzer, firmenintern an den Qualifizierungsingenieur übergeben. Dabei werden alle Schritte, die auch ein externer Kunde durchführen muss,

abgearbeitet. Dadurch wird die Installation des IP-Moduls in einer „sauberen“ Umgebung simuliert. Die Entwicklungsumgebung mit ihren spezifischen Einstellungen wird bei diesem Verfahren als potentielle Fehlerquelle ausgeschlossen. Qualitätsmängel, die aufgrund dieser Einstellungen auftreten, können noch beim IP-Anbieter beseitigt werden.

Die abschließende Qualifizierungsphase eignet sich auch für die Qualifizierung bereits entwickelter Module, in deren Entwurfsablauf keine entwurfsbegleitende Qualifizierung durchgeführt wurde. In diesem Fall ist mit einem erheblich höheren Aufwand bis zum Erreichen des **Qualifizierungsziels** zu rechnen.

Wurden Kriterien während der Entwurfsphase nicht beachtet oder werden Qualitätsmängel erst während der abschließenden Qualifizierung entdeckt, wird eine so genannte „**kurze Qualifizierungsiteration**“ ausgelöst. Kurz ist sie, weil kein externer IP-Nutzer involviert ist. Eine kurze Iteration wird innerhalb der IP-Anbieterfirma durchgeführt und kann dadurch in der Regel schneller abgewickelt werden. Entsprechend gibt es auch eine „**lange Qualifizierungsiteration**“, die den externen IP-Nutzer einbezieht. Das **Qualifizierungsziel** ist, lange Qualifizierungsiterationen durch angemessene Qualifizierung beim IP-Anbieter zu vermeiden.

Qualifizierungsiterationen sollen nach Möglichkeit immer über den ursprünglichen Entwickler abgewickelt werden. Er soll Korrekturen durchführen, denn nur dadurch wird ein Lernprozess in Gang gesetzt, der nachhaltig vermeidet, dass dieser Qualitätsmangel erneut auftreten kann. Aufgrund des häufig notwendigen, entwurfsspezifischen Wissens bei Quellcodekorrekturen ist es ratsam, diese vom ursprünglichen Entwickler korrigieren zu lassen. Eine angestoßene Qualifizierungsiteration endet immer mit einer vollständigen Qualifizierung, die erneut Qualifizierungsiterationen auslösen kann.

Ist die Qualifizierung inklusive aller Qualifizierungsiterationen erfolgreich abgeschlossen, werden alle Entwurfsdaten des qualifizierten IP-Moduls in einer Wiederverwendungsdatenbank abgelegt und als auslieferbereit markiert. Nur auf diese Weise markierte Daten dürfen als Integrationsdaten exportiert und an einen IP-Nutzer ausgeliefert werden.

Die abschließende Qualifizierung wird zunächst als Standardqualifizierung durchgeführt, bevor das IP-Modul in der Wiederverwendungsdatenbank zur Auslieferung bereitgestellt wird.

4.2.2.1 Standardqualifizierung

Standardqualifizierung bezeichnet eine Qualifizierung nach einem Standard beziehungsweise Defacto-Standard. Neben der Standardqualifizierung gibt es die Möglichkeit, eine kundenspezifische Qualifizierung durchzuführen. Im Folgenden werden beide Möglichkeiten vorgestellt. Die VSIA ist zwar kein Standardisierungsorgan, ihre QIP-Bewertung kann derzeit jedoch als Defacto-Standard angesehen werden. Im Weiteren bezeichnet eine Standardqualifizierung eine Qualifizierung nach VSIA-QIP-Richtlinien.

Für die Standardqualifizierung können die Qualitätsmessmethoden der entwurfsbegleitenden Qualifizierungsphase verwendet werden. Allerdings müssen sie für eine vollständige Qualifizierung durch weitere Qualitätsmessmethoden ergänzt werden.

4.2.2.2 Kundenspezifische Qualifizierung

In einigen Fällen ist es notwendig, eine kundenspezifische Qualifizierung durchzuführen. Eine kundenspezifische Qualifizierung berücksichtigt spezifische Qualitätskriterien des IP-Nutzers, welche die Standardqualifizierung ergänzen oder aber auch im Konflikt mit Standardqualifizierungskriterien stehen und diese dann ersetzen.

Kundenspezifisch qualifizierte Entwurfsdaten dürfen daher nicht mit den Entwurfsdaten, die nach dem Standard qualifiziert wurden, zusammengeführt werden. Sie lassen sich am besten durch eine separate Kennzeichnung in der Wiederverwendungsdatenbank voneinander trennen.

Für eine kundenspezifische Qualifizierung sind zumindest Qualitätsmessmethoden der Standardqualifizierung neu zu konfigurieren. Gegebenenfalls müssen neue Qualitätsmessmethoden entwickelt werden. Eine kundenspezifische Qualifizierung sollte erst nach einer erfolgreichen Standardqualifizierung erfolgen.

Soll das IP-Modul direkt nach Fertigstellung der Entwicklung an einen Kunden ausgeliefert werden, kann die kundenspezifische Qualifizierung direkt in der abschließenden Qualifizierungsphase erfolgen. Bei bereits zur Auslieferung markierten IP-Modulen, die erneut verwendet werden sollen, ist oft eine Iteration über eine kundenspezifische Qualifizierung während der IP-Exportphase (Abschnitt 5.5.1) notwendig.

Ein IP-Modul bietet in der Regel definierte Anpassungsmöglichkeiten durch Parameter (z. B. Parität, Datenwortlänge, Paritätsbitposition usw.). Das Setzen dieser Parameter kann beim IP-Anbieter erfolgen, falls dieser über die genaue Wiederverwendung informiert ist. Eine solche Dienstleistung wird in der Regel als zusätzlicher Entwurfsservice abgerechnet. Bei komplexen IP-Modulen ist die Parametrisierung durch den IP-Anbieter zu empfehlen, da häufig ein spezifisches Wissen über die Wertebereiche und die Abhängigkeiten zwischen den Parametern erforderlich ist. Vereinzelt wird die Parametrisierung durch Werkzeuge unterstützt.

Teilweise möchte ein IP-Nutzer auch technologiespezifische Anpassungen vom IP-Anbieter durchführen lassen. So kann in dieser Phase beispielsweise eine Synthese mit einer vom IP-Nutzer bestimmten Technologiebibliothek vorgenommen werden.

4.2.3 IP-Zertifizierungsphase

Die Ergebnisse einer abschließenden Qualifizierung müssen in der Zertifizierungsphase aufbereitet und dokumentiert werden. Das Erstellen der Zertifikate erfolgt auf Grundlage der Daten, die während der vorausgegangenen Qualifizierungsphasen durch die Qualitätsmessmethoden generiert wurden. Damit wird das Ziel verfolgt, einen Beleg für die

Durchführung der jeweiligen standard- oder kundenspezifischen Qualifizierung zu erzeugen.

Im Falle der QIP-Qualifizierung ist unter dem Zertifikat das ausgefüllte QIP-Arbeitsblatt im Microsoft®-Excel-Format zu verstehen. Im Falle einer Vollständigkeitsqualifizierung hinsichtlich des VSIA Deliverables Documents [15] wird als Zertifikat die ausgefüllte Deliverables-Tabelle in einem beliebigen Format akzeptiert.

Die Daten zum Ausfüllen der notwendigen Formulare und Zertifikate werden von Berichtsgeneratoren aus den Qualitätsmessergebnissen extrahiert. Über angepasste Schnittstellen können die Berichtsgeneratoren das gewünschte Format erzeugen.

Neben den genannten Standardbewertungen wird in dieser Phase auch ein eventuell notwendiges Zertifikat über die Einhaltung kundenspezifischer Kriterien erstellt.

4.3 Qualitätsmessmethoden

In den vorausgegangenen Abschnitten wurde die Automatisierbarkeit der Qualitätsmessung dargestellt, sowie Phasen im Entwurfsablauf identifiziert, in denen Qualitätsmessmethoden effizient eingesetzt werden können. Eine Auswahl an Messmethoden wird in diesem Abschnitt detailliert besprochen.

Abbildung 7 zeigt eine schematische Darstellung, wie mit automatisierten Qualitätsmessmethoden unqualifizierte IP-Module in qualifizierte IP-Module überführt werden können. Verschiedene Qualitätsmessmethoden werden auf teilweise oder vollständig unqualifizierte IP-Module angewendet [2]. Das Ergebnis einer angewendeten Teilmenge von Qualitätsmessmethoden ist ein teilweise qualifiziertes IP-Modul. Dies ist beispielsweise in der entwurfsbegleitenden Qualifizierungsphase der Fall. Alle Qualitätsmessmethoden müssen angewendet werden, um ein vollständig qualifiziertes IP-Modul zu erhalten (z. B. abschließende Qualifizierungsphase). Dabei ist zu beachten, dass nicht alle Qualitätskriterien automatisch überprüft werden können (Abschnitt 4.1).

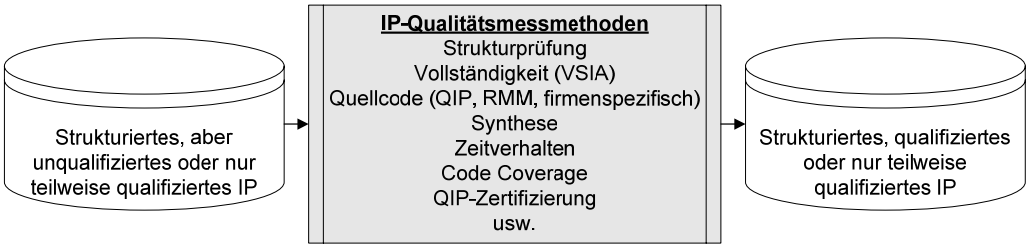


Abbildung 7: Qualitätsmessmethoden

Daher müssen neben den automatischen auch manuelle Qualitätsmessmethoden installiert sein, um ein IP-Modul vollständig, beispielsweise nach VSIA-QIP, zu qualifizieren. Manuelle Messmethoden sind häufig dann erforderlich, wenn der Sinn und die Intention von Daten erkannt werden muss. Das ist beispielsweise bei der Überprüfung, ob sinnvolle

Kommentare im Quellcode enthalten sind, ob sinnvolle Signalnamen vergeben wurden usw., erforderlich.

Es ist auch nicht immer möglich, einen Qualitätsmangel eindeutig festzustellen. Es ist aber möglich, Konstrukte zu identifizieren, die mit hoher Wahrscheinlichkeit einen Qualitätsmangel bedeuten. Untersucht man beispielsweise das Qualitätskriterium: „Enthält der Entwurf unvermeidbare asynchrone Logik?“ in QIP Zeile 224, fällt auf, dass mit Lintern asynchrone Logikkonstrukte entdeckt werden können. Jedoch kann nur der Entwickler entscheiden, ob dieses Konstrukt unvermeidbar ist.

Die Aufgabe einer Qualitätsmessmethode ist es, in solchen Fällen den Entwickler auf einen potentiellen Qualitätsmangel aufmerksam zu machen. Die endgültige Entscheidung, ob es sich um einen tatsächlichen Qualitätsmangel handelt, liegt allerdings beim Entwickler beziehungsweise Qualifizierungsingenieur. In solchen Fällen können daher nur teilautomatisierte Qualitätsmessmethoden entwickelt werden.

4.3.1 Quellcodequalität

Die Quellcodequalität hat Einfluss auf die Wiederverwendbarkeit, Wartbarkeit und Portabilität von IP-Modulen. Wobei unter Portabilität sowohl eine Portierung zwischen Entwicklungsumgebungen (z. B. von IP-Anbieter zu IP-Nutzer), als auch die Portierung zwischen Werkzeugen (z. B. Wechsel des Synthesewerkzeugs, Simulators usw.) zu verstehen ist. Häufig bedingt eine Portierung in eine neue Entwicklungsumgebung auch die Portierung auf die in der neuen Entwicklungsumgebung enthaltenen Werkzeuge. Bestimmte HDL-Quellcodekonstrukte lassen sich beispielsweise nur oder zumindest effizienter mit bestimmten Werkzeugen verarbeiten. Ein Beispiel ist die Synthese multidimensionaler Arrays. Einige Synthesewerkzeuge können Arrays nur bis zur zweiten Dimension synthetisieren.

Aus den automatisierbaren Qualitätskriterien, die den Quellcode betreffen, wurden Codierrichtlinien abgeleitet. Für diese wurden Qualitätsmessmethoden entwickelt. Die Messungen können zum Teil mit vorhandenen Werkzeugen (z. B. Linter) beziehungsweise mit selbst entwickelten Parsern durchgeführt werden. Quellcodeüberprüfungen sind in der entwurfsbegleitenden Phase sowie in der abschließenden Qualifizierungsphase integriert, um den Quellcode auf Einhaltung der Codierrichtlinien zu prüfen. Folgende Ziele werden damit verfolgt:

- Übereinstimmung mit bedeutenden QIP-Kriterien: Beispielsweise wird auf die Einhaltung eines synchronen Entwurfs geachtet, nicht aber auf vorgeschlagene Namensregeln.
- Synthetisierbarkeit des Quellcodes durch Prüfung gegen den synthetisierbaren Teil der VHDL-Spezifikation (IEEE 1076.6); weitere Synthetisierbarkeitsprüfungen werden in Abschnitt 4.3.3 beschrieben.
- Weitere Codierrichtlinien z. B. für FPGAs, VHDL-nach-Verilog Übersetzung usw. werden beachtet.

Ein programmierbarer Linter wurde in das IP-Qualifizierungsframework integriert. Dabei wurden so weit wie möglich die bereits definierten und konfigurierbaren Regelprüfungen verwendet. Viele Linter überprüfen Kriterien wie beispielsweise: „keine kombinatorischen Rückkopplungen“, „keine `File`-Deklarationen im `Architecture`-Body“ usw. bereits durch ihre Standardregeln. Dadurch wird ein großer Teil des Quellcodestandards umgesetzt. Der aktuell verwendete Linter führt zusätzlich eine Logiksynthese durch und kann daher Aussagen über Signalübergänge zwischen Bereichen mit unterschiedlichem Takt (engl. clock domain crossing) machen. Zusätzlich wird die Möglichkeit des Linters genutzt, eigene Regelprüfungen zu programmieren, um die Abdeckung des Quellcodestandards zu verbessern. Mit der Implementierung des Quellcodestandards generiert der Linter eine Liste der Regelverletzungen. Diese umfangreichen Daten werden von einem Berichtsgenerator in der Zertifizierungsphase analysiert und zusammengefasst. Für den internen Qualitätsbericht werden alle Regelverletzungen erfasst und die einzelnen Verletzungen nach Schweregrad und Häufigkeit ihres Auftretens sortiert. Weitere Zusammenfassungen werden an das zu erstellende Zertifikat angepasst und dienen damit als Grundlage für Zertifizierungsberichte (z. B. QIP). Der Entwickler muss jede Regelverletzung im Quellcode entsprechend korrigieren beziehungsweise rechtfertigen und im Qualitätsbericht dokumentieren. Beispielsweise meldete ein früher verwendeter Linter durch eine vordefinierte Standardregel fälschlicherweise beim Codekonstrukt

```
1  if (clk'event and clk = '1') then
```

Quellcode 1: Linter Falschmeldung

den Fehler, dass die VHDL-Entwurfseinheit intern einen Takt generiert. Eine solche Falschmeldung wurde mit dem Verweis auf den bekannten Mangel des Werkzeugs dokumentiert. Da nicht alle Zertifizierungskriterien automatisch geprüft werden können, müssen die nicht im Linter beziehungsweise in das Qualifizierungsframework integrierten Prüfungen für eine vollständige Zertifizierung manuell durchgeführt werden (Code-Überprüfung).

4.3.2 Validierung

Die Simulation ist die gängigste Validierungsart. In diesem Fall wird die Qualität der Simulation anhand der Simulationsabdeckung (Code-Coverage-Messungen [53]) beurteilt.

Dafür werden die verschiedenen Code-Coverage-Messverfahren Anweisungs- (engl. statement), Verzweigungs- (engl. branch), Bedingungs- (engl. condition) und Pfadabdeckung (engl. path) benutzt, um die Anforderungen der VSIA in QIP Version 1.11 auf Tabellenblatt „Digital Soft-IP“ in den Zeilen 263 bis 269 bezüglich Code-Coverage zu erfüllen. Im Entwurfsablauf, also während der entwurfsbegleitenden Qualifizierung, werden Code-Coverage-Messungen durch Setzen einer Variablen im Simulationskonfigurationsskript aktiviert. Zunächst sollen die Entwickler wegen der höheren Simulationsperformanz ohne Coverage-Messungen simulieren,

bis der Entwurf alle Testfälle fehlerfrei absolviert. Danach werden die Code-Coverage-Messungen aktiviert. Werden die geforderten Abdeckungsergebnisse nicht erreicht, müssen Testfälle hinzugefügt oder geändert werden, bis das geforderte Ergebnis erreicht ist und alle Testfälle fehlerfrei bestanden werden. Bevor das IP-Modul als qualifiziert markiert wird, muss der Qualifizierungsingenieur die Coverage-Messungen für alle Entwurfsteile in der abschließenden Qualifizierungsphase wiederholen. In Abschnitt 6.3.4.1 wird eine konkrete Implementierung der automatischen Code-Coverage-Messung beschrieben.

4.3.3 Entwurfsqualität

Der Begriff Entwurfsqualität fasst Qualitätseigenschaften der synthetisierten Schaltung zusammen. Den folgenden Eigenschaften wird besondere Aufmerksamkeit während des IP-Modul-Entwurfs geschenkt:

1. **Fläche/Gatteranzahl:** Durch Synthese und Abbildung des Entwurfs auf eine technologiespezifische Zellenbibliothek werden Informationen über Gatteranzahl und Flächenbedarf ermittelt.
2. **Testbarkeit:** Ein Scan-Pfad wird in die synthetisierte Schaltung eingefügt, und es werden automatisch Testmuster generiert. Mit Scan-Pfad und Testmustern wird die Testabdeckung bestimmt.
3. **Leistungsaufnahme:** Mit den im vorangegangenen Schritt generierten Testmustern, werden auch Werte für die Leistungsaufnahme der synthetisierten Schaltung geschätzt. Da Testmuster so generiert werden, dass möglichst viele interne Knoten gleichzeitig schalten, wird das Ergebnis als ungünstigster Fall angenommen. Diese Annahme wurde aber nicht bewiesen. Bevorzugt sollten anwendungsspezifische Testmuster verwendet werden, falls solche verfügbar sind.
4. **Zeitverhalten:** Die Syntheseberichte werden auf jegliche Art von Verletzungen des Zeitverhaltens hin untersucht. Ein IP-Modul besteht die abschließende Qualifizierung nur, wenn das spezifizierte Zeitverhalten in Form von Syntheseconstraints eingehalten ist. Das Zeitverhalten wird in vielen Fällen durch den typischen Anwendungstakt spezifiziert.

Um die genannten Qualitätskriterien überprüfen zu können, wird ein Soft-IP-Modul synthetisiert. Dadurch wird zum einen die Synthetisierbarkeit des IP-Moduls sichergestellt. Zum anderen kann anhand des Berichts, der während der Synthese erzeugt wird, geprüft werden, ob die synthetisierte Schaltung alle Qualitätskriterien erfüllt, die auf dieser Abstraktionsebene relevant sind.

4.3.4 Integritätsprüfung

Während der Integritätsprüfung werden durch Quellcodeanalyse aller Dateien, Abhängigkeiten und Verweise zwischen den Dateien aufgedeckt und validiert. Für die Analyse werden spezielle Parser eingesetzt, die die Abhängigkeiten anhand von formatspezifischen Schlüsselwörtern und Konstrukten erkennen. Die Abhängigkeiten werden anschließend überprüft. Bestanden ist die Prüfung, wenn alle referenzierten Dateien

zum Umfang des IP-Moduls gehören. Am Beispiel eines Synopsys Design-Compiler-Skripts in Quellcode 2 wird das Vorgehen verdeutlicht. Ein Parser identifiziert zunächst die schwarz hinterlegten Schlüsselwörter, die Abhängigkeiten von weiteren Dateien, aber nicht notwendigerweise des gleichen Dateiformats, spezifizieren. Anschließend werden die virtuellen Dateipfade in derselben Codezeile bestimmt. Die virtuellen Dateipfade werden durch Ersetzen von Variablen durch deren Werte (dunkelgrau hinterlegte Variablennamen) zu realen Dateipfaden, die entweder absolut oder relativ definiert sind.

```
1  /* General variable settings */
2  SYN_SELECTOR = get_unix_variable("SYN_SELECTOR")
3  OS_SELECTOR = get_unix_variable("OS_SELECTOR")
4
5  /* General variable settings */
6  DEFAULT_WORK = "./work/" + SYN_SELECTOR + "/" +
   OS_SELECTOR
7
8  netlist_path = "./netlist/"
9
10 define_design_lib work -path DEFAULT_WORK
11 define_design_lib synopsys -path synopsys_root +
   "/packages/synopsys/lib"
```

Quellcode 2: Ausschnitt aus einem Synopsys Design-Compiler-Skript

In einem weiteren Schritt werden die realen Dateipfade überprüft, ob sie zum Integrationsdatenumfang des IP-Moduls gehören. Das bedeutet, dass sie unterhalb des IP-Modulquellverzeichnisses im Verzeichnisbaum abgelegt sein müssen. Nur wenn das für alle Dateiabhängigkeiten zutrifft, ist die Integritätsprüfung bestanden. Ausnahmen bilden Dateien, die in jedem relevanten Entwicklungsablauf als vorhanden vorausgesetzt werden können. Dazu zählen beispielsweise die VHDL-Bibliotheken `ieee` und `std`.

Mit den beschriebenen Qualitätsmessmethoden kann ein großer Teil der Standardqualifizierung automatisiert werden. Mit den bisher beschriebenen Methoden wird die Qualität eines IP-Moduls während der Entwicklung beziehungsweise nach Entwicklungsabschluss gemessen und kann entsprechend des Messergebnisses verbessert werden, bis das festgesetzte Qualifizierungsziel erreicht ist. Weitere wichtige Aspekte der IP-Qualifizierung sind die Erhaltung der Qualität nach Abschluss der Entwicklung während des IP-Austauschs, der Auslieferungsphase an einen IP-Nutzer und des Imports inklusive eventueller Anpassungen in die Wiederverwendungsdatenbank des IP-Nutzers.

4.3.5 Konsistenzprüfung

Durch die Analysen während der Integritätsprüfung ist gewährleistet, dass alle benötigten Dateien, beispielsweise für die Kompilierung, vorhanden sind. Eine Konsistenzprüfung ist notwendig, um Fehler, die nicht in der Kompilierphase entdeckt werden, dateiformatübergreifend zu erkennen.

Solche Inkonsistenzen treten nicht ausschließlich bei der parallelen Entwicklung von Syntheseskripten und HDL-Quellcode auf.

Beispielsweise werden durch Synthesebeschränkungen (engl. constraints) bestimmte Zeitverhalten zwischen Ein- und Ausgängen von Komponenten, für bestimmte Signale definiert. Zur Überprüfung beziehungsweise Einhaltung dieser Beschränkungen während der Synthese müssen die Objektnamen der kompilierten HDL-Dateien mit denen im Syntheseskript übereinstimmen. Werden Änderungen im HDL-Quellcode vorgenommen, die nicht in das Syntheseskript einfließen, wird evtl. für Pfade ein irrelevantes Zeitverhalten erzwungen oder das spezifizierte Zeitverhalten für relevante Pfade nicht eingehalten. Ursachen für inkonsistente IP-Module sind nicht auf Syntheseskripte beschränkt. Ebenfalls sind die Verifikationsumgebung und alle weiteren Prozesse während des Entwurfs betroffen, die auf konsistente Verweise zwischen unterschiedlichen Dateiformaten angewiesen sind.

Ähnlich wie bei der Integritätsprüfung werden während der Konsistenzprüfung mit speziellen Parsern Konstrukte entdeckt, die relevante Verweise modellieren. Mithilfe der automatischen Konsistenzprüfung können ungültige Verweisquellen (z. B. Beschränkungen im Syntheseskript) entdeckt werden, für die das Verweisziel (z. B. im HDL-Quellcode) fehlt. In einem solchen Fall muss der Entwickler entscheiden, ob die Verweisquelle überflüssig geworden ist und überflüssiger Quellcode gelöscht wird oder ob sie an ein verändertes Verweisziel angepasst wird. Fehler aufgrund von wechselseitig vertauschten Objektnamen können nicht automatisch aufgedeckt werden, da ein Verweisziel existiert, auch wenn dieses falsch ist. Die Entscheidung, dass ein falsches Verweisziel vorliegt, kann nicht generell automatisiert werden. Solche Inkonsistenzen können nur durch manuelle Prüfungen gefunden werden.

5 IP-Austausch und -Auslieferung

Im vorigen Kapitel wurde die Methode zur Entwicklung qualifizierter IP-Module beschrieben. Bei Beachtung dieser Methode wird die Wiederverwendbarkeit von IP-Modulen verbessert und gleichzeitig das Integrationsrisiko beim Systementwurf mit IP-Modulen reduziert. Im Folgenden werden die Phasen eines IP-Moduls untersucht, die sich an die Entwicklung anschließen.

In Abschnitt 5.1 hat der Autor den gesamten Austauschprozess untersucht. Basierend auf diesen Ergebnissen werden im darauf folgenden Abschnitt 5.2 die Anforderungen an ein einheitliches IP-Format diskutiert. Das verfolgte Ziel der Diskussion ist die Automatisierung der Qualifizierungsphasen, eine Automatisierung der Auslieferung und die Erhaltung der Qualität bei der Auslieferung. Des Weiteren schlägt der Autor in Abschnitt 5.3 das IPQ-Format als einheitliches IP-Format vor, das diese Anforderungen erfüllt. Detailliert wird das vom Autor entwickelte IPQ-Integrationsdatenformat in Abschnitt 5.4 betrachtet, das Thema der weiteren Arbeit ist. Im darauf folgenden Abschnitt 5.5 werden die bisher noch unbehandelten Qualifizierungsphasen: der IP-Export, die Eingangskontrolle und der IP-Import, ausführlich behandelt. Die in diesen Phasen verwendeten Methoden basieren auf dem IPQ-Format und wurden ebenfalls vom Autor entwickelt.

5.1 IP-Austausch – ein Überblick

Der Begriff „IP-Austausch“ impliziert einen Informations- beziehungsweise Datenfluss in zwei Richtungen: sowohl vom IP-Nutzer zum IP-Anbieter als auch umgekehrt. Daher wird dieser Begriff für den Gesamtprozess verwendet, der die IP-Suche (engl. IP retrieval) eines geeigneten IP-Moduls in einer Wiederverwendungsdatenbank (Informationsfluss vom IP-Nutzer zum IP-Anbieter) und die IP-Auslieferung (Datenfluss vom IP-Anbieter zum IP-Nutzer) einschließt. Der IP-Austausch wird in mehrere voneinander abhängige Teilaufgaben unterteilt. Diese Aufgaben müssen ausgeführt werden, bis ein IP-Modul schließlich in einen SoC-Entwurf integriert werden kann. In Abbildung 8 auf Seite 49 ist der Ablauf des IP-Austauschs mit den einzelnen Teilaufgaben dargestellt, die im Folgenden erläutert werden.

Ein Entwickler stößt den IP-Austausch an, indem er nach einem geeigneten IP-Modul für den SoC-Entwurf sucht. Aus der Spezifikation des benötigten Blocks leitet der Entwickler die Eingabedaten für die Suchmaske der Wiederverwendungsdatenbank ab (siehe auch Abbildung 6 rechts auf Seite 37). Zunächst wird die Anfrage an die firmeninterne Wiederverwendungsdatenbank geleitet (Abbildung 8 auf Seite 49, interne IP-Suche). Die Wiederverwendungsdatenbank enthält Daten der intern verfügbaren IP-Module, die entweder firmenintern entwickelt wurden oder von externen IP-Anbietern eingekauft und in die Datenbank importiert wurden. Zum einen enthält die Datenbank charakterisierende Daten, die

ein IP-Modul für die Suchalgorithmen beschreiben, und zum anderen die Integrationsdaten [21]. Die Integrationsdaten sind an den firmenintern verwendeten Entwurfsablauf angepasst. Das bedeutet, dass die Integrationsdaten in eine firmenintern verwendete Struktur eingebettet sind. Charakterisierungs- und Integrationsdaten sind untereinander verknüpft. Dadurch kann nach einer erfolgreichen Suche, basierend auf den Charakterisierungsdaten, auf die Integrationsdaten zugegriffen werden.

Möglicherweise ist aber kein geeignetes IP-Modul intern verfügbar. In diesem Fall wird die Anfrage automatisch an eine externe IP-Charakterisierungsdatenbank eines IP-Anbieters beziehungsweise IP-Maklers weitergeleitet ([17], [18]) (Abbildung 8 auf Seite 49, externe IP-Suche beziehungsweise Abbildung 6 links auf Seite 37, zertifiziertes IP). Der interne Datenbankserver agiert in diesem Fall als Proxyserver. Die externen IP-Charakterisierungsdatenbanken bieten ebenfalls eigene Suchmethoden und -masken an oder können direkt, beispielsweise über Webservices [21], abgefragt werden. Als Ergebnis erhält der Entwickler eine Liste der verfügbaren IP-Module, die der Anfrage entsprechen.

Findet der Entwickler ein geeignetes IP-Modul, das den Anforderungen entspricht, müssen sowohl Integrations- als auch Charakterisierungsdaten für den Import in die firmeninterne Wiederverwendungsdatenbank vorbereitet werden. Voraussetzung dafür ist die Auslieferung der IP-Daten vom IP-Anbieter an den IP-Nutzer.

In dieser Arbeit werden keine (vertrags)rechtlichen Aspekte diskutiert. Selbstverständlich müssen sie geklärt werden, bevor ein IP-Anbieter Integrationsdaten für einen IP-Nutzer freigibt. Beispielsweise bietet Virtual Component Exchange (VCX) eine Online-Plattform für die Klärung rechtlicher Belange an [22]. Die Verhandlungen über die Online-Plattform führen zur Einigung über einen Vertrag, indem auch die auszuliefernde Teilmenge der Entwurfsdaten, also der Umfang der Integrationsdaten definiert ist. Eine Defacto-Standard für den Auslieferungsumfang wurde von der VSIA Virtual Component Transfer (VCT) DWG erarbeitet [15]. Kundenspezifische Abweichungen vom Standard können während der Vertragsverhandlungen geklärt werden.

Der folgende Auslieferungsprozess teilt sich in die drei Phasen IP-Export (siehe auch Abbildung 6 rechts auf Seite 37, ❹), IP-Übermittlung und IP-Import (siehe auch Abbildung 6 rechts auf Seite 37, ❻) auf. Während der IP-Übermittlung wechselt das IP-Modul vom Verantwortungsbereich des IP-Anbieters in den des IP-Nutzers. Daher ist an dieser Stelle eine Eingangskontrolle (siehe auch Abbildung 6 rechts auf Seite 37, ❺) aufseiten des IP-Nutzers vorgesehen. Im Anschluss an die Auslieferung und Aufnahme in die IP-Nutzer Wiederverwendungsdatenbank (IP-Import) schließen sich beim IP-Anbieter Kompatibilitätsprüfungen und Anpassungen des IP-Moduls an die verwendete Entwurfsplattform an.

Eine Automatisierung firmenübergreifender Prozesse ist nur möglich, wenn es ein gemeinsames Verständnis von den ausgetauschten Daten gibt. Dies zu ermöglichen ist die Aufgabe eines IP-Formats. Die Anforderungen an ein solches Format werden im folgenden Abschnitt diskutiert.

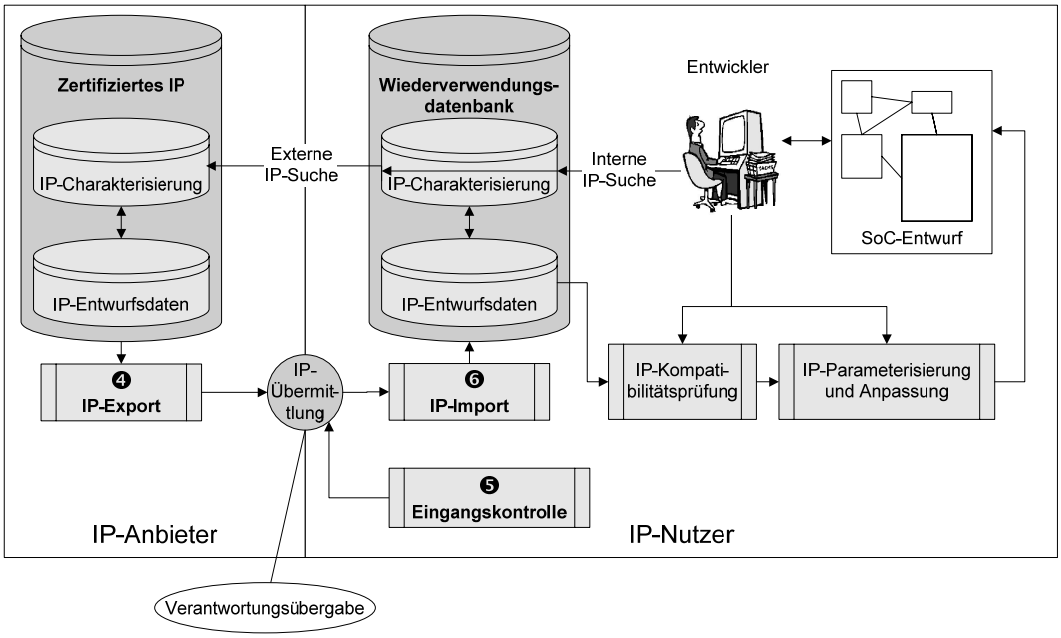


Abbildung 8: IP-Austausch Szenario

5.2 IP-Format Anforderungen

In diesem Abschnitt werden die Anforderungen an ein IP-Format diskutiert, welches die Automatisierung der Qualifizierungsmethoden ermöglicht. Im Folgenden wird daher vom IPQ-Format (IP-Qualifizierungsformat) gesprochen. Des Weiteren trägt das IPQ-Format zu einer Beschleunigung der IP-Auslieferung bei und bietet dem IP-Nutzer die Möglichkeit ein IP-Modul automatisch in seine Wiederverwendungsdatenbank zu importieren und an seinen Entwurfsablauf anzupassen.

Dieser Abschnitt konzentriert sich im Anschluss an einen Überblick über das IPQ-Gesamtformat auf das Integrationsdatenformat. Der interessierte Leser findet weitere Informationen zum Charakterisierungsformat in [19] und [21]. Integrations- und Charakterisierungsformat sind Teil des IPQ-Gesamtformats.

5.2.1 IP-Struktur

In den meisten Fällen, insbesondere aber bei der Wiederverwendung von extern entwickelten IP-Modulen, unterscheiden sich die IP-Strukturen in den Wiederverwendungsdatenbanken zwischen IP-Anbieter und IP-Nutzer. Die enthaltenen, meist firmenspezifischen IP-Strukturen können nicht einfach geändert werden, da sie an den Entwurfsablauf angepasst sind. Diese Tatsache macht eine Anpassung der IP-Struktur an die Struktur des IP-Nutzers während des IP-Imports notwendig. Eine Angleichung der IP-Struktur zwischen IP-Anbieter und IP-Nutzer ist keine Lösung, da sie erhebliche Anpassungen des Entwurfablaufs nach sich ziehen würde. Andererseits wäre eine Angleichung zwischen zwei Parteien nur eine sehr eingeschränkte Lösung des Problems. Weitere Angleichungen müssten folgen, was eine Gesamtlösung praktisch unmöglich macht. Eine automatisierte Lösung zur Unterstützung allgemeiner IP-Anbieterstrukturen ist zu

aufwendig, wenn nur jeweils eine Zielstruktur bei der direkten Umstrukturierung zwischen einem IP-Anbieter und einem IP-Nutzer unterstützt wird. Aufgrund dieser wirtschaftlichen und praktischen Erwägungen ist nicht mit einer Angleichung der unterschiedlichen IP-Strukturen als Gesamtlösung des Problems zu rechnen. Gegen eine Anpassung aufseiten des IP-Anbieters spricht, dass der IP-Anbieter in der Regel kein detailliertes Wissen über die beim IP-Nutzer verwendete IP-Struktur hat. Daher wird der IP-Anbieter das IP-Modul im Allgemeinen in seiner eigenen IP-Struktur an den IP-Nutzer ausliefern. Die Anpassung wird dann in der Regel aufseiten des IP-Nutzers manuell durchgeführt. Eine Ausnahme stellt die virtuelle Auslieferung dar, die in Abschnitt 5.5.2 erläutert wird. Sie erfolgt vollständig aufseiten des IP-Anbieters. Eine sinnvolle Automatisierungslösung muss daher auf der Unabhängigkeit der IP-Strukturen aufbauen.

5.2.2 Unabhängigkeit der IP-Struktur

Interessant für beide Parteien ist eine Lösung, die es einem IP-Anbieter erlaubt, in ein unabhängiges Zwischenformat zu exportieren, das dem IP-Nutzer einen automatischen Import erlaubt. Mit einer solchen Lösung können unabhängige IP-Strukturen weiterhin bestehen bleiben, Entwurfsabläufe müssen nicht geändert werden und beim Export beziehungsweise Import muss nur das Zwischenformat als einziges, weiteres Format neben der eigenen IP-Struktur unterstützt werden. Auf dieser Grundlage lassen sich automatisierte Lösungen für den Export und Import entwickeln. Um die Unabhängigkeit der IP-Strukturen und eine automatisierte Strukturanpassung zu ermöglichen, muss die Semantik der Integrationsdaten im Zwischenformat erfasst werden.

5.2.3 Semantik der IP-Struktur

IP-Module müssen mit maschinenlesbaren Informationen über ihre Struktur, die enthaltenen Datenformate und ihrer Verwendung im Entwurfsablauf ausgeliefert werden. Das ermöglicht eine automatische Umstrukturierung und Anpassung während der Importphase. Zusätzlich ermöglichen solche Zusatzinformationen automatische Qualitätsmessungen. Die QIP-Analyse in Abschnitt 4.1 zeigt, dass sich von den (teil-)automatischen Prüfungen ungefähr 27 % auf die IP-Struktur beziehen. Dazu zählen Messungen, welche die Vollständigkeit des IP-Moduls betreffen. Beispielsweise lässt sich die Frage, ob eine bestimmte Dokumentation Teil der IP-Auslieferung ist, nur werkzeuggestützt beantworten, wenn die Qualifizierungswerkzeuge die Semantik der Integrationsdaten erkennen können, also Wissen darüber haben, an welcher Stelle der IP-Struktur unter welchem Dateinamen, die entsprechende Dokumentation vorhanden sein muss, falls sie verfügbar ist. Es ist also absolut notwendig, dass ein IP-Format sowohl IP-Anbietern als auch IP-Nutzern erlaubt, solche Strukturabhängigkeiten automatisch zu prüfen. Ein IP-Format sollte aber nicht zur Folge haben, dass alle Werkzeuge des Entwurfsablaufs dieses Format unterstützen müssen.

5.2.4 Vermeidung einer Werkzeug-API

Durch einen automatisierten Import in die spezifische IP-Struktur des IP-Nutzers wird vermieden, dass alle Werkzeuge im Entwurfsablauf eine standardisierte Schnittstelle (API) zum Zugriff auf die IP-Daten unterstützen müssen. Die Aufgabe des IP-Imports ist die Daten des IP-Moduls derart an den Entwurfsablauf des IP-Nutzers anzupassen, dass die verwendeten Werkzeuge ohne Modifikation auf die Integrationsdaten zuzugreifen können. Zusätzlich muss der Entwurfsablauf dokumentiert werden.

5.2.5 Dokumentation des Entwurfablaufs

Integrationsdaten müssen von bestimmten Werkzeugen verarbeitet werden. Die Daten können nur dann automatisch den richtigen Werkzeugen zugeführt werden (Integrationsunterstützung), falls der Entwurfsablauf maschinenlesbar dokumentiert wird. Dadurch wird beispielsweise dem IP-Nutzer während der Anpassung an den Entwurfsablauf beim Import gemeldet, dass das bei der Entwicklung verwendete Werkzeug in seinem Entwurfsablauf nicht vorhanden ist und von daher mit unterschiedlichen Qualifizierungsergebnissen gerechnet werden muss. Außerdem kann automatisch ein vergleichbares Werkzeug, welches im Entwurfablauf vorhanden ist vorgeschlagen werden. Beispielsweise ein Synthesewerkzeug eines anderen Herstellers oder eines, das sich lediglich in der Version unterscheidet.

5.2.6 Nachvollziehbarkeit der Qualifizierungsergebnisse

Ein weiterer Punkt ist die Nachvollziehbarkeit der Qualifizierungsergebnisse. Die Dokumentation dieser Ergebnisse ist ein wesentlicher Bestandteil eines IP-Moduls. Häufig werden Qualifizierungsergebnisse durch werkzeugspezifische Berichtsdateien ergänzt, die ohne Informationen über das erstellende Werkzeug, seiner Version und Konfiguration nicht während der gesamten Lebensdauer eines IP-Moduls reproduziert werden können. Der IP-Nutzer hat in der Regel kein ausreichendes Wissen über den Entwurfsablauf bei der IP-Entwicklung. Daher ist es sowohl für die Dokumentation, als auch für die Transparenz erzielter Qualifizierungsergebnisse wesentlich, den verwendeten Entwurfsablauf dokumentieren zu können.

5.2.7 Qualitätsmessungen und -dokumentation

Derzeit gibt es kein Format, das Qualifizierungsmethoden effizient unterstützt. In der Praxis werden manuelle Anpassungen nach dem IP-Import und manuelle Codekontrollen, unterstützt durch Linter, durchgeführt. Das ist nicht nur mühsam, sondern es können bei der vorherrschenden Praxis auch nicht alle Qualitätskriterien, die in Kapitel 2.5.3 beschrieben sind, abgedeckt werden. Ein IP-Format muss daher zum einen die Qualifizierung, und zum anderen die Dokumentation von Qualifizierungsergebnissen unterstützen. Notwendige Zwischenschritte der Qualifizierung können so festgehalten werden und auch an Dritte übermittelt werden, um dort die endgültige Qualifizierung durchzuführen. Als ein Beispiel sei hier die Exportphase genannt, in der Informationen aufseiten des IP-Anbieters

generiert werden müssen, aber erst beim IP-Nutzer in der Importphase zur Umstrukturierung weiterverarbeitet werden können. Zusätzlich ist es notwendig, die in den Entwurfs- beziehungsweise Integrationsdaten dokumentierten Informationen für die Qualifizierung zu nutzen.

5.2.8 Vollständigkeits- und Integritätsmessungen

Aktuelle IP-Entwürfe bestehen bereits aus mehreren hundert Dateien, die voneinander abhängig sind. Beispielsweise sind Syntheseskripte von den HDL-Quellcodedateien abhängig, HDL-Testbenchdateien hängen von I/O-Testmustern ab usw. Es ist notwendig diese Abhängigkeiten für Qualifizierungszwecke unabhängig von den üblicherweise verarbeitenden Werkzeugen auswerten zu können, um automatische Vollständigkeits- und Integritätsprüfungen, wie sie von der VSIA gefordert werden, zu ermöglichen [7]. Es ist wichtig, dass solche Abhängigkeiten in einem IP-Format modelliert werden.

5.2.9 Modellierung von Abhängigkeiten

Zwar melden HDL-Compiler fehlende HDL-Dateien und vereinzelt können Linter Integritätsprüfungen zwischen Syntheseskripten und HDL-Dateien durchführen [12], aber es gibt bisher kein Format, das generell Abhängigkeiten zwischen allen in einem IP-Modul vorkommenden, unterschiedlichen Dateiformaten darstellen kann. Die Modellierung dieser Abhängigkeiten ist aber notwendig, um die bereits genannten Integritäts- und Konsistenzmessungen sowie Anpassungen ungültiger Abhängigkeiten nach einer Umstrukturierung zu ermöglichen. Bereits während der Entwicklung elektronischer Entwürfe entstehen Inkonsistenzen, wenn beispielsweise Syntheseskripte nicht manuell an geänderte Signal- oder Komponentennamen des HDL-Entwurfs angepasst werden. Durch die Möglichkeit Abhängigkeiten formatübergreifend zwischen Dateien zu modellieren, ist auch die Möglichkeit zur automatischen Anpassung während des Imports gegeben. Zusätzlich kann beispielsweise auch die Kompilierreihenfolge aus der Abhängigkeitsstruktur bestimmt werden. Diese Information muss einigen Werkzeugen (z. B. [73]) bereitgestellt werden. Bedingt durch die Werkzeugunterstützung, die ein solches Format ermöglicht, wird sowohl die Qualifizierungs-, als auch die Importdauer verkürzt.

5.2.10 Dauer des IP-Imports

Wegen der unterschiedlichen IP-Strukturen und nicht automatisch anpassbarer Dateiabhängigkeiten dauert der Import eines IP-Moduls mittlerer Größe derzeit ungefähr eine Personenwoche. Dieser Zeitaufwand wird benötigt, weil keine generelle Automatisierung mit den verfügbaren IP-Formaten möglich ist.

Der Importaufwand für einen SoC-Entwurf muss mit der Anzahl der zu integrierenden IP-Module und der Anzahl der notwendigen Aktualisierungen multipliziert werden. Dabei ist zu beachten, dass die Anzahl der verwendeten IP-Module stark wächst [29]. Es werden bereits SoC-Entwürfe begonnen, die 150 IP-Module enthalten werden [76]. Auch die wachsende

Anzahl der IP-Anbieter ist ein Indikator für den steigenden Bedarf an IP-Modulen [10]. Legt man die Methodik des plattformbasierten Entwurfs zugrunde, sind in der Entwurfsplattform bereits IP-Module fest integriert. Deren Importaufwand schlägt nur einfach (bei der Plattformentwicklung) pro Produktfamilie zu Buche. Jedoch müssen weitere Fremd-IP-Module, wie beispielsweise CPU, DSP, Audio-/Video Codecs, A/D-Konverter usw.) zusätzlich in den SoC-Entwurf integriert werden, um das Endprodukt von verwandten Produkten der eigenen Produktfamilie oder denen der Wettbewerber zu unterscheiden. Hinzu kommen erneute Imports aufgrund von Fehlerkorrekturen beziehungsweise Updates des IP-Moduls.

5.2.11 Zusammenfassung der Anforderungen

Für die Automatisierung von Qualifizierungs- und Auslieferungsprozessen wird ein IP-Format benötigt, das unabhängig ist und die Semantik der IP-Struktur dokumentieren kann. Somit kann vermieden werden, dass Werkzeuge mit einer API ausgestattet werden müssen, um auf die Daten eines IP-Moduls zugreifen zu können. Durch die Dokumentation des Entwurfsablaufs werden Ergebnisse von Werkzeugabläufen transparent und nachvollziehbar. Vollständigkeits- und Integritätsmessungen werden durch die Modellierung von Abhängigkeiten im IP-Format ermöglicht. Auf dieser Basis können automatische Qualifizierungs- und Auslieferungsprozesse aufsetzen, wodurch die Zeiten für die Qualifizierung und die IP-Auslieferung stark verkürzt werden können. Ein Format, das die oben genannten Anforderungen erfüllt, wird im nächsten Abschnitt beschrieben.

5.3 IPQ-Format

In diesem Abschnitt wird ein unabhängiges IP-Format als Grundlage für automatische Auslieferungen und automatische Qualifizierungen als Lösung für die im vorangegangenen Abschnitt genannten Anforderungen vorgeschlagen. Ein schematischer Überblick über das IP-Qualifizierungsformat (IPQ-Format) ist in Abbildung 9 dargestellt.

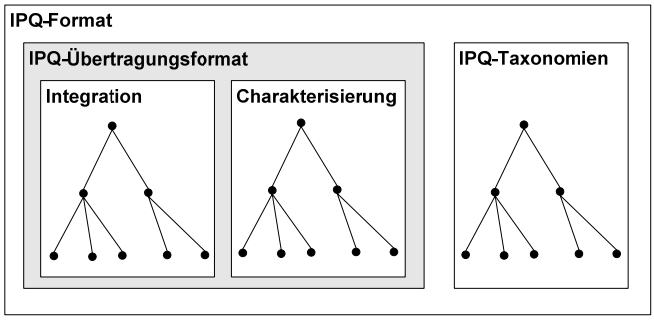


Abbildung 9: IPQ-Format

Abbildung 9 zeigt, dass das IPQ-Format ein IP-Modul als eine Kombination aus Charakterisierungs-, Integrationsdaten und Taxonomien darstellt [21]. Charakterisierungsdaten beschreiben ein IP-Modul durch Attribute. Benutzt wird die Charakterisierung für die computergestützte IP-Suche, zur Evaluierung und zum Vergleich von IP-Modulen. Für die IP-Suche wird zusätzlich noch eine IPQ-Taxonomie benötigt. Dabei kann es sich um eine

Standardtaxonomie oder eine firmenspezifische Taxonomie handeln, welche die Firmenpräferenzen bei der IP-Suche widerspiegelt. Beispielsweise beschränken viele IP-Nutzer die Anzahl ihrer IP-Anbieter, um den Lizenzierungsaufwand beim Kauf zu reduzieren. Die Vorauswahl der bevorzugten IP-Anbieter kann in einer firmenspezifischen Taxonomie eingestellt werden. Die Integrationsdaten eines Soft-IP-Moduls sind die vertraglich zugesicherten Dateien, die zur Integration, Verifikation und Synthese notwendig sind. Integrations- und Charakterisierungsdaten sind zum IPQ-Übertragungsformat zusammengefasst. IPQ-Taxonomien sind nicht eingeschlossen, da sie firmenspezifisches Wissen enthalten können und daher nicht ausgeliefert werden.

Das IPQ-Übertragungsformat erlaubt, alle notwendigen Informationen und Daten zwischen IP-Nutzer und IP-Anbieter zu übertragen. Die Menge der Daten ist durch den Anwendungkskontext beschränkt. Beispielsweise sind im IPQ-Übertragungsformat keine Integrationsdaten enthalten, wenn der IP-Nutzer zum ersten Mal Kontakt mit dem IP-Anbieter bei der IP-Suche aufnimmt. In diesem Fall werden nur relevante Charakterisierungsdaten übertragen. Erst bei einer IP-Auslieferung werden die vollständigen Integrations- und Charakterisierungsdaten eines IP-Moduls im IPQ-Übertragungsformat an den IP-Nutzer übermittelt.

Die in Abbildung 9 angedeutete Baumstruktur der Integrations-, Charakterisierungs- und Taxonomiedaten spiegelt die IP-Struktur beziehungsweise die beschreibende XML-Dokumentstruktur wieder. Tatsächlich handelt es sich aber um keine klassische Baumstruktur, da durch Kennnummern und Referenzen die Baumstruktur aufgeweicht ist. Dies ist notwendig, um Abhängigkeiten zwischen den Knoten modellieren zu können.

Wie bereits angekündigt, befasst sich die weitere Arbeit mit dem Integrationsdatenformat, auf dessen Bestandteile im nächsten Abschnitt eingegangen wird.

5.4 IPQ-Integrationsdatenformat

Das Integrationsdatenformat ermöglicht eine einheitliche Semantik der IP-Daten und eine Strukturierung des IP-Moduls für automatische Prozesse, ohne die IP-Daten verändern zu müssen. Im IPQ-Integrationsdatenformat wird ein IP-Modul in Bibliotheken, Zellen, physikalische und logische Einheiten, sowie logische Rollen und Teile gegliedert.

Abbildung 10 zeigt ein UML-Diagramm, das die Beziehungen zwischen den Bestandteilen darstellt. Durch die Modellierung des IP-Moduls mit dem IPQ-Format, wird das Anpassungsproblem beim IP-Import auf das korrekte Generieren von Metadaten während der IP-Exportphase und die Wiederherstellung der physikalischen Abhängigkeiten aus den logischen Abhängigkeiten während der IP-Importphase reduziert. Auf die einzelnen Bestandteile wird im Folgenden eingegangen.

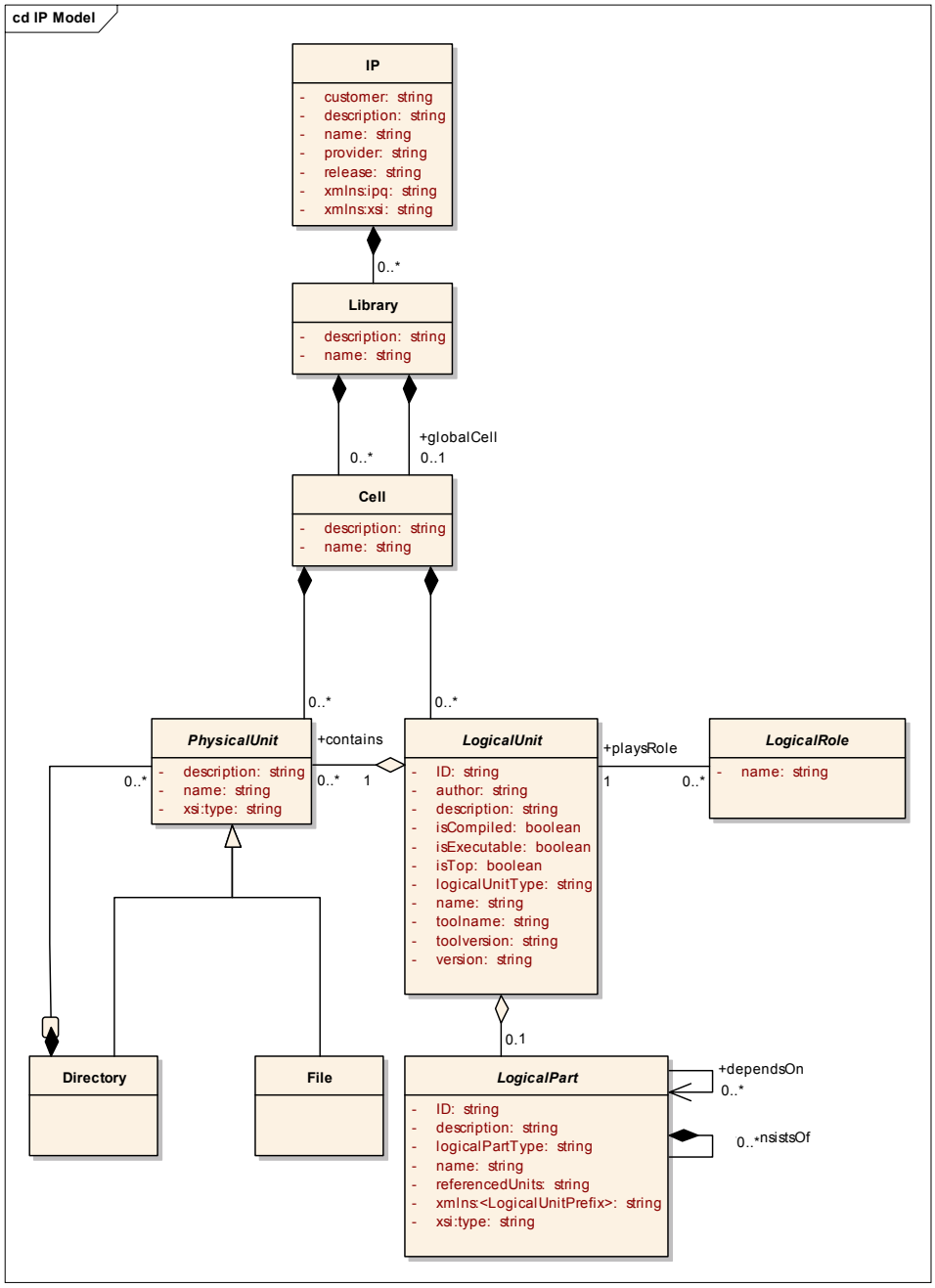


Abbildung 10: IPQ-Integrationsdatenformat – UML-Modell

5.4.1 Bibliotheken

Bibliotheken (engl. libraries) gliedern ein IP-Modul auf höchster Ebene. Während IP-Module geringer Komplexität in der Regel mit einer Bibliothek auskommen, werden komplexe IP-Module, die selbst aus IP-Modulen aufgebaut sind in mehrere Bibliotheken gegliedert. Dabei enthält eine Bibliothek die Daten eines IP-Moduls.

5.4.2 Zellen

Zellen (engl. cells) unterteilen Bibliotheken und ermöglichen eine feinere Strukturierung eines IP-Moduls. In einer Zelle können beispielsweise Funktionsblöcke eines IP-Moduls darstellen, wie Taktgenerierung, Ausgangssynchronisation, Befehlsschnittstelle usw. In einer besonderen Zelle, der so genannten „Global Cell“, werden Daten zusammengefasst, die von allen Funktionsblöcken geteilt werden. Zu solche Daten zählen z. B. HDL-Bibliotheken und Dokumentationsdaten.

5.4.3 Physikalische Einheiten

Physikalische Einheiten (engl. physical unit) entsprechen den Dateien und Verzeichnissen eines Funktionsblocks. Bei weniger komplexen IP-Modulen, wird die Strukturierung in der Regel auf dieser Ebene beginnen, da in diesem Fall eine Unterteilung in Zellen oder gar Bibliotheken den nachteiligen Effekt hat, das IP-Modul sogar unübersichtlicher darzustellen.

5.4.4 Logische Einheiten

Eine logische Einheit steht für den Inhalt einer Datei. Eine physikalische Einheit enthält genau eine logische Einheit (engl. logical unit). Logische Einheiten benennen das Dateiformat der übergeordneten physikalischen Einheit. Während Verzeichnissen die gleichnamige logische Einheit zugewiesen wird, werden Dateien, die relevanten Dateiformate zugewiesen, wie beispielsweise VHDL, Verilog, Tcl, C-Shell usw. Mit logischen Einheiten wird die Entkopplung zwischen anbieter- und nutzerspezifischer IP-Struktur erreicht. Abhängigkeiten, die zwischen logischen Einheiten bestehen, sind formatübergreifend formuliert und werden im Gegensatz zu den Abhängigkeiten auf physikalischer Ebene bei einer Umstrukturierung nicht ungültig.

5.4.5 Logische Rollen

Logische Rollen (engl. logical roles) geben logischen Einheiten und damit den übergeordneten physikalischen Einheiten eine Bedeutung. Dabei handelt es sich um Schlüsselwörter, wie Entwurf, Verifikation, Dokumentation usw., die der logischen Einheit zugewiesen werden. Aufgrund dieser Semantik kann ein Importwerkzeug die Bedeutung einer Datei erkennen und bei einer notwendigen Umstrukturierung die Zuordnung in die Verzeichnisstruktur unabhängig von der Quellstruktur durchführen. Für dieses Vorgehen muss ein einheitliches Verständnis der Integrationsdatensemantik vorausgesetzt werden. Dies wird durch die Definition von logischen Rollen im IPQ-Format erreicht. Ausschließlich definierte Rollen werden von den darauf aufsetzenden Werkzeugen berücksichtigt. Aus diesem Grund dürfen keine beliebigen Schlüsselwörter für logische Rollen verwendet werden. Einer logischen Einheit dürfen beliebig viele der definierten logischen Rollen zugewiesen werden, wobei keine Priorisierung durch die Reihenfolge der Aufzählung erfolgt und Mehrfachzuweisungen der gleichen Rolle ignoriert werden beziehungsweise bei werkzeuggestützter Generierung

des Formats Mehrfachzuweisungen automatisch zu einer Rolle zusammengefasst werden.

5.4.6 Logische Teile

Logische Teile (engl. logical parts) sind logischen Einheiten zugeordnet. Sie modellieren dateiformatübergreifende Abhängigkeiten und Verweise zwischen logischen Einheiten. Auf dieser Modellierung basieren die Integritäts- (Abschnitt 4.3.4) und Konsistenzprüfung eines IP-Moduls. Wie bereits bei den logischen Rollen beschrieben wurde, dürfen auch hier lediglich definierte logische Teile einer logischen Einheit zugewiesen werden. Es dürfen aber ebenfalls beliebig viele logische Teile zugewiesen werden. Für Mehrfachzuweisungen und die Zuweisungsreihenfolge gilt ebenfalls das Gleiche wie für logische Rollen.

Logische Teile enthalten detaillierte Informationen über eine logische Einheit. Eine logische Einheit steht für ein Dateiformat. Unterschiedliche logische Einheiten enthalten daher in der Regel unterschiedliche logische Teile. Im Falle des Dateiformats „VHDL“ stehen logische Teile für reservierte Worte der Sprache, die Abhängigkeiten zu anderen Dateien darstellen (z. B. `entity`, `architecture`, `configuration`, `file`, `foreign`). In Syntheseskripten werden unter anderem die `library`-Anweisung und Verweise auf HDL-Dateien durch logische Teile dargestellt.

5.4.7 Erweiterbarkeit

Neben der Erfüllung der Anforderungen an ein IP-Format muss auch die Erweiterbarkeit möglich sein. Ein einheitliches Format wird nicht in der Lage sein, alle Bedürfnisse abzudecken, daher muss es die Möglichkeit geben, das Format um zusätzliche Informationen zu erweitern, ohne den auf dem Format basierenden Entwurfsablauf zu beeinträchtigen. Insbesondere wird das langfristig, die Definition weiterer logischer Rollen und Teile betreffen. Daher ist das IPQ-Format um zusätzliche Informationen erweiterbar. Die auf dem IPQ-Format basierenden Werkzeuge sind so entworfen worden, dass sie Erweiterungen ignorieren und ihre gewohnte Funktionalität beibehalten, bis eine Interpretation der Erweiterungen implementiert wird. Dadurch sind firmenspezifische Erweiterungen möglich, ohne den unabhängigen Austauschcharakter des Formats zu gefährden. Jedoch sollte die individuelle Erweiterbarkeit nicht dazu führen, dass keine Verbesserungen in den Standard einfließen oder wiederum proprietäre Formate entstehen. Einer Beliebigkeit der Formaterweiterung kann nur durch eine agile Formatpflege entgegengetreten werden.

5.4.8 Formatpflege

Für die Pflege des IPQ-Formats wird ein Konsortium vorgeschlagen, das in regelmäßigen Abständen die Anforderungen der Nutzer prüft, über Erweiterungen des Standards berät und diese gegebenenfalls in den Standard einfließen lässt.

In Abbildung 6 auf Seite 37, ❺ wurde dargestellt, dass nach erfolgreichem Abschluss der Zertifizierungsphase das IP-Modul an einen Kunden ausgeliefert werden kann. Bis zu diesem Punkt sind Qualifizierungskriterien definiert und Qualifizierungsmethoden zu deren Überprüfung aufseiten des IP-Anbieters entwickelt worden. Im nächsten Schritt sollen qualifizierte IP-Module an einen IP-Nutzer übertragen werden. Dabei muss beachtet werden, dass bereits qualifizierte Merkmale eines IP-Moduls erhalten bleiben beziehungsweise automatisch wieder hergestellt werden können. Im Folgenden wird der IP-Auslieferungsprozess genauer untersucht. Hier sind die drei verbleibenden Qualifizierungsphasen installiert.

5.5 IP-Qualifizierungsphasen

Die Auslieferung eines IP-Moduls an den IP-Nutzer ist in drei Phasen unterteilt: IP-Export, IP-Übermittlung und IP-Import (Abbildung 8 auf Seite 49). Jede dieser Phasen wird von Qualifizierungsphasen unterstützt. Während die Export- und Importphase von den gleichnamigen Qualifizierungsphasen unterstützt werden, ist in der Übermittlungsphase eine Eingangskontrolle installiert. Die Qualifizierungsmethoden dieser Phasen werden in den folgenden Abschnitten beschrieben.

5.5.1 IP-Export

In der Exportphase, wählt der IP-Anbieter die vereinbarten Integrationsdaten für die Übermittlung an den IP-Nutzer aus. Möglicherweise wurde eine kundenspezifische Qualifizierung vereinbart, die dann als Teil der Exportphase durchgeführt wird (Abschnitt 4.2.2). Anschließend generiert der IP-Anbieter aus den Integrations- und Charakterisierungsdaten das IPQ-Format, das den automatischen Import aufseiten des IP-Nutzers ermöglicht. Eine qualitativ hochwertige Auslieferung wird in der Exportphase durch verschiedene, installierte Qualifizierungsmethoden gewährleistet, die im Folgenden näher erläutert werden.

5.5.1.1 Versionsprüfung

Vor der Auswahl der Integrationsdaten wird zunächst automatisch das korrekte Auslieferrelease der Entwurfsdaten anhand des Kundennamens und des IP-Modulnamens bestimmt. Aufgrund von Verbesserungen und Korrekturen am IP-Modul können mehrere Releases eines IP-Moduls in der Wiederverwendungsdatenbank enthalten sein. Für eine Auslieferung wird daher immer das aktuelle Release verwendet. Ein Release ist an einer speziellen Markierung der Daten zu erkennen, die besagt, dass die vorausgegangene, abschließende Qualifizierung erfolgreich durchgeführt wurde. Anhand des Datums, das der Markierung zugeordnet ist, wird das aktuelle Release bestimmt. Ist für den IP-Modulnamen eine kundenspezifische Markierung vorhanden, wird die aktuelle, kundenspezifisch qualifizierte Version verwendet.

5.5.1.2 Vollständigkeitsprüfung

Eine qualitativ hochwertige Auslieferung zeichnet sich auch durch ihre Vollständigkeit in Bezug auf den vereinbarten Lieferumfang aus. In der Exportphase wird sichergestellt, dass alle für die Integration in den Systementwurf benötigten Dateien in der Auslieferung enthalten sind. Die Integrationsdaten, die an einen IP-Nutzer ausgeliefert werden, hängen von einem vereinbarten Vertragsmodell ab. Mit einem Vertragsmodell ist ein definierter, vollständiger Auslieferumfang verknüpft, der dem Auslieferer automatisch als initialer Auslieferumfang vorgeschlagen wird. Zusätzlich besteht die Möglichkeit, die Auslieferung manuell anzupassen. Wird der Vorschlag übernommen, ist die Vollständigkeit des IP-Moduls gewährleistet. Wird von der manuellen Anpassungsmöglichkeit gebrauch gemacht, muss die Vollständigkeit durch den Auslieferer sichergestellt werden. Dabei wird er von der automatischen Integritätsprüfung, die im nächsten Abschnitt beschrieben wird, unterstützt. Tabelle 4 zeigt die Zuordnung von Vertragsmodellen zu dem jeweiligen Lieferumfang.

Vertragsmodelle	Lieferumfang
VHDL-Quellcode	VHDL-Quellcode VHDL-Testbenches VHDL-Bibliotheken Eingabevektoren für die Simulation Ausgabevektoren für die Simulation Syntheseskripte Dokumentation Lizenzinformationen Werkzeugkonfigurationen
Verilog-Quellcode	Verilog-Quellcode Verilog-Testbenches Verilog-Bibliotheken Eingabevektoren für die Simulation Ausgabevektoren für die Simulation Syntheseskripte Dokumentation Lizenzinformationen Werkzeugkonfigurationen

Vertragsmodelle	Lieferumfang
VHDL- und Verilog-Quellcode	VHDL-Quellcode VHDL-Testbenches VHDL-Bibliotheken Verilog-Quellcode Verilog-Testbenches Verilog-Bibliotheken Eingabevektoren für die Simulation Ausgabevektoren für die Simulation Syntheseskripte Dokumentation Lizenzinformationen Werkzeugkonfigurationen
(Gatter-)Netzliste	Technologieabhängige Gatternetzliste Dokumentation Lizenzinformationen Werkzeugkonfigurationen
FPGA Dateien	FPGA-Dateien Dokumentation Lizenzinformationen Werkzeugkonfigurationen
Benutzerdefinierte Auslieferung	Zu Beginn leer

Tabelle 4: Vertragsmodelle

5.5.1.3 Integritätsprüfung

Aufgrund der manuellen Anpassungsmöglichkeit des Auslieferumfangs nach der Auswahl eines Vertragsmodells, kann die während der abschließenden Qualifizierungsphase geprüfte Integrität des IP-Moduls nicht mehr gewährleistet werden. Die Integrität muss erneut qualifiziert werden. Sie wird nach der gleichen Methode wie in der abschließenden Qualifizierungsphase durchgeführt. Diese Methode wurde in Abschnitt 4.3.4 beschrieben.

5.5.1.4 Bewertung der IP-Struktursemantik

Um die Bedeutung der Dateien im IP-Modulkontext für den automatischen Import vorzubereiten, wird ihre Semantik in logischen Einheiten und Rollen erfasst. Diese werden anhand einer anbieterspezifischen Konfigurationsdatei den physikalischen Einheiten der Anbieter IP-Struktur zugewiesen. Für die Zuweisung sind in der Konfigurationsdatei Regeln definiert, die Pfadmustern bestimmte logische Rollen beziehungsweise Einheiten zuweisen. Das Vorgehen wird am Beispiel in Abbildung 12 auf Seite 64 erläutert.

5.5.1.5 Dokumentation, Überprüfbarkeit und Verschlüsselung der Übertragung

Trotz der automatischen Importmöglichkeit, soll dem IP-Nutzer die manuelle Navigation in der IP-Struktur erleichtert werden. Zusätzlich soll er die Möglichkeit haben, sich von der korrekten Übertragung der Daten zu überzeugen. Dafür wird in der Exportphase zum einen die IP-Struktur des IP-Anbieters für den vereinbarten Integrationsdatenumfang dokumentiert, indem eine hierarchische Darstellung der IP-Struktur generiert wird. Des Weiteren werden Prüfsummen für alle auszuliefernden Dateien generiert und das erzeugte Auslieferungsarchiv inklusive Prüfsummen, IP-Metadaten und Strukturdokumentation automatisch verschlüsselt, um das IP-Modul vor Missbrauch während der Übertragung zu schützen.

Eine besondere Form des IP-Exports ist der virtuelle IP-Export.

5.5.2 Virtueller IP-Export

Die Methode des virtuellen IP-Exports bietet die Möglichkeit, eine Umstrukturierung beziehungsweise eine kundenspezifische IP-Formaterzeugung aufseiten des IP-Anbieters durchzuführen. Eine virtuelle Auslieferung bedeutet, dass Export, Übermittlung und Import in das Zielformat aufseiten des IP-Anbieters durchgeführt werden. Dadurch kann der IP-Anbieter automatisch ein bekanntes Zielformat erzeugen und übermitteln. Dies ist nur möglich, wenn der IP-Anbieter über das Zielformat informiert ist. In der Regel wird eine Formaterzeugung auf Anbieterseite in Erwägung gezogen, um IP-Module gezielt für Kunden mit besonderen Formatansprüchen anzubieten. Durch das Angebot einer Umstrukturierung beziehungsweise Formatgenerierung kann zusätzlich auch das Portfolio der Entwurfsdienstleistung erweitert werden. Beispielsweise ist die Erzeugung des Platform Express Objektformats (Kapitel 3) eine mögliche Anwendung des virtuellen Exports. Die benötigten zusätzlichen Informationen können während des virtuellen Exports im IPQ-Format modelliert werden. In der virtuellen Importphase wird anhand dieser Informationen automatisch das Platform Express Objektformat erzeugt.

5.5.3 IP-Übermittlung und Eingangskontrolle

Das exportierte IP-Modul wird vom IP-Anbieter an den IP-Nutzer übermittelt (Abbildung 8, IP-Übermittlung). Während dieser Phase geht die Verantwortung für das IP-Modul vom IP-Anbieter auf den IP-Nutzer über (Abbildung 8, Verantwortungsübergabe). Das bedeutet, dass bis zu diesem Punkt der IP-Anbieter für Qualität und Funktion des IP-Moduls verantwortlich ist.

Die anschließende Importphase macht möglicherweise eine Umstrukturierung und Anpassungen des IP-Moduls notwendig, um das IP-Modul an den Entwicklungsablauf anzupassen. Qualitätsmängel oder Funktionseinschränkungen, die sich daraus ergeben, fallen nicht in den Verantwortungsbereich des IP-Anbieters. Daher müssen beide Parteien die Möglichkeit haben, sich von Qualität und Funktion an einem definierten Punkt im Verlauf der IP-Auslieferung und auf Basis eines einheitlichen Formats

überzeugen zu können. Ein einheitliches Format hat den Vorteil, dass werkzeuggestützte Qualitätsmessungen und Funktionstest ohne Anpassungen verwendet werden können. Der IP-Nutzer kann beispielsweise eine automatisierte Eingangskontrolle mit dem gleichen Qualifizierungsframework durchführen, wie sie beim IP-Anbieter für die entwurfsbegleitende beziehungsweise abschließende Qualifizierung verwendet wird. Dadurch kann der IP-Nutzer sich von der vertraglich vereinbarten Qualität überzeugen, bevor er das IP-Modul in seinen Entwurfsablauf importiert.

Auf die Messungen der Quellcode-, Verifikations- und Entwurfsqualität sowie Integritäts- und Konsistenzprüfung wurde bereits in den Abschnitten 4.3.1 bis 4.3.5 eingegangen. Ebenfalls wurde die Methode zur Vollständigkeitsprüfung in Abschnitt 5.5.1.2 beschrieben. Diese Methoden sind auch in der Eingangskontrolle enthalten, werden nun aber vom IP-Nutzer durchgeführt.

5.5.4 IP-Import

Im Anschluss an eine erfolgreiche Eingangskontrolle wird während der Importphase das IP-Modul automatisch an die Entwicklungsumgebung des IP-Nutzers angepasst. Handelt es sich um eine externe Wiederverwendung, wird sich die Entwicklungsumgebung des IP-Anbieters in der Regel von der des IP-Nutzers unterscheiden. Daher muss das IP-Modul in die veränderte IP-Struktur überführt werden. Dieser Vorgang wird Umstrukturierung genannt. Anschließend müssen während der Anpassungsphase Dateiinhalte an die neue Struktur angepasst werden. Die Automatisierung der Umstrukturierung und Anpassung an den veränderten Entwurfsablauf während des Imports wird durch das IPQ-Integrationsdatenformat ermöglicht. Der IP-Nutzer muss seine firmenspezifische Struktur dafür nicht ändern.

In Abbildung 11 ist die Problematik der Importphase dargestellt. Dieser Prozess kann wegen der Abhängigkeit von anbieter- und nutzerspezifischen Strukturen nicht profitabel automatisiert werden. Untersucht man die Ausführbarkeit von Dateien nach der Umstrukturierung (z. B. Syntheseskripte, wie *synopsys-setup.csh* in Abbildung 11), stellt man Abhängigkeiten von der IP-Struktur fest, welche die Ausführbarkeit nach einer Umstrukturierung verhindern, da die im Skript referenzierten Abhängigkeiten (Pfade) ungültig geworden sind. Dagegen werden PDF-Dokumente, von Werkzeugen generierte Berichte oder ein vollständiges Verzeichnis mit synthetisierbarem HDL-Quellcode durch eine Umstrukturierung nicht in ihrer Lesbarkeit oder weiteren Verarbeitung eingeschränkt.

Zusätzlich zum Verschieben der Dateien in ein neues Verzeichnis kann es notwendig sein, den Dateinamen anzupassen, um Konformität mit den Namenskonventionen des IP-Nutzers zu erzielen. Häufig spiegeln Dateinamen den Dateiinhalt wieder. Enthält eine Datei beispielsweise eine VHDL `Entity`, kann gefordert werden, dass der Dateiname *<entity_name>.vhd* lautet und der Dateiname der zugehörigen VHDL `Architecture` *<entity_name>_<architecture_name>.vhd*. Solche An-

passungen lassen sich nur durch eine Analyse des Quellcodes durchführen.

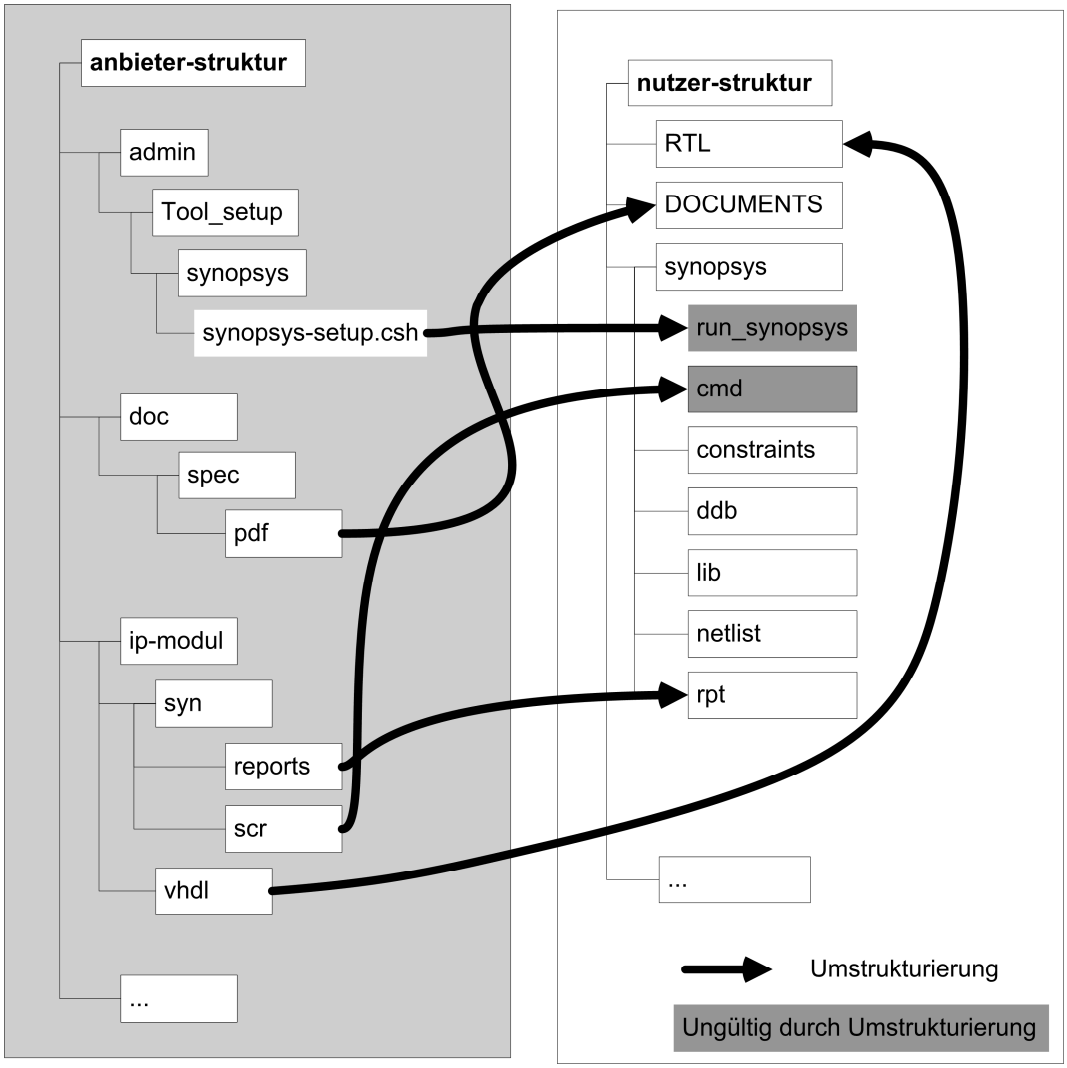


Abbildung 11: IP-Import verursacht ungültige Dateiabhängigkeiten.

Mit dem IPQ-Integrationsdatenformat kann dieser bisher manuell durchgeführte, zeitaufwendige und fehlerträchtige Prozess automatisiert werden (Abbildung 12). Die physikalischen Abhängigkeiten werden durch logische Abhängigkeiten ersetzt, die auch nach einer Umstrukturierung ihre Gültigkeit behalten (Abschnitt 5.4). In Abbildung 12 wird beispielsweise das Syntheseskript *synopsys-setup.csh* zunächst in das unabhängige IPQ-Integrationsdatenformat exportiert. Die physikalische Struktur ändert sich dabei nicht. Zusätzlich werden automatisch IPQ-Metadaten erzeugt. Über eine Konfigurationsdatei des IP-Anbieters werden der physikalischen Einheit eine logische Einheit und logische Rollen zugewiesen. Diese werden in den IPQ-Metadaten gespeichert und mit der physikalischen Struktur an den IP-Nutzer gesendet. In der Importphase kann die Datei *synopsys-setup.csh* aufgrund der logischen Einheit und logischen Rollen korrekt in die Zielstruktur eingeordnet werden.

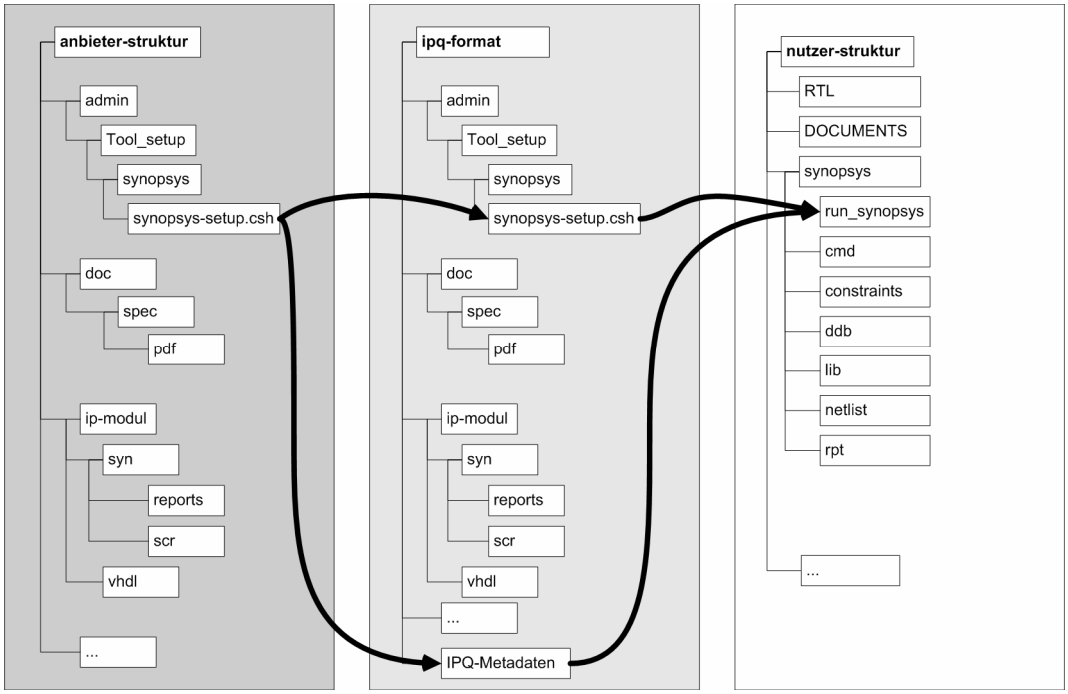


Abbildung 12: Unabhängiges Zwischenformat – IP-Integrationsdaten

Für die automatische Umstrukturierung und Anpassung der Integrationsdaten müssen sowohl Anbieter als auch Nutzer zumindest intern eine definierte IP-Struktur einhalten. Diese Anforderung wird von den meisten IP-Anbietern und IP-Nutzern erfüllt, da die automatisierten Entwurfsabläufe ohnehin auf einer definierten Struktur aufsetzen müssen. Durch diese Forderung wird die Automatisierung für beliebige IP-Module ermöglicht, die diese Struktur einhalten. Für die automatische Umstrukturierung und Anpassung eines IP-Moduls wird die Semantik der Dateien, also welche Rolle eine Datei im IP-Modulkontext erfüllt, aus den IPQ-Metadaten ausgewertet.

5.5.4.1 Semantik der IP-Struktur

Anhand des Beispiels in Abbildung 12 wird im Folgenden erklärt, wie die Semantik der physikalischen Einheit *synopsys-setup.csh* erkannt und in den IPQ-Metadaten festgeschrieben wird. Der Datei

/anbieter-struktur/admin/Tool_setup/synopsys/synopsys-setup.csh

werden anhand definierter Regeln in der anbieterspezifischen Konfigurationsdatei die Rollen „Synthese“, „Setup“ und „Skript“ während dem IP-Export zugewiesen. Anhand der anbieterspezifischen Dateinamenserweiterung wird das Dateiformat C-Shell-Skript erkannt. Dokumentiert in den IPQ-Metadaten werden diese Informationen an den IP-Nutzer übermittelt. Herkömmliche Werkzeuge im Entwurfsablauf können aus der reinen Pfadangabe des IP-Anbieters nicht erkennen, dass es sich, um ein Skript zur Erzeugung der für das IP-Modul spezifischen Syntheseumgebung handelt. Nur anhand der firmenintern verwendeten IP-Struktur ist die Bedeutung dieser Datei klar. Durch das IPQ-Integrationsdatenformat wird diese Information an den IP-Nutzer weitergegeben, um dort durch Um-

strukturierung die nutzerspezifische Bedeutung aus den IPQ-Metadaten abzuleiten und dem Entwurfsablauf verständlich zu machen. Durch die logischen Rollen in Kombination mit einer nutzerspezifischen Konfigurationsdatei wird der Zielpfad in der IP-Struktur bestimmt. Im Beispiel ist das */nutzer-struktur/synopsys/run_synopsys*. Im nächsten Abschnitt wird das Vorgehen bei der Umstrukturierung beschrieben.

5.5.4.2 Umstrukturierung

Während der Strukturanpassung wird das Zielverzeichnis in der IP-Nutzer Struktur für Dateien aus der IP-Anbieter Struktur bestimmt. Die Bestimmung des Zielverzeichnisses wird anhand der logischen Rollen und einer Konfigurationsdatei beim IP-Nutzer vorgenommen. In der Konfigurationsdatei ordnen Regeln logischen Rollen Dateipfade zu. Als weiteres differenzierendes Element wird das Dateiformat, gegeben durch die logische Einheit, in die Regeln mit einbezogen. Aufgrund der Tatsache, dass mehrere Regeln für die Verschiebung einer Datei in die Zielstruktur in Frage kommen können, muss eine Bewertung der Regeln durchgeführt werden. Andernfalls nimmt nicht die am besten passende Regel die Umstrukturierung vor, sondern es würde zu einer Priorisierung der Regeln anhand ihrer Reihenfolge in der Konfigurationsdatei kommen.

Regelbewertung

Es gibt Regeln R_i mit $i \in \mathrm{N}$ und Dateien D_j mit $j \in \mathrm{N}$. Für eine Bewertung wird eine Datei D_j, genau genommen ihre physikalische Einheit, gegen alle Regeln R_i der Konfigurationsdatei geprüft. Dabei kann es zu folgenden Ergebnissen kommen:

1. D_j wird von keiner R_i abgedeckt.
2. D_j wird von genau einer R_i abgedeckt.
3. D_j wird von mehr als einer R_i abgedeckt.

Für die einzelnen Fälle werden unterschiedliche Aktionen ausgelöst:

1. Es wird keine Regel auf die Datei angewendet und eine Fehlermeldung ausgegeben. Die Fehlermeldung weist darauf hin, dass die Umstrukturierung der Datei nicht von einer Regel abgedeckt ist und die Datei daher nicht automatisch in das Zielverzeichnis eingeordnet werden kann.
2. In diesem Fall wird die Regel auf die Datei angewendet und eine Information ausgegeben.
3. Es kommt mehr als eine Regel in Frage, um die Datei zu verschieben. Daher wird eine Bewertung vorgenommen. Die Regel mit der höchsten Bewertung bestimmt das Zielverzeichnis, falls $g_{i\,\mathrm{j}} > 0$ gilt. Zusätzlich wird eine Warnung generiert, dass die Umstrukturierung aufgrund einer Regelbewertung erfolgt ist, diese aber nicht unbedingt eindeutig ist.

Eine Regel R_i besteht aus mehreren Kriterien: Einem logischen Einheitskriterium, die Datei muss im Dateiformat mit der Regel übereinstimmen, und mindestens einem logischen Rollenkriterium. Die Bewertung bezüglich Einheitskriterium und Rollenkriterien erfolgt unabhängig voneinander. Das Einheitskriterium ist das ausschlaggebende Kriterium. Ist für eine logische Einheit, also das Dateiformat, keine Regel in der firmenspezifischen Konfiguration vorgesehen, kann davon ausgegangen werden, dass auch keine Prozesse zum Verarbeiten dieses Format (z. B. ein werkzeugspezifisches Syntheseskript) zur Verfügung steht. Für die Weiterverarbeitung müssen also zusätzlich zur Umstrukturierung die entsprechenden Prozesse bereitgestellt werden. In diesem Fall wird eine entsprechende Fehlermeldung ausgegeben und die weitere Bewertung gestoppt.

Vorausgesetzt das Einheitskriterium ist erfüllt wird für die Bewertung im dritten Fall der Übereinstimmungsgrad $g_{ij} \in \mathrm{N}$ einer Datei D_j mit der Regel R_i in die so genannte Bewertungsmatrix (Abbildung 13) eingetragen. Der Übereinstimmungsgrad gibt die Anzahl der Rollenübereinstimmungen einer Datei D_j mit der Regel R_i an. Zu Beginn ist $g_{i\mathrm{j}} = 0$. Für jede Übereinstimmung eines Rollenkriteriums wird $g_{i\mathrm{j}}$ um eins erhöht.

	R_1	R_2	...	R_i	...	R_n
D_1	0	2	...	g_{i1}	...	g_{n1}
D_2	1	1	...	g_{i2}	...	g_{n2}
...	...	...	...	...	...	...
D_j	$g_{1\mathrm{j}}$	$g_{2\mathrm{j}}$	...	$g_{i\mathrm{j}}$	...	g_{nj}
...	...	...	...	...	...	...
D_n	g_{1n}	g_{2n}	...	g_{in}	...	g_{nn}

Abbildung 13: Bewertungsmatrix

Eine Regel passt zu einer Datei, wenn in der Bewertungsmatrix $g_{i\mathrm{j}} > 0$ gilt, also Regel und Datei in mindestens einem Rollenkriterium übereinstimmen. Je mehr Kriterien übereinstimmen, desto besser ist die Bewertung. Im Beispiel in Abbildung 13 bestimmt Regel R_2 das Zielverzeichnis für Datei D_1. Im Fall von D_2 ist wegen der Gleichbewertung der Regeln nicht eindeutig, ob Regel R_1 oder Regel R_2 angewendet werden soll. Die Methode gibt in diesem Fall Regel R_1 aufgrund des kleineren Index den Vorzug. Zusätzlich wird eine Warnmeldung generiert.

Abhängigkeiten

Nach einer erfolgreichen Umstrukturierung müssen ungültig gewordene Dateiinhalte angepasst werden. Dafür ist die Dokumentation der ursprünglichen Abhängigkeiten in den IPQ-Metadaten erforderlich. Im IPQ-Integra-

tionsdatenformat werden die physikalischen Abhängigkeiten in der IP-Struktur durch logische Abhängigkeiten ersetzt. Während die physikalischen Abhängigkeiten bei der Umstrukturierung ungültig werden, bleiben die logischen Abhängigkeiten gültig. Physikalische Abhängigkeiten existieren zwischen Dateien und Verzeichnissen beziehungsweise zwischen Dateien und Dateien. Logische Abhängigkeiten, dargestellt zwischen logischen Einheiten des IPQ-Integrationsdatenformats, abstrahieren von den physikalischen Abhängigkeiten und dokumentieren dadurch auch nach einer Umstrukturierung die ursprüngliche physikalische IP-Struktur. Aufgrund dieser Tatsache können Dateiinhalte an die tatsächliche Struktur nach dem Import angepasst werden.

Anpassungen

Es gibt zwei Möglichkeiten, ungültig gewordene Abhängigkeiten in einem IP-Modul anzupassen. Beide Möglichkeiten haben Vor- und Nachteile, die im Folgenden diskutiert werden.

1. Bereits bei der Entwicklung können Umgebungsvariablen, Verweise auf physikalische Einheiten (Pfadnamen) in Skripten und Konfigurationsdateien ersetzen. Diese Variablen können von Konfigurationsskripten gesetzt werden. Diese Möglichkeit erfordert vom IP-Anbieter, dass zusätzliche Codierrichtlinien beim Erstellen der Dateien beachtet werden müssen. Obwohl diese Möglichkeit die geringste Werkzeugunterstützung benötigt, ist sie aber die am wenigsten automatisierbare Lösung. Besonders für Dateien, die nicht interpretiert werden, in deren Inhalt also keine Variablen ausgewertet werden, müssen zusätzlich Parser bereitgestellt werden, die die Ersetzungen durchführen.
2. Unabhängig von der ursprünglichen Entwicklung können mit Parsern für die relevanten Dateiformate (logische Einheiten) Abhängigkeiten (z. B. Pfadnamen) gefunden werden und über so genannte Aktionen (engl. actions) die notwendigen Informationen aus den IPQ-Metadaten für ihre Ersetzung ausgewertet werden. Diese Möglichkeit erfordert keine zusätzlichen Codierrichtlinien, erfordert aber die Entwicklung von Parsern für alle relevanten Dateiformate. Parser haben den Vorteil, dass sie für IP-Anbieter und IP-Nutzer transparent arbeiten. Nach einer vom Autor durchgeführten Analyse einiger IP-Module wird eine gute Abdeckung erreicht, wenn Parser für die folgenden Dateiformate bereitgestellt werden: C-Shell-, Perl- und Tcl-Skripte, Syntheseskripte, VHDL, Verilog, C und C++.

Aufgrund der Tatsache, dass nur ein kleiner Teil des jeweiligen Sprachumfangs durch die Parser unterstützt werden muss, um das IPQ-Format zu modellieren, handelt es sich bei dem zweiten Lösungsansatz, um eine praktikable Möglichkeit. Angesichts der Transparenz für IP-Anbieter und IP-Nutzer, der geringeren zusätzlich zu beachtenden Richtlinien und der Anwendbarkeit auf nicht interpretierte Dateiinhalte, wird von den Anpassungsmethoden der zweite Lösungsansatz verfolgt. Durch eine Integritätsprüfung (Abschnitt 4.3.4) werden ungültige Abhängigkeiten entdeckt und

mithilfe der Informationen aus den IPQ-Metadaten und den Umstrukturierungsinformationen in der nutzerspezifischen Konfigurationsdatei angepasst.

Die Pfadnamen der IP-Struktur beim IP-Anbieter vor dem Import sind im IPQ-Format dokumentiert und können über die logischen Einheiten von ihren übergeordneten physikalischen Einheiten abgerufen werden. Den logischen Einheiten werden zusätzlich während der Umstrukturierung die Pfadnamen in der Zielstruktur des IP-Nutzers zugewiesen. Mit diesen Informationen wird die Ersetzung „ungültiger“ durch „gültige“ Abhängigkeiten durchgeführt.

Bisher wurden lediglich Anpassungen nach einer Umstrukturierung betrachtet. Die grundlegende Anpassungsmethode kann aber auch in anderen Qualifizierungsphasen angewendet werden. Je nach Qualifizierungsphase unterscheiden sich die notwendigen Anpassungen. Beispielsweise ist es während der entwurfsbegleitenden Qualifizierungsphase notwendig, Objektnamen im Syntheseskript an Änderungen des HDL-Quellcodes anzupassen. Durch entsprechende, automatische Dokumentation der Quellcodeänderungen in den IPQ-Metadaten kann eine Anpassungsmethode automatisch die abhängige Referenz im Syntheseskript anpassen. Quellcodeänderungen werden durch Meldungen der Quellcode-Versionsverwaltung bei Änderung bestimmter Konstrukte in den IPQ-Metadaten dokumentiert.

Ebenfalls wird die notwendige Anpassung an den Werkzeugablauf adressiert. Die Werkzeuge (z. B. Simulator, Synthesewerkzeug, Qualifizierungswerkzeuge), die zur Verarbeitung der IP-Daten notwendig sind, werden nicht an den IP-Nutzer übermittelt. Implizit bestehen aber Abhängigkeiten zwischen den mitgelieferten Qualitätsmessmethoden, Synthese- und Simulationsskripten usw. und diesen Werkzeugen oder genauer gesagt, der benötigten Umgebung, in der das Werkzeug ausgeführt werden kann. Diese Umgebung wird durch ein firmenspezifisches Installationsskript hergestellt. Installationsskripte werden nicht an den IP-Nutzer übermittelt. Aufgrund der impliziten Abhängigkeit der modulspezifischen Startskripte von den firmenspezifischen Installationsskripten erzeugt eine Analysemethode anhand von Werkzeuginformationen automatisch die notwendige Umgebung oder beschleunigt die Installation im Dialog mit dem Anwender. Die benötigten Werkzeuginformationen sind den logischen Einheiten zugeordnet (Abschnitt 6.1.5). In einer firmenspezifischen Konfigurationsdatei werden Werkzeugnamen und -versionen mit den Dateipfaden zu den intern verwendeten Installationsdateien der Werkzeuge verknüpft. Durch einen Abgleich der beiden Datenquellen, IPQ-Metadaten und Konfigurationsdatei für den Werkzeugablauf, werden zu Beginn jedes Startskripts die entsprechenden Installationsdateien aufgerufen.

Am Ende der Importphase, nach erfolgreicher Umstrukturierung und Anpassung werden die Integrations- und Charakterisierungsdaten des IP-Moduls in die interne Wiederverwendungsdatenbank aufgenommen und durch eine entsprechende Markierung für die interne Wiederverwendung freigegeben. Hat der IP-Anbieter ein parametrisierbares IP-Modul geliefert,

können projektspezifische Parameter gesetzt werden (Abbildung 8, IP-Parametrisierung und Anpassung). Parametrisierbar bedeutet, dass die Parameter noch nicht oder nicht vollständig vom IP-Anbieter in der kundenspezifischen Qualifizierungsphase festgelegt wurden. Ein parametrisierbares IP-Modul ist dann notwendig, wenn das IP-Modul in mehreren Projekten wieder verwendet werden soll. Normalerweise ist der IP-Anbieter nur vage über das Integrationsprojekt informiert, was ebenfalls eine Parametrisierung und Anpassung aufseiten des IP-Nutzers notwendig macht. Nach erfolgreicher Anpassung, Parametrisierung und Import des IP-Moduls, steht es in der Wiederverwendungsdatenbank zur Integration für SoC-Projekt zur Verfügung (Abbildung 8 auf Seite 49, SoC-Entwurf).

6 Implementierung

Für die Demonstration der Funktions- und Leistungsfähigkeit der automatischen Qualifizierungs- und Auslieferungsmethodik hat der Autor QIP-Qualitätsmessungen implementiert und in den bestehenden Entwurfsablauf eines IP-Anbieters integriert (Abbildung 6 auf Seite 37, ❶-❻) [16].

Im Folgenden wird die Implementierung des IP-Integrationsdatenformats, der Wiederverwendungsdatenbank und des IP-Qualifizierungsframeworks (IPQ-Framework) für die Qualifizierung und Auslieferung von digitalen Soft-IP-Modulen beschrieben. Mithilfe des IPQ-Frameworks können Qualitätsmessungen, Qualitätsbewertungen, qualitätserhaltende Maßnahmen, Umstrukturierungen und Anpassungen an IP-Modulen durchgeführt werden.

In Abschnitt 6.1 wird die XML-Implementierung des IPQ-Integrationsdatenformats erläutert. Daran schließt sich die Integrationssbeschreibung der Wiederverwendungsdatenbank in Abschnitt 6.2 an. Die Implementierung des IPQ-Frameworks selbst ist Bestandteil von Abschnitt 6.3. In das Framework werden so genannte Experten für die unterschiedlichen Aufgaben integriert. Es werden die Experten für die Qualifizierung, die Integration externer Qualifizierungsexperten sowie Experten für die Zertifizierung, den Export und den Import von IP-Modulen beschrieben.

6.1 IPQ-Integrationsdatenformat

Die Implementierung des IPQ-Integrationsdatenformats besteht physikalisch aus einem komprimierten Archiv der Integrationsdaten und einem XML-Dokument, in dem automatisch generierte Metadaten zu einem IP-Modul gespeichert werden. Aufgrund dieser Metadaten werden die in Abschnitt 5.2 genannten Anforderungen erfüllt. Unter anderem ist dadurch ein flexibler, vom Format des IP-Anbieters beziehungsweise Nutzers unabhängiger und automatischer Import in den Entwurfsablauf möglich. Zusätzlich werden während der Qualifizierungsphasen Informationen im XML-Dokument gespeichert, die von Berichtsgeneratoren während der Zertifizierungsphase ausgewertet werden.

Für die Speicherung der Metadaten werden XML-Elemente und deren Attribute verwendet. Welche Elemente und Attribute zur Verfügung stehen, wird im Folgenden beschrieben. Quellcode 3 auf Seite 72 gibt einen Überblick über das XML-Datenformat. Der Überblick beschränkt sich auf die Darstellung der hierarchischen XML-Struktur der Elemente ohne ihre Attribute und ohne mögliche wiederholte Verschachtelung der Elemente. Diese gehen zum einen aus Abbildung 10 auf Seite 55 und zum anderen aus der detaillierten Beschreibung in den folgenden Abschnitten 6.1.1-6.1.7 hervor. Für die Modellierung der IPQ-Metadaten wird der eigene Namensraum `ipq` verwendet, der jedem Element als Präfix vorangestellt ist (`ipq:<Elementname>`). `<Elementname>` steht in diesem Fall als Platzhalter für einen XML-Elementnamen und nicht für einen XML-Tag.

```
1   <ipq:IP>
2     <ipq:Library>
3       <ipq:Cell>
4         <ipq:PhysicalUnit>
5           <ipq:LogicalUnit>
6             <ipq:LogicalRole/>
7             <ipq:LogicalPart/>
8           </ipq:LogicalUnit>
9         </ipq:PhysicalUnit>
10      </ipq:Cell>
11    </ipq:Library>
12  </ipq:IP>
```

Quellcode 3: Hierarchischer Aufbau des IPQ- Integrationsdatenformats

Ausgehend vom Wurzelknoten „IP“ konkretisiert die XML-Struktur ein IP-Modul durch Bibliotheken, Zellen, physikalische und logische Einheiten mit logischen Teilen und logischen Rollen. Diese Struktur ist in Quellcode 3 zu sehen.

6.1.1 IP-Modul

Das Wurzelelement der XML-Metadaten ist der `<ipq:IP>-Tag`. Die erlaubten Attribute für dieses Element sind in Tabelle 5 dargestellt.

Attribut	Datentyp	Verwendung
customer	String	Name des IP-Nutzers
description	String	Optionale Beschreibung des IP-Moduls
name	String	Name des IP-Moduls
provider	String	Name des IP-Anbieters
release	String	Auslieferrelease
xmlns:ipq	String	Deklaration des IPQ-Namensraums Wert: http://www.fzi.de/IPQ
xmlns:xsi	String	Deklaration des XML-Schemanamensraums Wert: http://www.w3.org/2001/XMLSchema-instance

Tabelle 5: Attribute des <ipq:IP>-Elements

6.1.2 Bibliotheken

Die nächste Hierarchieebene unterhalb von `<ipq:IP>` wird durch das XML-Element `<ipq:Library>` modelliert. Die erlaubten Attribute für dieses Element sind in Tabelle 6 dargestellt.

Attribut	Datentyp	Verwendung
description	String	Optionale Beschreibung der IP-Bibliothek
name	String	Name der IP-Bibliothek

Tabelle 6: Attribute des <ipq:Library>-Elements

6.1.3 Zellen

Die Hierarchieebene unterhalb von `<ipq:Library>` wird durch das XML-Element `<ipq:Cell>` modelliert. Die erlaubten Attribute für dieses Element sind in Tabelle 7 auf Seite 73 dargestellt.

Attribut	Datentyp	Verwendung
description	String	Optionale Beschreibung der IP-Zelle
name	String	Name der IP-Zelle (z. B. „Global“)

Tabelle 7: Attribute des <ipq:Cell>-Elements

6.1.4 Physikalische Einheiten

Die Hierarchieebene unterhalb von `<ipq:Cell>` wird durch das XML-Element `<ipq:PhysicalUnit>` modelliert. Die erlaubten Attribute für dieses Element sind in Tabelle 8 dargestellt.

Attribut	Datentyp	Verwendung
description	String	Optionale Beschreibung der physikalischen Einheit
name	String	Datei- beziehungsweise Verzeichnisname; der gesamte Pfadname ergibt sich aus der Zusammensetzung der Namen der übergeordneten physikalischen Einheiten vom Typ: `ipq:DirectoryType` (siehe xsi:type).
xsi:type	String	Typ der physikalischen Einheit; zulässig ist einer von zwei möglichen Werten `ipq:DirectoryType` für Verzeichnisse oder `ipq:FileType` für Dateien.

Tabelle 8: Attribute des <ipq:PhysicalUnit>-Elements

6.1.5 Logische Einheiten

Die Hierarchieebene unterhalb von `<ipq:PhysicalUnit>` wird durch das XML-Element `<ipq:LogicalUnit>` modelliert. Die erlaubten Attribute für dieses Element sind in Tabelle 9 auf Seite 74 dargestellt.

Attribut	Datentyp	Verwendung
ID	String	Eindeutige ID der logischen Einheit im IP-Kontext
author	String	Name des Autors der logischen Einheit (z. B. Name des VHDL-Entwicklers); wird aus dem CVS-Schlüsselwort `$Author:$` aus dem Dateiinhalt ausgelesen.
description	String	Optionale Beschreibung der logischen Einheit; als Standardwert wird die Zeile nach dem CVS-Schlüsselwort `$Purpose:$` eingetragen.
isCompiled	Boolean	Indikator dafür, dass die logische Einheit im Quellcode (isCompiled=false) oder in kompilierter Form enthalten ist (isCompiled=true)
isExecutable	Boolean	Indikator dafür, dass die logische Einheit ausführbar (isExecutable=true) (z. B. C-Shell-Skripte zum Start der Synthese oder Simulation) oder nicht ausführbar ist (isCompiled=false) (z. B. VHDL-Quellcode)
isTop	Boolean	Indikator dafür, dass die logische Einheit die höchste Ebene einer hierarchischen Struktur repräsentiert (isTop=true) (z. B. Toplevel VHDL Entity) oder eine untergeordnete Hierarchieebene repräsentiert (isTop=false). Diese Information wird beispielsweise für die Bearbeitung des IP-Moduls durch Simulations- und Synthesewerkzeuge benötigt.
logicalUnitType	String	Type der logischen Einheit (z. B. „VHDL“, „DCSH“)

Attribut	Datentyp	Verwendung
name	String	Name der IP-Zelle (z. B. „Global“)
toolname	String	Name des erzeugenden oder bearbeitenden Werkzeugs (z. B. „XEmacs“, „Synopsys Design-Compiler“, „Modeltech Modelsim“)
toolversion	String	Version des erzeugenden oder bearbeitenden Werkzeugs (z. B. „21.4.13“)
version	String	Version der logischen Einheit; Sie dokumentiert den Entwicklungsstand und wird aus den CVS-Header-Informationen beziehungsweise dem Dateibearbeitungsdatum extrahiert.

Tabelle 9: Attribute des <ipq:LogicalUnit>-Elements

6.1.6 Logische Rollen

Logische Rollen beschreiben die Aufgabe einer logischen Einheit im IP-Kontext. Eine der beiden Hierarchieebenen unterhalb von `<ipq:LogicalUnit>` wird durch das XML-Element `<ipq:LogicalRole>` modelliert. Die erlaubten Attribute für dieses Element sind in Tabelle 10 dargestellt.

Attribut	Datentyp	Verwendung
name	String	Name der logischen Rolle (z. B. „Documentation“, „Synthesis“, „Design“, „Simulation“ usw.). Eine logische Einheit kann beliebig viele `LogicalRole`-Elemente enthalten. Es sollten möglichst viele passende Rollen zugewiesen werden, um die Bedeutung einer Einheit so exakt wie möglich zu beschreiben.

Tabelle 10: Attribute des <ipq:LogicalRole>-Elements

6.1.7 Logische Teile

Logische Teile modellieren relevante Konstrukte eines Datenformats zur Darstellung von Abhängigkeiten oder Qualitätskriterien. Auf gleicher Hierarchieebene wie `<ipq:LogicalRole>` ist das XML-Element `<ipq:LogicalPart>` zur Konkretisierung von `<ipq:LogicalUnit>` angesiedelt. Die erlaubten Attribute für dieses Element sind in Tabelle 11 dargestellt.

Attribut	Datentyp	Verwendung
ID	String	Eindeutige ID des logischen Teils im IP-Kontext
description	String	Optionale Beschreibung des logischen Teils
logicalPartType	String	Typ des logischen Teils (z. B. „VHDLArchitecture“); eine Liste der modellierten Typen ist in Tabelle 18 im Anhang A.1 auf Seite 121 abgedruckt.
name	String	Name des logischen Teils (z. B. „Architecture“)

Attribut	Datentyp	Verwendung
referencedUnits	String	Eine durch Leerzeichen separierte Liste der LogicalUnit-IDs, auf die der logische Teil verweist; von denen die enthaltende logische Einheit also abhängig ist (z. B. VHDLArchitecture → VHDL Entity)
xmlns:<LogicalUnitPrefix>	String	Jede logische Einheit hat ihren eigenen XML-Namensraum (xmlns), um Konflikte zwischen gleichen Teilen unterschiedlicher Einheiten zu vermeiden. <LogicalUnitPrefix> steht für eine Abkürzung der logischen Einheit (z. B. vhdl). Eine Auswahl von Präfixen sind in Tabelle 12 auf Seite 75 aufgelistet.
xsi:type	String	Typ des logischen Teils (z. B. „vhdl:VHDLArchitectureType“); ordnet im Unterschied zu `logicalPartType` noch den Namensraum `vhdl` zu. Eine Liste der modellierten Typen ist in Tabelle 18 im Anhang A.1 auf Seite 121 abgedruckt.

Tabelle 11: Attribute des <ipq:LogicalPart>-Elements

In Tabelle 12 auf Seite 75 ist eine Auswahl von Präfixen für das XML-Attribut xmlns:<LogicalUnitPrefix> dargestellt. Die mit den Kürzeln verbundenen Datenformate müssen während einer Quellcodeanalysephase näher untersucht werden.

LogicalUnitPrefix	Erklärung
c	C-Format
cpp	C++-Format
csh	C-Shell-Format
dcrpt	Synopsys Design-Compiler Berichtsformat
dcsh	Synopsys Design-Compiler Shell-Format
dctcl	Synopsys Design-Compiler Tcl-Format
h	C-Header-Format
hpp	C++-Header-Format
perl	Perl-Format
spyglass	Atrenta Spyglass Berichtsformat
systemc	SystemC-Format
tcl	Allgemeines Tcl-Format
tcl-s	Das Tcl-s-Format wird in werkzeugspezifischen Konfigurationsdateien *.synopsys_dc.setup* verwendet. Es handelt sich um eine Teilmenge, des im tcl-shell-Modus unterstützten Tcl-Sprachumfangs.
verilog	Verilog-Format
vhdl	VHDL-Format
vncover	TransEDA VN-Cover Berichtsformat

Tabelle 12: Präfixe für logische Einheiten

Müssen weitere Datenformate für die Qualifizierung berücksichtigt werden, kann das Qualifizierungsframework um entsprechende Analysatoren

erweitert werden. Dadurch lassen sich dann weitere logische Einheiten mit ihren logischen Teilen in XML abbilden. Wie das Framework erweitert werden kann, ist in Abschnitt 6.3.2 beschrieben.

Tabelle 18 im Anhang A.1 auf Seite 121 gibt eine Übersicht über die Typen von logischen Teilen, die das Qualifizierungsframework aus einem VHDL-Quellcode extrahiert. In der zweiten Spalte ist der Grund der Modellierung erklärt. Alle Typen haben gemeinsam, dass sie Abhängigkeiten zwischen den einzelnen VHDL-Entwurfseinheiten modellieren, die in der Regel auch auf unterschiedliche physikalische Einheiten, also Dateien, aufgeteilt sind.

Logische Teile sind beispielsweise in VHDL Konstrukte, die auf logische Einheiten verweisen, also Abhängigkeiten darstellen. Die `Architecture` ist von der `Entity` abhängig, die `Configuration` von der `Architecture` und der `Entity` usw. Durch die Identifizierung dieser Verweise werden zum einen Konsistenz- und Integritätsprüfungen (z. B. des VHDL-Entwurfs) ermöglicht und zum anderen werden die Abhängigkeiten zwischen den logischen Einheiten, also z. B. zwischen den einzelnen VHDL-Entwurfseinheiten (dateiformatintern), und dateiformatübergreifend z. B. zwischen VHDL-Entwurfseinheiten und Syntheseskript erkennbar. Abhängigkeitsinformationen werden unter anderem zur Generierung der Kompilierreihenfolge verwendet. Diese Reihenfolge ist für viele Werkzeuge, wie Simulatoren und Synthesewerkzeuge, wichtig. Von IP-Anbietern bereitgestellte Syntheseskripte setzen häufig voraus, dass ein kompilierter VHDL-Entwurf vorliegt. Für den IP-Nutzer ist die Aufgabe, die Kompilierungsreihenfolge ohne Werkzeugunterstützung festzustellen, mit hohem Zeitaufwand verbunden, da der IP-Nutzer in der Regel kein Wissen über die Struktur und Hierarchie des IP-Moduls hat.

Als Werte für das XML-Metadatenattribut `referencedUnits` werden die IDs der entsprechenden logischen Zieleinheiten eingetragen.

Das UML-Modell aus Abbildung 10 auf Seite 55 erlaubt die Generierung eines XML-Schemas, das zur Validierung von IPQ-Metadaten genutzt wird. Die aus dem UML-Modell in Abbildung 10 auf Seite 55 generierte Schemadatei ist in Abbildung 14 grafisch dargestellt. Diese Abbildung zeigt, dass ein IP-Modul im IPQ-Integrationsdatenformat aus mehreren Bibliotheken bestehen kann. Eine Bibliothek darf mehrere Zellen enthalten. Als besondere Zelle ist die „Global Cell“ zu erkennen, in der gemeinsam genutzte Daten der gesamten Bibliothek enthalten sind. Das können beispielsweise Dokumentation, Technologiebibliotheken usw. sein.

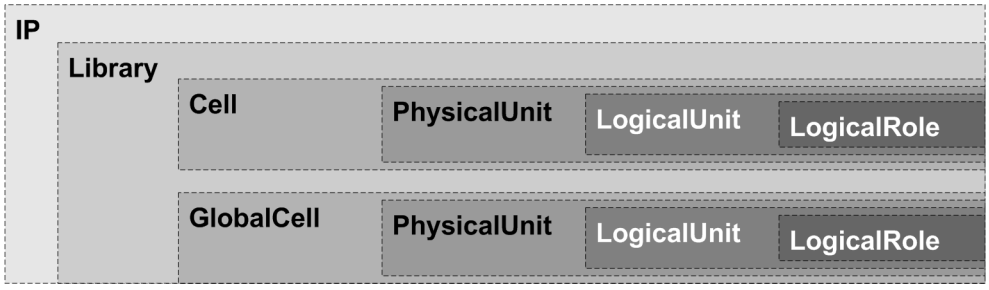


Abbildung 14: IPQ Integrationsdaten – XML-Schema

Diese Struktur ergibt sich aus dem Vorschlag in [25], welcher der Ausgangspunkt für die Entwicklung des IPQ-Integrationsdatenformats war. Eine Zelle wiederum besteht aus physikalischen Einheiten. Genau eine logische Einheit ist einer physikalischen Einheit zugewiesen. Einer logischen Einheit können wiederum mehrere logische Rollen und logische Teile zugewiesen werden.

Ein Ausschnitt eines weiteren, generierten XML-Schemas zeigt die logischen Teile eines Design-Compiler-Skripts in Abbildung 15. Das XML-Schema wird werkzeuggestützt aus dem erweiterten UML-Modell in Abbildung 10 auf Seite 55 generiert. Auf dieses XML-Schema wird im Schema aus Abbildung 14 verwiesen. Durch das Einbinden weiterer Schema-Dateien für jedes relevante Datenformat, können alle IPQ-Metadaten auf Konformität mit der IPQ-Formatspezifikation geprüft werden, ohne das allgemeine Schema bei Erweiterungen wesentlich modifizieren zu müssen.

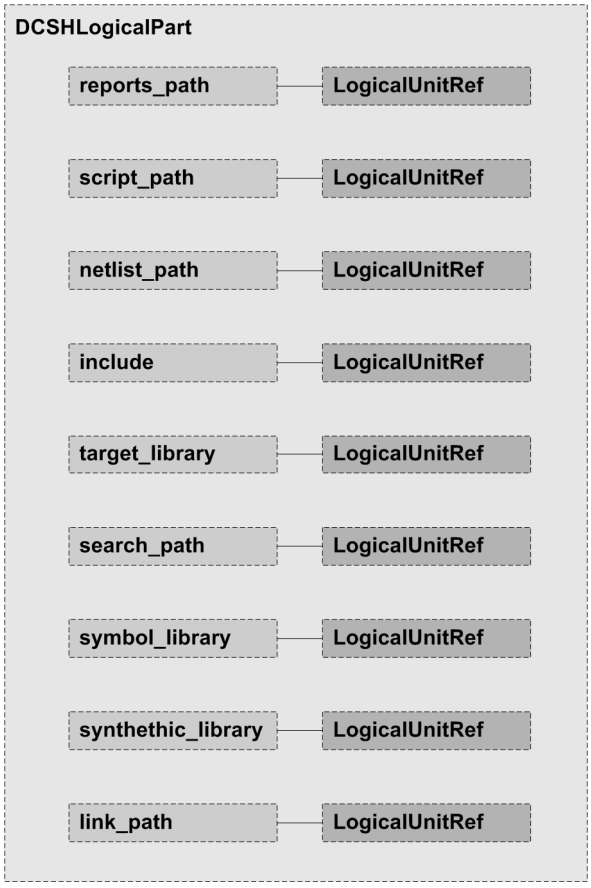


Abbildung 15: Logische Teile eines Design-Compiler-Skripts – XML Schema

Dateiformatübergreifende Abhängigkeiten

Am Beispiel eines Syntheseskripts werden dateiformatübergreifende Abhängigkeiten aufgezeigt. Abbildung 16 auf Seite 78 zeigt die Abhängigkeiten eines Design-Compiler-Syntheseskripts von der hierarchischen Struktur der werkzeugspezifischen Konfigurationsdateien (*.synopsys_dc.-*

setup), den HDL-Dateien und der entwurfsspezifischen Umgebungskonfiguration (Setup-Dateien) des IP-Moduls.

Die hierarchische Struktur der werkzeugspezifischen Konfigurationsdateien (*.synopsys_dc.setup*) wird folgendermaßen aufgelöst: Optionen, die im Installationsverzeichnis gesetzt werden, werden von Optionen im HOME-Verzeichnis des Entwicklers überschrieben und diese wiederum von den Optionen im aktuellen (Synthese-)Verzeichnis des IP-Moduls. Aus der dargestellten Struktur wird die Notwendigkeit abgeleitet, dass der IP-Anbieter alle für die Synthese benötigten Optionen in einer werkzeugspezifischen Konfigurationsdatei mit dem IP-Modul ausliefern muss. Zusätzlich muss beachtet werden, dass Optionen der HOME-Verzeichnisebene beziehungsweise Installationsverzeichnisebene während der abschließenden Qualifizierung (vgl. Abschnitt 4.2.2) deaktiviert werden, um die Unabhängigkeit der werkzeugspezifischen Konfigurationsdatei von der Umgebung des IP-Anbieters prüfen zu können.

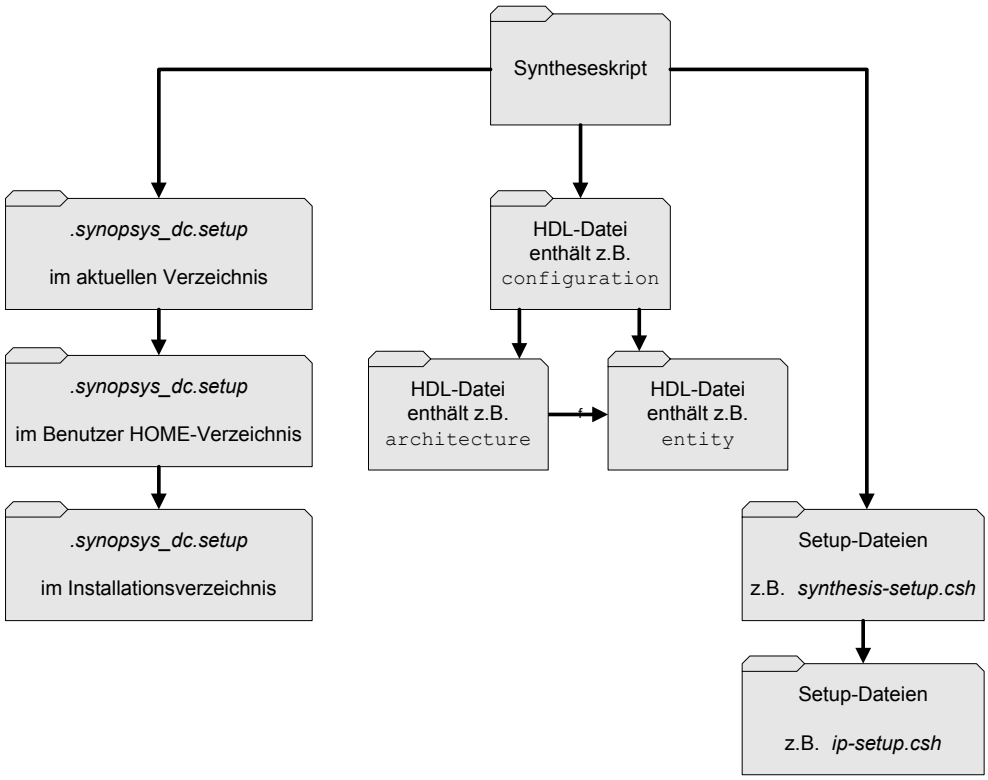


Abbildung 16: Abhängigkeitshierarchie eines Design-Compiler-Syntheseskripts

Tabelle 13 gibt eine Übersicht über die erlaubten Kombinationen der Dateiformate in der werkzeugspezifischen Abhängigkeitshierarchie.

DC_shell Modus	Synopsys root	User Home	Aktuelles Arbeitsverzeichnis
Tcl-Modus	Tcl-s	Tcl	Tcl
dcsh-Modus	Tcl-s	dcsh	dcsh
	Tcl-s	Tcl-s	dcsh
	Tcl-s	Tcl-s	Tcl-s

Tabelle 13: Erlaubte Kombinationen aus Kommandomodus und Setup Dateien

Entwurfsspezifische Konfigurationen werden durch Shellskripte (z. B. C-Shell-Skripte) erzeugt. Sowohl für HDL-Dateien als auch für Shellskripte können durch die Modellierung mit logischen Einheiten und Teilen im IPQ-Format ähnliche Abhängigkeitshierarchien wie die für die werkzeugspezifischen Konfigurationsdateien erzeugt werden.

Die logischen Abhängigkeiten des Syntheseskripts (z. B. von Umgebungsvariablen, Dateinamen, Entwurfsobjektnamen des HDL-Entwurfs) können ebenfalls auf diese Weise dateiformatübergreifend durch logische Einheiten und logische Teile modelliert werden. Während der Importphase werden die Abhängigkeiten aufseiten des IP-Nutzers entsprechend angepasst. Abhängigkeiten eines Syntheseskripts im Tcl-Format von Umgebungsvariablen werden beispielsweise durch die Modellierung des Konstrukts `get_unix_variable{…}` als logischer Teil der logischen Einheit Tcl-Skript dargestellt. Bei der Ersetzung werden die in Abbildung 16 dargestellten Hierarchien ausgewertet, um die ursprünglichen Werte der Variablen zu bestimmen und anhand der Umstrukturierungsinformationen zu ersetzen. Die Umstrukturierung und Anpassung werden in Abschnitt 6.3.4.5 detailliert beschrieben.

Welche Sprachkonstrukte als logische Teile modelliert werden, veranschaulicht Tabelle 19 im Anhang A.1 auf Seite 123. In der linken Spalte werden die Tcl-s-Kommandos genannt und in der rechten Spalte die Art der Abhängigkeit erläutert, die durch das jeweilige Kommando erzeugt wird. Der in den werkzeugspezifischen Konfigurationsdateien *.synopsys_dc.setup* verwendete Tcl-Sprachumfang ist eine Teilmenge des von Synopsys unterstützten Sprachumfangs im tcl-shell-Modus. Diese Tcl-s genannte Teilmenge wird sowohl vom dc-shell-, als auch vom tcl-shell-Modus des Design-Compilers unterstützt [79].

Zur Darstellung des Tcl-s-Sprachumfangs in den IPQ-Metadaten müssen lediglich 47% der Kommandos modelliert werden. Obwohl keine umfassende Untersuchung des Modellierungsaufwands durchgeführt wurde, wird doch die Aussage bestätigt, dass die zu implementierenden Parser nur einen geringen Anteil des jeweiligen Sprachumfangs erkennen müssen.

6.2 IP-Wiederverwendungsdatenbank

Im vorangegangenen Abschnitt wurde das Integrationsdatenformat vorgestellt. Für das im folgenden Abschnitt beschriebene Qualifizierungsframework müssen die Entwurfs- beziehungsweise Integrationsdaten eines IP-Moduls geeignet verwaltet werden.

Dabei ist zu beachten, dass während der Entwicklung eines IP-Moduls die Daten im Team verfügbar sein müssen und eventuelle Konflikte bei zeitgleichen Änderungen am Quellcode aufgelöst werden müssen. Nach der abschließenden Qualifizierung müssen die Daten zuverlässig gespeichert werden und ein einfacher Zugriff auf das aktuelle Release möglich sein. Nach einer Auslieferung müssen im Fall von „langen Qualifizierungsiterationen“ Mängel zurückverfolgt werden können und nach der Korrektur der Mängel in einem neuen Release bereitgestellt werden.

Diese Anforderungen werden von Versionierungssystemen erfüllt. Für den Zugriff auf die Daten eines IP-Moduls wurde in das IP-Qualifizierungsframework eine Schnittstelle zum Concurrent Versions System (CVS) integriert. Die Dateien eines abschließend qualifizierten IP-Moduls werden als auslieferbereite Version (engl. release) im CVS markiert. Dabei wird zwischen Markierungen für IP-Module, die die Standardqualifizierung bestanden haben und denen die nach kundenspezifischen Richtlinien qualifiziert wurden, unterschieden. Aufgrund der Release-Markierungen kann das Qualifizierungsframework IP-Modul-Daten für die Auslieferung (an bestimmte Kunden) von den Daten, die sich in der (Weiter-)Entwicklung befinden und entwurfsbegleitend qualifiziert werden, unterscheiden. Alle Daten werden beim IP-Anbieter im gleichen CVS verwaltet.

Neben der Möglichkeit auf die Daten eines IP-Moduls per CVS-Schnittstelle zuzugreifen, wird während der entwurfsbegleitenden Qualifizierungen häufig auf die Daten im Dateisystem des Entwicklers zugegriffen.

6.3 IP-Qualifizierungsframework

Im Folgenden wird die Implementierung des IP-Qualifizierungsframeworks (IPQ-Framework) beschrieben. Es bietet dem IP-Anbieter beziehungsweise IP-Nutzer automatisierte Qualifizierungs- und Auslieferungsunterstützung.

Im nächsten Abschnitt 6.3.1 wird zunächst die Architektur des Frameworks erläutert. Die Erweiterung des Frameworks um neue Datenformate ist Inhalt von Abschnitt 6.3.2. Das Framework analysiert die Daten eines IP-Moduls und erzeugt ein internes Objektmodell, welches die Grundlage für die weiteren Qualifizierungen ist. Nach Abschluss der Qualifizierung wird das intern verwendete Objektmodell als IPQ-Integrationsdatenformat serialisiert. Die Nutzung des Objektmodells für Qualitätsmessungen wird in Abschnitt 6.3.3 beschrieben. Qualitätsmessungen werden von so genannten Qualifizierungsexperten durchgeführt. Experten für die einzelnen Qualifizierungsphasen sind das Thema von Abschnitt 6.3.4.

6.3.1 Tafelarchitektur

Abbildung 17 auf Seite 81 zeigt die Tafelarchitektur (engl. Blackboard Architecture), die für die Implementierung des Qualifizierungs- und Auslieferungswerkzeugsatzes gewählt wurde. Diese Architektur ist ein bekannter Ansatz aus der künstlichen Intelligenz (KI) [37]. Der Ansatz ermöglicht die Integration von unabhängigen Experten, in diesem Fall Qualifizierungsmethoden, wie sie in den Kapiteln 4 und 5 beschrieben wurden. Die Tafelarchitektur kann man sich folgendermaßen vorstellen:

Eine Gruppe menschlicher Experten sitzt vor einer Tafel. Die Experten arbeiten kooperativ an einer Problemlösung, indem sie die Tafel zur Lösungsentwicklung verwenden. Das Lösen des Problems beginnt, wenn das Problem und die Ausgangsdaten an die Tafel geschrieben wurden. Die Experten betrachten die Tafel und warten auf die Möglichkeit ihr Fachwissen zur Problemlösung beizutragen. Findet ein Experte ausreichend Informationen um einen Beitrag zu leisten, wird er den

Beitrag an die Tafel schreiben und damit hoffentlich ermöglichen, dass andere Experten ihr Fachwissen anwenden können. Dieser Prozess des Hinzufügens von Beiträgen wiederholt sich so lange, bis das Problem gelöst ist.

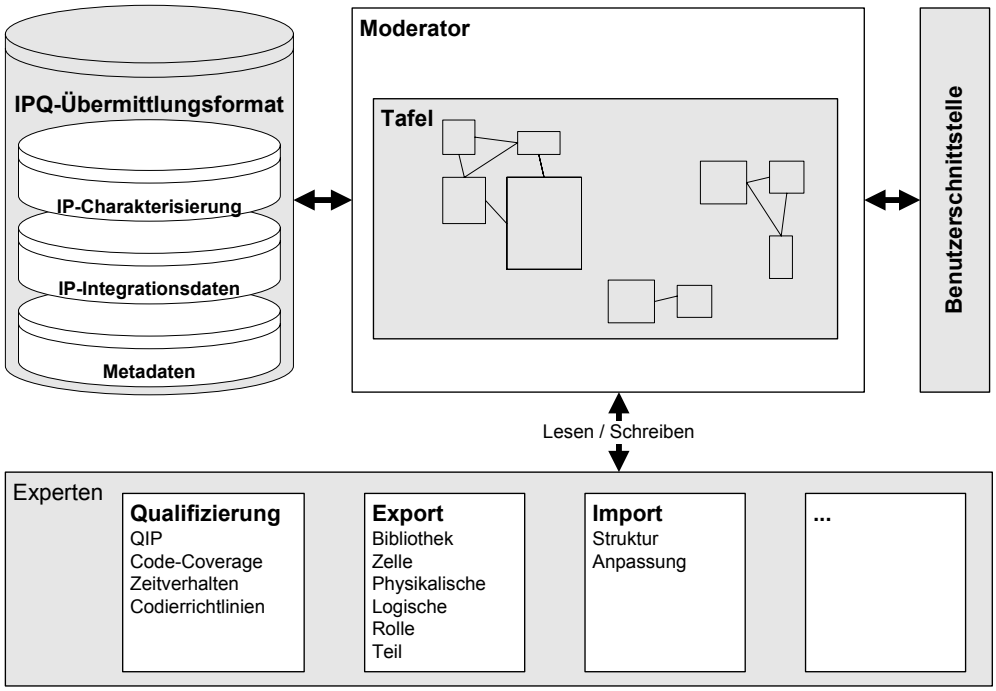


Abbildung 17: Tafelarchitektur

Wichtig ist an dieser Stelle, dass die Qualifizierung und Auslieferung von IP-Modulen das Fachwissen unterschiedlicher, voneinander unabhängiger Experten (Werkzeuge) benötigt. Die Lösungsgarantie ist durch einen Moderator, eine Ablaufsteuerung in Form einer Konfigurationsdatei, gegeben. Dadurch wird der Fall vermieden, dass es noch keine Lösung gibt, aber auch kein Experte mehr Beiträge liefern kann. Durch die externe Ablaufsteuerung ist ebenfalls ausgeschlossen, dass sich Experten gegenseitig behindern.

Die gewählte Architektur erlaubt das Zusammenführen einzelner Qualifizierungsbeiträge zu einer Gesamtqualitätssicht. Dadurch können spezifische Qualifizierungswerkzeuge (z. B. Linter, Code-Coverage, Synthese) zu einem Qualitätszertifikat beziehungsweise zu einer Auslieferung beitragen, die automatisch abgewickelt werden kann. Ändern sich Qualifizierungskriterien oder Auslieferungsbedingungen, können die entsprechenden Experten aktualisiert, neu hinzugefügt oder gelöscht werden, ohne die anderen Experten an ihrer Arbeit zu hindern. Auf diese Weise ist eine flexible Erweiterung des IPQ-Frameworks gegeben.

Die Benutzerschnittstelle in Abbildung 17 ist durch Kommandozeilenaufruf beziehungsweise eine grafische Benutzerschnittstelle (GUI) realisiert. Dadurch kann der Nutzer des IPQ-Frameworks die Anfangsdaten und die Aufgabe an die „Tafel“ schreiben. Mögliche Aufgaben, die vom IPQ-Framework gelöst werden können, sind beispielsweise das Zusammenstellen einer IP-Auslieferung, die Durchführung eines IP-Imports oder die

Qualifizierung eines IP-Moduls entsprechend der aktuellen Entwicklungsphase. Die Anfangsdaten sind das Quellverzeichnis des strukturierten IP-Moduls und die der Aufgabe entsprechende Framework-Konfigurationsdatei. Zur Lösung der Aufgaben wurden Experten implementiert. Ihre Funktionsweise wird ab Abschnitt 6.3.4 erklärt.

6.3.2 Erweiterung um neue Datentypen

Aufgrund der Tafelarchitektur kann das IPQ-Framework durch Experten erweitert werden. Sollen beispielsweise qualitätsrelevante Konstrukte eines neuen Datenformats erkannt werden, ist zum einen die Funktionalität des Experten, in diesem Fall eines Parsers, zu implementieren und andererseits das intern verwendete Objektmodell des Frameworks zu erweitern. Dadurch wird automatisch die neue logische Einheit mit ihren logischen Teilen den XML-Metadaten hinzugefügt. Die Erweiterungen werden, dem in Java realisierten Framework, durch Implementierungen in den folgenden `packages` hinzugefügt.

Für die Erweiterung des Objektmodells, wird ein neues `package de.fzi.ipq.model.<LogicalUnit>` integriert. Darin sind zwei Klassen und mehrere Interfaces zu implementieren:

- In der `<LogicalUnit>`-Klasse (z. B. VHDL) werden die Attribute der logischen Einheit definiert. Diese Attribute werden während der Serialisierung zu XML-Attributen in den XML-Metadaten.
- Die `<LogicalUnit>Creator`-Klasse (z. B. VHDLCreator) erzeugt ein `<LogicalUnit>`-Element des internen Objektmodells.
- Ein `<LogicalUnit>Unit`-Interface modelliert die allgemeine logische Einheit (z. B. `VHDLUnit`).
- Ein `<LogicalUnit>Part`-Interface modelliert den allgemeinen logischen Teil (z. B. `VHDLPart`).
- Jeweils ein `<LogicalUnit><LogicalPart>`-Interface für jeden konkreten `LogicalPart` (z. B. `VHDLArchitecture`) konkretisieren die jeweilige allgemeine `<LogicalUnit>Part`-Klasse.

Für die Implementierung der Objektmodellerweiterung wird dem Framework ein `package de.fzi.ipq.impl.model.<LogicalUnit>` hinzugefügt. Darin sind die folgenden Klassen zu implementieren:

- Eine `<LogicalUnit>PartImpl`-Klasse (z. B. `VHDLPartImpl`) zum Hinzufügen, Auslesen, Entfernen und Serialisieren von Verweisen auf logische Einheiten.
- Eine `<LogicalUnit>UnitImpl`-Klasse (z. B. `VHDLUnitImpl`) zum Hinzufügen, Auslesen und Entfernen von logischen Teilen sowie zum Setzen der Attributwerte.
- Jeweils eine `<LogicalUnit><LogicalPart>Impl`-Klasse (z. B. `VHDLArchitectureImpl`) konkretisiert die allgemeine `<LogicalUnit>PartImpl`-Klasse und implementieren das `<LogicalUnit><LogicalPart>`-Interface.

Zusätzlich zu den Quellcodeerweiterungen muss die neue logische Einheit in der Konfigurationsdatei für Modellerweiterungen *model-extensions.xml* registriert werden. Quellcode 4 zeigt einen Ausschnitt aus dieser Konfigurationsdatei. Im dargestellten Beispiel, wird das Modell um eine logische Einheit des C++-Headerformats erweitert. In den Zeilen 2-10 werden die Java-Klassen und -Interfaces der logischen Einheiten und Teile für die Modellerweiterung definiert. In den Zeilen 11-18 werden die Parser für die Serialisierung der logischen Einheiten und Teile eingetragen. Die Parser für die XML-Serialisierung müssen im `package de.fzi.ipq.impl.parser.<LogicalUnit>` implementiert werden.

```
1  <ModelExtension urn="http://www.fzi.de/ipq/extensions/hpp">
2    <Definitions>
3      <LogicalUnits>
4        <LogicalUnit type="HPP"
           interface="de.fzi.ipq.model.hpp.HPPUnit"
           class="de.fzi.ipq.impl.model.hpp.HPPUnitImpl"/>
5      </LogicalUnits>
6      <LogicalParts>
7        <LogicalPart type="HPPGuard"
           interface="de.fzi.ipq.model.hpp.HPPGuard"
           class="de.fzi.ipq.impl.model.hpp.HPPGuardImpl"/>
8      </LogicalParts>
9      <LogicalRoles/>
10   </Definitions>
11   <Serialization
       schemaLocation="http://www.fzi.de/ipq/extensions/core-
       schema.xsd" targetNamespace="http//www.fzi.de/ipq"
       defaultPrefix="hpp">
12     <LogicalUnits>
13       <LogicalUnit type="HPP" elementType="HPPUnitType"
           parser="de.fzi.ipq.impl.parser.hpp.HPPUnitParser"/>
14     </LogicalUnits>
15     <LogicalParts>
16       <LogicalPart type="HPPGuard"
           elementType="HPPGuardType"
           parser="de.fzi.ipq.impl.parser.hpp.HPPPartParser"/>
17     </LogicalParts>
18   </Serialization>
19 </ModelExtension>
```

Quellcode 4: Registrierung einer logischen Einheit in *model-extensions.xml*

Die Expertenfunktionalität wird in einem neuen `package de.fzi.ipq.impl.action.<experte>` (z. B. `package de.fzi.ipq.impl.action.pack.vhdl`) implementiert. Dabei ist darauf zu achten, dass die Analysen eine bestimmte Reihenfolge einhalten müssen. Die Ablaufsteuerung erfolgt klassenweise. Daher sind unterschiedliche Klassen zu implementieren. Bestimmte Operationen können beispielsweise erst nach einer

Quellcodeanalyse durchgeführt werden. Für die Abhängigkeitsmodellierung wird aus diesem Grund eine eigene Quellcodeanalyse-Klasse implementiert, die in einem ersten Schritt alle relevanten logischen Teile in einer logischen Einheit erfasst. In einer Abhängigkeitsanalyse-Klasse werden anschließend die entsprechenden Verweise von den logischen Teilen (Quelle) auf die logischen Einheiten (Ziel) in das Objektmodell eingetragen.

Die Experten-Klassen (z. B. Quellcodeanalyse, Abhängigkeitsanalyse) müssen dann entsprechend ihrer Reihenfolge in der Ablaufsteuerung registriert werden. Quellcode 5 zeigt als Beispiel die Registrierung des Rollenzuweisungsexperten.

```
1  <Action name="role"
     class="de.fzi.ipq.impl.action.pack.role.RoleAgent">
2    <Param name="RuleFile"
       value=".\config\config-ip-anbieter-pack.xml"/>
3  </Action>
```

Quellcode 5: Expertenregistrierung in der Ablaufsteuerung

Die Ablaufsteuerung ist als XML-Konfigurationsdatei realisiert. Für die zu lösenden Aufgaben gibt es jeweils angepasste Framework-Konfigurationsdateien. Ein Experte wird mit dem XML-Element Action und den Attributen Name und seinem Java-Klassennamen registriert. Optional steht das untergeordnete Param-Element für eine evtl. benötigte Expertenkonfiguration bereit. Im Beispiel wird eine firmenspezifische Regeldatei übergeben, in der die IP-Struktur des IP-Anbieters spezifiziert ist. Quellcode 6 zeigt den hierarchischen Aufbau der Konfigurationsdatei ohne untergeordnete Regeln. Auf oberster Hierarchieebene wird mit dem Namensattribut die zu konfigurierende Expertenaufgabe genannt, die Firma und das Werkzeug, in welcher der Experte verwendet wird. Die untergeordneten Regeln dienen dem Nicht-Beachten physikalischer Einheiten beim Zusammenstellen einer IP-Auslieferung. Das betrifft beispielsweise CVS-Dateien, Dateien, die aufgrund von Werkzeugabstürzen generiert wurden usw. Des Weiteren werden die Regeln zum Erkennen der logischen Einheiten und für die Zuweisung der logischen Rollen spezifiziert.

```
1  <Rules name="export" company="ip-anbieter" tool="IPQ-Framework">
2    <Rule name="Ignore"/>
3    <Rule name="TypeRecognition"/>
4    <Rule name="RoleAssignment"/>
5  </Rules>
```

Quellcode 6: Export-Expertenkonfiguration

Das auf diese Weise konfigurierte Framework erzeugt ein Metadaten-Objektmodell, auf dem die nachfolgenden Qualifizierungs- und Zertifizierungsphasen basieren. Das Objektmodell repräsentiert einen Graph, dessen Knoten die Elemente des IPQ-Integrationsdatenformats sind. Die Knotentypen werden durch unterschiedliche Java-Klassen dargestellt.

6.3.3 Nutzung des Objektmodells

Auf das Objektmodell greifen die verschiedenen qualitätsrelevanten Operationen zu, wie Erfassen der Abhängigkeiten, Erzeugen von Metainformationen, Export und Import. Gemeinsam ist diesen Operationen, dass das Objektmodell traversiert werden muss und bestimmte Methoden für definierte Objekte durchgeführt werden müssen. Zur Trennung des Objektmodells von den Operationen ist das Besucher-Entwurfsmuster [1] implementiert worden. Dadurch wird eine Modifikation des Objektmodells vermieden, falls neue Operationen, die keine Erweiterung des Objektmodells benötigen, implementiert werden sollen,

Dabei traversiert ein Besucher, als abstrakte Klasse implementiert, das Objektmodell, wobei die dem besuchten Element entspreche Methode aufgerufen wird. Der Vorteil ist, dass die Elemente des Objektmodells nur eine Methode zum Entgegennehmen einer abstrakten Besucherklasse bereitstellen müssen. Operationen auf dem Objektmodell werden in den konkreten Besucherklassen als Methode implementiert und machen daher keine Änderung des Objektmodells notwendig. Die Möglichkeit zum Aufrufen der Methode, die zum Knotentyp passt, wird durch eine Double-Dispatch genannte Technik [1] bereitgestellt. Double-Dispatch bedeutet, dass die auszuführende Operation durch die Typen zweier Empfänger, des Besuchers und des besuchten Elements, bestimmt wird.

In der Implementierung des IPQ-Frameworks durchläuft (traversiert) die abstrakte Besucherklasse `FilterVisitor` das Objektmodell. Die eigentliche Funktionalität, z. B. die Erkennung qualitätsrelevanter Konstrukte, ist in konkrete `Filter`-Klassen ausgelagert, die von der abstrakten Klasse `Filter` erben. So ist die Suche nach bestimmten Elementen in der `Finder`-Klasse implementiert. Auch die `Serializer`-Klasse implementiert die abstrakte Besucherklasse `FilterVisitor`. In der `Serializer`-Klasse ist die konkrete Serielaisierung für jedes Element des internen Objektmodells nach XML implementiert.

Jedes Element des internen Objektmodells implementiert eine `accept`-Methode (z. B. Quellcode 7) mit der sich das Objekt selbst an den abstrakten Besucher übergibt. Aufgrund der Double-Dispatch Technik wird die entsprechende Operation ausgeführt, wie z. B. das Zurückliefern eines Bibliotheksobjekts im Falle der `Finder`-Klasse oder der Serialisierung eines Bibliotheksobjekts nach XML im Falle der `Serializer`-Klasse.

```
1  public void accept(Visitor visitor) throws IPQException
2  {     visitor.visitLibrary(this);   }
```

Quellcode 7: Akzeptanz eines konkreten Bibliotheksbesuchers

Für die Bereitstellung neuer Funktionalität muss also lediglich eine neue Klasse bereitgestellt werden, die von der `Filter`-Klasse erbt.

Ein Problem stellt das Ausführen funktional gleicher Methoden mit unterschiedlichen Signaturen dar. Falls z. B. unterschiedliche Elementsuchmethoden bereitgestellt werden sollen. Oft benötigt wird zum einen die Rückgabe aller Elemente eines bestimmten Typs, z. B. die Rückgabe aller logi-

schen VHDL-Einheiten, und zum anderen die Rückgabe aller Elemente eines bestimmten Typs die noch eine weitere Eigenschaft erfüllen, z. B. die Rückgabe aller logischen VHDL-Einheiten, die eine `Configuration` als logischen Teil enthalten. Für dieses Problem gibt es zwei Lösungsmöglichkeiten:

1. Die Implementierung in jeweils einer eigenen Klasse erlaubt die Pflege nur eines einzigen Rückgabeobjekts.
2. Sollen mehrere Methoden in einer einzigen Klasse implementiert werden, müssen von den Methoden ggf. unterschiedliche Rückgabeobjekte gepflegt werden, da die Rückgabemethoden nicht unterscheiden können, von welcher Methode sie aufgerufen wurden, denn am Ende der Traversierung ist die aufgerufene Methode nur noch für die Rückgabe des „richtigen“ Objekts zuständig.

6.3.4 Qualifizierungsexperten

Nach der Einführung in den generellen Aufbau des IPQ-Frameworks werden in den folgenden Abschnitten die implementierten Experten näher betrachtet. Der Autor hat Qualifizierungsexperten für QIP-Kriterien, wie Code-Coverage, Zeitverhalten und Codierrichtlinien implementiert.

Für den Export beziehungsweise Import von IP-Modulen wurden Experten zur Generierung des IPQ-Integrationsdatenformats aus der IP-Struktur beziehungsweise zur Generierung der IP-Struktur aus dem IPQ-Integrationsdatenformat implementiert.

6.3.4.1 Integration externer Qualifizierungsexperten

In diesem Abschnitt wird beschrieben, auf welche Weise externe, bereits verfügbare Qualifizierungswerkzeuge als Experten in das IPQ-Framework integriert werden können. Beschrieben werden Experten für die entwurfsbegleitende beziehungsweise abschließende Qualifizierungsphase.

In diesen Qualifizierungsphasen werden Code-, Verifikations- und Entwurfsqualität geprüft. Code- und Verifikationsqualität werden in den folgenden beiden Abschnitten behandelt. Alle Werkzeugaufrufe erfolgen automatisch. Für die definierten Qualifizierungsphasen sind die jeweils benötigten Werkzeuge registriert. Sie werden über Werkzeug-Wrapper (*Makefile*s) aufgerufen und sind so an das Framework gekoppelt. Lediglich technologiespezifische Beschränkungen, falls eine andere Technologiebibliothek als die für die Standardqualifizierung verwendet werden soll, müssen durch Editieren des Syntheseskripts manuell bearbeitet werden.

Das *Makefile* erzeugt die für das Werkzeug benötigte Umgebung, indem eine werkzeugspezifische Installationsdatei aufgerufen wird. Eine solche Datei wird für alle Qualifizierungswerkzeuge im Werkzeuginstallationspfad vorausgesetzt. In der Installationsdatei wird der Werkzeuginstallationspfad gesetzt, die `Path`-Variable angepasst und der Lizenzserver angegeben. Darüber hinaus werden weitere, werkzeugspezifische Umgebungsvariablen gesetzt.

Nach dem Erzeugen der Umgebung, werden die `Targets` definiert und mit welchen Werkzeugaufrufen sie erzeugt werden. Dadurch werden u.a. auch unnötige, teilweise sehr langwierige Qualifizierungen (Synthese, Verifikation) vermieden, da das *Makefile* zunächst überprüft, ob die Quellen aktueller als die Targets sind und nur dann den Qualifizierungsschritt ggf. erneut durchführt.

Die einzelnen Qualifizierungswerkzeuge erhalten als Eingabedaten die für sie relevanten Entwurfsdateien. Zusätzlich werden weitere Dateien, z. B. In-/Output-Pattern für die Simulation beziehungsweise Code-Coverage, und die entsprechenden werkzeugspezifischen Konfigurationsdateien, z. B. Coding-Style-Regeln im Falle der Quellcode-Analyse, übergeben. Es sind keine Konfigurationsdateien erforderlich, wenn die Regeln im Qualifizierungswerkzeug beziehungsweise dem Aufruf-Wrapper fest codiert sind.

Durch die *Makefile*-gesteuerten Werkzeugaufrufe werden die für die Qualifizierungsberichtserzeugung benötigten Daten bereitgestellt, die in der Zertifizierungsphase von Berichtsgeneratoren ausgewertet werden. Das externe Werkzeug ist über die generierten Ausgabedaten des Qualifizierungswerkzeugs mit dem IPQ-Framework gekoppelt. Der erzeugte Qualifizierungsbericht wird dann noch einmal manuell mit der Spezifikation beziehungsweise Dokumentation verglichen. Tauchen Unstimmigkeiten auf die nicht vom Designer gerechtfertigt und dokumentiert sind, wird eine neue Qualifizierungsiteration gestartet, nachdem das IP-Modul dem entsprechenden Entwickler zur Nachbesserung übergeben wurde (Lerneffekt beim Entwickler).

Bedingt durch die Kommunikation über generierte Ausgabedateien und Leseschnittstellen, ist die Interaktion zwischen dem IPQ-Framework und den externen Qualifizierungsexperten nur eingeschränkt möglich. Ein solches System wird lose gekoppeltes System genannt. Im Gegensatz dazu gibt es eng gekoppelte Systeme. Solche Systeme benötigen eine in jedem Experten implementierte API. Die Interaktionsmöglichkeiten zwischen den Experten werden dadurch gesteigert. Das lose gekoppelte System hat jedoch insbesondere bei Qualifizierungswerkzeugen, die von Fremdanbietern erworben wurden, den Vorteil, dass vorhandene Werkzeuge ohne Quellcodeänderungen verwendet werden können.

Die Qualitätsmessungen werden unter dem Betriebssystem UNIX durchgeführt, da für dieses Betriebssystem alle benötigten Qualifizierungswerkzeuge verfügbar sind. Alle für die Qualifizierung benötigten Werkzeuge sind im Verzeichnis */tools* installiert. Der Link */tools/<tool_name>* verweist jeweils auf das aktuelle Versionsverzeichnis des Werkzeugs. Es ist aber auch möglich eine bestimmte Werkzeugversion aufzurufen, indem das Verzeichnis */tools/<tool_name>-<version>* direkt benutzt wird.

Für eine Qualitätsmessung wird das IP-Modul in das Verzeichnis *$IP_HOME/$IP_NAME* aus dem CVS ausgecheckt (`cvs checkout $IP_NAME`). Das IP-Modul wird dem Qualifizierungsframework über Umgebungsvariablen bekannt gemacht, indem die Umgebung durch ein C-Shell-Skript bereitgestellt wird (`source $IP_HOME/$IP_NAME/-`

`setup.$IP_NAME.csh`). Zusätzlich müssen die Umgebungen für die Qualifizierungswerkzeuge erzeugt werden (`source /tools/-<tool_name>/setup.<tool_name>.csh`). Beispielsweise werden mit den Befehlen:

```
source /tools/modelsim/setup.modelsim.csh
source /tools/vncheck/setup.vncheck.csh
```

Die Umgebungen für die jeweils aktuellen Versionen von Modeltechs Modelsim und TransEDAs VN-Check erzeugt. VN-Check gehört zur Verification Navigator (VN) Werkzeugfamilie, in der auch VN-Cover für Coverage-Messungen enthalten ist. Die gesamte VN-Werkzeugfamilie ist abhängig von HDL-Simulatorbibliotheken. Aus diesem Grund wird zusätzlich die Modelsim Simulatorumgebung erzeugt.

Während der Installation des IPQ-Frameworks für ein IP-Modul werden für die einzelnen Qualifizierungsschritte aus *Makefile*-Vorlagen konkrete *Makefile*s generiert und in den Verzeichnissen

$IP_HOME/$IP_NAME/quality/<Qualifizierungsschritt>

abgelegt. Zur Durchführung einer entwurfsbegleitenden Qualitätsmessung wechselt der Entwickler manuell in das entsprechende Verzeichnis und ruft `make` auf. Es besteht aber auch die Möglichkeit, dass die Qualifizierung durch das IPQ-Framework gesteuert wird und diese Schritte automatisch durchführt. Dies ist beispielsweise bei der abschließenden Qualifizierung der Fall.

Automatische Code-Qualifizierung

Im Folgenden wird der automatische Ablauf einer Codequalifizierung beschrieben. In *$IP_HOME/$IP_NAME/quality/code* befindet sich das *Makefile* für die Quellcodequalifizierung. Es erzeugt den Quellcode-Qualifizierungsbericht im selben Verzeichnis. *$IP_HOME* ist das Wurzelverzeichnis des zu qualifizierenden IP-Moduls. Der folgende Quellcode 8 zeigt das generierte *Makefile* für die automatische Quellcodequalifizierung.

```
1  .SUFFIXES:
2
3  INPUTDIR = $(IP_HOME)/$IP_NAME/vhdl
4  INPUTFILES = $(INPUTDIR)/*.vhd
5
6  OUTPUTDIR = $(IP_HOME)/$IP_NAME/quality/code
7  OUTPUTFILE = codequality
8  OUTPUTFORMAT = htm # <htm|rpt>
9
10 RULEDIR = $(VNAVIGATOR)/vncheck_databases/vncheck-vh
11 RULEFILE1 = BestPractices.rdb
12 RULEFILE2 = DFT.rdb
13 RULEFILE3 = FPGA.rdb
14 RULEFILE4 = FSM.rdb
15 RULEFILE5 = OpenMORE.rdb
16 RULEFILE6 = RMM.rdb
```

```
17  RULEFILE7 = synth.rdb
18  RULEFILE8 = synthlite.rdb
19  RULEFILE9 = vhdl2verilog.rdb
20
21  OPTIONS= -rule $(RULEDIR)/$(RULEFILE1) \
22        -rule $(RULEDIR)/$(RULEFILE2) \
23        -rule $(RULEDIR)/$(RULEFILE4) \
24        -rule $(RULEDIR)/$(RULEFILE5) \
25        -rule $(RULEDIR)/$(RULEFILE6) \
26        -rule $(RULEDIR)/$(RULEFILE7) \
27        -nolimit \
28        -ignorepragmas \
29        -html \
30        -checklibraries
31
32  TOOL_OPTIONS = -sim modelsim_vhdl5_5
33
34  $(OUTPUTDIR)/$(OUTPUTFILE).$(OUTPUTFORMAT): $(INPUTFILES)
35        mkdir -p $(OUTPUTDIR)
36        vncheck-vh $(INPUTFILES) -output $@ $(OPTIONS)
    $(TOOL_OPTIONS)
```

Quellcode 8: Makefile für die automatische Quellcodequalifizierung

Ein Ausschnitt einer firmenspezifischen Verzeichnisstruktur eines IP-Moduls ist in Abbildung 18 dargestellt.

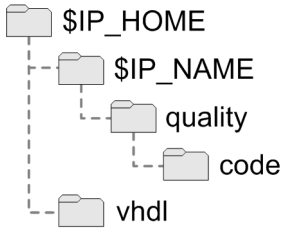


Abbildung 18: IP-Verzeichnisstruktur (Ausschnitt)

Automatische Code-Coverage-Messung

Dieser Abschnitt beschreibt die automatische Durchführung einer Code-Coverage-Messung. Der nachstehende Quellcode 9 zeigt das generierte *Makefile* für die automatische Code-Coverage-Messung.

```
1  SHELL = /bin/csh
2
3  COV_TOOL_ENV = /tools/vnavigator/setup.vnavigator.csh
4  SIM_COV_TOOL_ENV = /tools/modelsim-5.5f/setup.modelsim-
   5.5f.csh
5
6  INPUTDIR = $(IP_HOME)/$(IP_NAME)/vhdl
7
8  OUTPUTDIR = $(IP_HOME)/$(IP_NAME)/quality/coverage
```

```
9  OUTPUTFILE = $(IP_NAME)_coverage_quality_report
10 OUTPUTFORMAT = htm
11
12 $(OUTPUTDIR)/$(OUTPUTFILE).$(OUTPUTFORMAT): $(INPUTDIR)/*.vhd
   $(INPUTDIR)_tb/*.vhd *.inp *.out
13
14 source $(COV_TOOL_ENV); \
15 source $(SIM_COV_TOOL_ENV); \
16
17 echo Checking code coverage quality of $(IP_NAME) ...; \
18
19 mkdir -p $(OUTPUTDIR); \
20
21 vnlib ; \
22
23 vnvhdl -noinstrument $(INPUTDIR)/ip_conv_pkg.vhd
   $(INPUTDIR)/ip_io_pkg.vhd $(INPUTDIR)/ip_pkg.vhd; \
24
25 vnvhdl $(INPUTDIR)/ip_cg.vhd $(INPUTDIR)/ip_cg_rtl.vhd -
   declaration $(INPUTDIR)/ip_cg.vhd; \
26
27 vnvhdl $(INPUTDIR)/ip_ci.vhd $(INPUTDIR)/ip_ci_rtl.vhd -
   declaration $(INPUTDIR)/ip_ci.vhd; \
28
29 vnvhdl $(INPUTDIR)/ip_do.vhd $(INPUTDIR)/ip_do_rtl.vhd -
   declaration $(INPUTDIR)/ip_do.vhd; \
30
31 vnvhdl $(INPUTDIR)/ip_do_top.vhd $(INPUTDIR)/ip_do_top_rtl.vhd
   -declaration $(INPUTDIR)/ip_do_top.vhd; \
32
33 vnvhdl $(INPUTDIR)/ip_os.vhd $(INPUTDIR)/ip_os_rtl.vhd -
   declaration $(INPUTDIR)/ip_os.vhd; \
34
35 vnvhdl $(INPUTDIR)/ip.vhd $(INPUTDIR)/ip_rtl.vhd -declaration
   $(INPUTDIR)/ip.vhd; \
36
37 vnvhdl $(INPUTDIR)/ip_cg_cfg.vhd $(INPUTDIR)/ip_ci_cfg.vhd
   $(INPUTDIR)/ip_do_cfg.vhd $(INPUTDIR)/ip_do_top_cfg.vhd
   $(INPUTDIR)/ip_os_cfg.vhd $(INPUTDIR)/ip_cfg.vhd; \
38
39 vnvhdl -noinstrument $(INPUTDIR)_tb/ip_tb_pkg.vhd; \
40
41 vnvhdl -noinstrument $(INPUTDIR)_tb/read_pages.vhd
   $(INPUTDIR)_tb/read_pages_beh.vhd; \
42
43 vnvhdl -noinstrument $(INPUTDIR)_tb/read_pages_cfg.vhd; \
44
45 vnvhdl -noinstrument $(INPUTDIR)_tb/ip_tb.vhd
   $(INPUTDIR)_tb/ip_tb_beh.vhd; \
```

```
46
47  vnvhdl -noinstrument $(INPUTDIR)_tb/ip_tb_cfg.vhd; \
48
49  vnsim -overwrite ip_tb_cfg; \
50
51  vnresults -style html -htmlfile
    $(OUTPUTDIR)/$(OUTPUTFILE).$(OUTPUTFORMAT) -htmldir
    $(OUTPUTDIR)/$(OUTPUTFILE)
```

Quellcode 9: Makefile für automatische Code-Coverage-Messung

Während der Installation des IP-Qualifizierungsframework für ein IP-Modul wird automatisch ein *Makefile* wie in Quellcode 9 dargestellt generiert. Im Werkzeuginstallationsverzeichnis befindet sich eine Vorlage, die automatisch an das Projekt angepasst wird. Auf die notwendigen Anpassungen der Vorlage wird später noch eingegangen. Es wird davon ausgegangen, dass es sich um kein neues Projekt handelt und daher bereits die notwendigen VHDL-Dateien vorliegen und diese durch das IPQ-Framework analysiert werden können.

Die erste Zeile bewirkt, dass an Stelle der meist voreingestellten Bourne-Shell die C-Shell zur Ausführung der Kommandos verwendet wird. Für die Erzeugung der Werkzeugumgebung waren in der Fallstudie bereits C-Shell-Skripte vorhanden. Diese Skriptnamen werden in den Zeilen 3 und 4 an Makros übergeben. Weitere Makrodefinitionen folgen bis zur Target-Definition in Zeile 12. Das Ziel (Target) ist ein HTML-Code-Coverage-Bericht. Seine Erzeugung ist abhängig von den VHDL-Dateien im Entwurfsverzeichnis *vhdl*, den VHDL-Dateien im Testbench-Verzeichnis *vhdl_tb* und sowie den Eingabe- (**.inp*) und Referenz-Ausgabemustern (**.out*). Durch diese Abhängigkeitserklärung wird durch die `make` eigene Funktionalität vermieden, dass ein Qualifizierungsbericht neu erzeugt wird, falls die Quelldateien nicht verändert wurden. Das kann insbesondere bei der Code-Coverage-Messung, in deren Verlauf eine komplette Simulation durchgeführt werden muss, erhebliche Zeiteinsparungen bei der Qualifizierung mit sich bringen.

In den beiden folgenden Zeilen werden die Werkzeugumgebungen erzeugt. Wichtig ist, dass alle Kommandos in einer logischen Kommandozeile ausgeführt werden, dafür müssen die Zeilen mit „`; \`“abgeschlossen werden. Andernfalls verwendet `make` für jede Kommandozeile eine eigene Subshell. Das hätte zur Folge, dass die erzeugten Werkzeugumgebungen in den Folgezeilen nicht mehr verfügbar sind. Die Werkzeugumgebungen werden im *Makefile* erzeugt, da sie dadurch automatisch mit Beendigung des *Makefile*s die Entwurfsumgebung nicht mit zusätzlichen Umgebungsvariablen belasten und so Konflikte zwischen konkurrierenden Umgebungen weitgehend vermieden werden können.

In Zeile 19 wird das Ausgabeverzeichnis für den Qualifizierungsbericht in der IP-Verzeichnisstruktur erzeugt. Da in Zeile 21 nur das Kommando `vnlib` ausgeführt wird, werden die Kommandoparameter von der werkzeugspezifischen Konfigurationsdatei `vnavigator.par` bestimmt. Sie

wird von den Werkzeugaufrufen `vnlib`, `vnvhdl`, `vnsim` und `vnresults` ausgewertet. Dabei wird eine hierarchische Reihenfolge beachtet. Zunächst wird die `vnavigator.par` Datei im Installationspfad (firmenspezifische Grundeinstellungen) ausgewertet. Ihre Werte können durch eine `vnavigator.par` Datei im aktuellen Arbeitsverzeichnis (projekt- beziehungsweise entwurfsspezifische Grundeinstellungen) ergänzt beziehungsweise überschrieben werden. Die höchste Priorität haben schließlich Kommandozeilenparameter mit denen individuelle Abweichungen für einzelne Dateien durchgesetzt werden können. Quellcode 10 zeigt einen Ausschnitt aus `vnavigator.par`. Zeile 1 wählt für `vnlib` den VHDL-Modus und Zeile 2 bestimmt das Modeltech Modelsim 5.5 Kompilierungsfomat. Durch Zeile 3 wird ein physikalisches Verzeichnis `./vlib` im aktuellen Arbeitsverzeichnis erzeugt und durch Zeile 4 mit dem logischen Bibliotheksnamen `coverlib` verknüpft.

```
1  vnlib vnvhdl
2  vnlib simulator modelsim_vhdl5_5
3  vnlib create coverlib=./vlib
4  vnlib setwork coverlib
```

Quellcode 10: Parameter für vnlib (vnavigator.par - Ausschnitt)

Im folgenden Quellcode 11 sind die `vnvhdl`-Optionen gesetzt, die für die Messung der relevanten Coverage-Messungen nach QIP 1.11 benötigt werden.

```
1  vnvhdl instrument
2  vnvhdl statement
3  vnvhdl branch
4  vnvhdl path
5  vnvhdl condition
6  vnvhdl nosubexpressions
7  vnvhdl notriggering
8  vnvhdl concurrent
9  vnvhdl toggle
10 vnvhdl unit
11 vnvhdl data_dir .
12 vnvhdl pragma_synthesis_off synopsys translate_off
13 vnvhdl pragma_synthesis_on synopsys translate_on
```

Quellcode 11: Parameter für vnvhdl (vnavigator.par - Ausschnitt)

In die durch Quellcode 10 auf Seite 92 erzeugte Bibliothek, werden im Folgenden durch das `vnvhdl` Kommando die VHDL-Dateien für den verwendeten Simulator kompiliert und gegebenenfalls instrumentiert. Da die Coverage-Messung für Packages und Testbenches uninteressant ist, werden diese Dateien mit der `-noinstrument` Option von der Instrumentierung ausgeschlossen. Die Instrumentierung erzeugt eine Kopie der VHDL-Dateien und reichert sie mit Informationen für die Coverage-Messung an. Sind VHDL `Architecture` und `Entity` in getrennten Dateien beschrieben, müssen bei der `Architecture` Kompilierung die

`Port`-Informationen mit der `-declaration` Option und nachgestelltem `Entity`-Dateipfad bekannt gemacht werden.

Die Entwurfseinheiten (`Entity`, `Architecture`, `Configuration` usw.), die an die `vnvhdl` und `vnsim` Kommandos übergeben werden müssen, sind stark vom Aufbau des IP-Moduls abhängig. Daher müssen die Zeilen 23 bis 49 in Quellcode 9 modulspezifisch generiert und in die werkzeugspezifische Vorlage eingefügt werden. Für die automatische Generierung muss die VHDL-Hierarchie aufgelöst werden. Das IP-Qualifizierungsframework kann die Hierarchie aus den IPQ-Metadaten extrahieren. Die Analyse der VHDL-Dateien durch das Qualifizierungsframework erkennt die einzelnen Entwurfseinheiten und kann daher den notwendigen Code für diese Zeilen generieren. Mit dem abschliessenden Kommando `vnresults` wird schließlich der HTML-Qualifizierungsbericht im modulspezifisch definierten Ausgabeverzeichnis erzeugt.

```
1  vnresults statement
2  vnresults branch
3  vnresults condition
4  vnresults path
5  vnresults toggle
6  vnresults excluded
7  vnresults fsmarc
8  vnresults fsmstate
9  vnresults fsmpath
10 vnresults activity
11 vnresults data_dir .
12 vnresults fec_diagnostic
13 vnresults fec_choices 5
14 vnresults nosingle_edge
15 vnresults notrim_rootname
16 vnresults condition_criterion expr
17 vnresults combinations 4096
18 vnresults statement_target 100
19 vnresults branch_target 100
20 vnresults condition_target 98
21 vnresults trace_target 95
22 vnresults toggle_target 98
23 vnresults triggering_target 95
24 vnresults path_target 75
25 vnresults excluded_target 95
26 vnresults fsm_arc_target 100
27 vnresults fsm_state_target 100
28 vnresults fsm_path_target 100
29 vnresults statement_signoff 95
30 vnresults branch_signoff 95
31 vnresults condition_signoff 95
32 vnresults toggle_signoff 95
33 vnresults path_signoff 75
34 vnresults state_signoff 80
```

```
35 vnresults fsmpath_signoff 60
36 vnresults arc_signoff 80
```

Quellcode 12: Parameter für vnresults (vnavigator.par - Ausschnitt)

Quellcode 12 zeigt die Einstellungen für die Messung für die von der VSIA in QIP Version 1.11 auf Tabellenblatt „Digital Soft-IP“ in den Zeilen 263 bis 268 geforderten Coverage-Ergebnisse. Diese Einstellungen werden in der Datei `vnavigator.par` konfiguriert.

6.3.4.2 Experten für die IP-Zertifizierung

Während der IP-Zertifizierungsphase wird das IP-Modul auf Übereinstimmung mit definierten Kriterien überprüft, wie sie beispielsweise durch den VSIA-QIP-Standard, VSIA-Deliverable Document und RMM festgelegt sind. In das IP-Qualifizierungsframework sind Experten integriert, die den IP-Anbieter beim Erstellen der erforderlichen Berichte durch (teil-)automatische Erzeugung unterstützen. Die Zertifizierungsphase nutzt dafür die Berichte, welche die externen Qualifizierungswerkzeuge generiert haben beziehungsweise die Dokumentation in den IPQ-Metadaten.

VSIA-Deliverable Document

Die VSIA hat vor QIP das VSIA-Deliverable Document veröffentlicht, in dem der Integrationsdatenumfang definiert ist, den ein IP-Anbieter an einen IP-Nutzer ausliefern muss, wenn das IP-Modul VSIA-konform sein soll [15]. Dieses Dokument besteht im Wesentlichen aus einem für jedes IP-Modul manuell auszufüllenden Anforderungskatalog. Dieser Katalog wurde in eine Datenbank abgebildet. Zusätzlich zu der VSIA-Spezifikation stellt die Datenbankversion Informationen aus den referenzierten VSIA-Dokumenten bereit und bietet die Möglichkeit Checklistenpunkten eine bestimmte (Dokumentations-)Datei zuzuweisen. Aufgrund dieser Information werden automatisch VSIA-konforme Vorlagen generiert, die dem Autor der Dokumentation die notwendigen und erwarteten Abschnitte vorgeben.

In Abbildung 19 auf Seite 95 ist eine Ansicht der Datenbankversion dargestellt. In den Zeilen eins bis drei werden das VSIA-Dokument und der Abschnitt, in dem die erwartete Auslieferung beschrieben ist und die Auslieferung selbst benannt. In Zeile vier wird die Notwendigkeit der Auslieferung durch eine von vier Möglichkeiten („obligatorisch“, „bedingt obligatorisch“, „empfohlen“ oder „bedingt empfohlen“) angegeben. Um die Arbeit des Ausfüllenden zu erleichtern werden die Informationen, auf die durch die Zeilen eins bis drei verwiesen wird, in Zeile sechs „Beschreibung“ (engl. description) wiedergegeben. In Zeile fünf kann eine das IP-Modul begleitende (Dokumentations-)Datei zugewiesen werden, in der der Sachverhalt dokumentiert werden soll. Die Datenbankversion erweitert das VSIA-Deliverable Document um die Zeilen fünf und sechs.

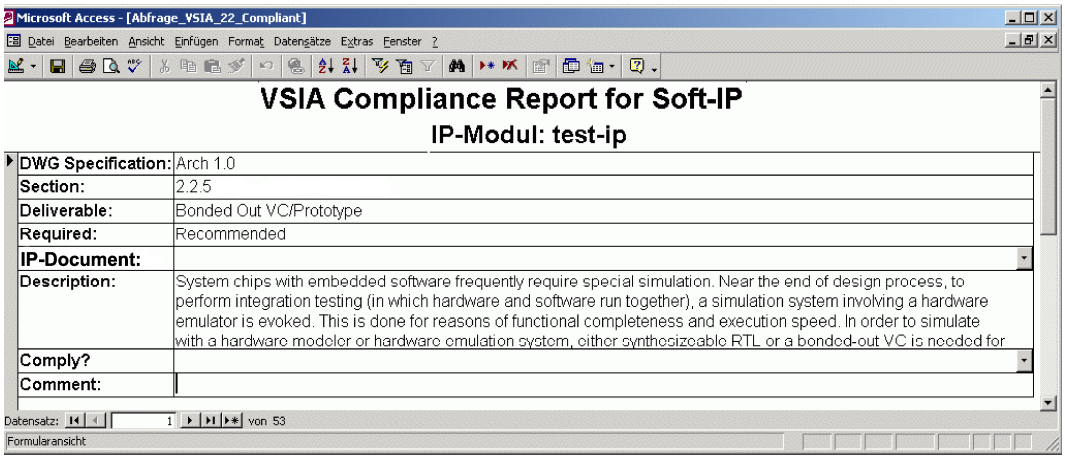


Abbildung 19: VSIA-Vollständigkeitsdatenbank

Der VSIA-Anforderungskatalog für das Zertifikat definiert Auslieferungen für vier verschiedene IP-Härtegrade („Soft“, „Firm“, „Hard“ und „Analog-Mixed-Signal“). Die Datenbankversion erleichtert das Erstellen des Zertifizierungsberichts, indem sie die für den IP-Härtegrad relevanten Kriterien filtert. Während der Zertifizierung muss der Ausfüllende lediglich die beiden Zeilen sieben „Übereinstimmung“ (engl. comply) und acht „Kommentar“ (engl. comment) ausfüllen. Nur diese beiden Zeilen enthalten spezifische Informationen des IP-Moduls, die in der Datenbank gespeichert, abgerufen und angepasst werden können. Daher ist es einfach das erzeugte Zertifikat an zukünftige Versionen des VSIA-Deliverable Documents anzupassen. Aus der Datenbank wird für jedes zertifizierte IP-Modul der entsprechende Bericht generiert.

Die Datenbankversion des VSIA-Deliverable Documents erleichtert das Erstellen des VSIA-Zertifizierungsberichts bereits erheblich. Die Datenbank wird jedoch bisher manuell gepflegt. Im folgenden Abschnitt wird für eine QIP-Zertifizierung beschrieben, wie der Aufwand zur Erstellung von Zertifizierungsberichten weiter reduziert werden kann.

Experte für QIP-Kriterium

Das folgende Beispiel zeigt, wie eine QIP-Kriterienprüfung implementiert werden kann. VSIAs QIP für digitales Soft-IP definiert in Zeile 189 als Qualitätskriterium, dass jede Quellcodedatei eine Copyrightinformation enthalten muss. Dafür wird dem Experten für die Überprüfung der Copyrightinformation ein String mit der firmenspezifischen Copyrightinformation übergeben. Findet der integrierte Parser diese Information im Quellcode, dokumentiert der Experte in den IPQ-Metadaten den logischen Teil vom Typ „QIP189“. Die entsprechende QIP-Quantifizierung, die in diesem Fall den Wert „yes“ oder „no“ vorsieht, wird im Attribut Namen eingetragen.

ID	logicalPartType	name	referencedUnits	xmlns:qip	xsi:type
id-94565664-50	QIP189	yes		http//www.fzi.de/ipq	qip:QIP189

Quellcode 13: Dokumentation eines QIP-Kriteriums

Als einfache Erweiterung dieses Experten kann der gesamte firmenspezifische Dateikopf auf Einhaltung der Firmenrichtlinien überprüft werden.

QIP-Zertifizierung

Für eine Zertifizierung nach VSIA-QIP muss ein Zertifizierungsbericht für das gesamte IP-Modul, nicht einzelner logischer Einheiten, erzeugt werden. Als Vorlage dient eine Microsoft®-Excel-Arbeitsmappe in die für jedes Kriterium eine Bewertung, nach der vorgesehen Quantifizierung eingetragen werden muss. Für jede IP-Art ist ein Arbeitsblatt beziehungsweise eine Kombination von Arbeitsblättern auszufüllen. Für die Erzeugung eines Teils der automatisch überprüfbaren QIP-Kriterien stehen Berichtgeneratoren bereit, welche die entsprechenden Informationen aus den IPQ-Metadaten extrahieren und daraus den QIP-Bericht erzeugen.

Beispielsweise wird im QIP-Berichtsgenerator für das digitale Soft-IP Kriterium in Zeile 189 zur Überprüfung der Copyrightinformation ausgewertet, ob für alle Quellcodedateien, also alle logischen Einheiten mit XML-Attribut isCompiled=“false“ und das Namensattribut des logischen Teils vom Typ „QIP189“ den Wert „yes“ enthält. In diesem Fall wird in die QIP-Bewertung des gesamten IP-Moduls der Wert „yes“ eingetragen, andernfalls „no“.

Bei Kriterienüberprüfungen mit anderer Quantifizierung, beispielsweise „always“, „often“ und „never“, werden die Werte „always“ und „never“ eingetragen, falls alle relevanten logischen Einheiten den entsprechenden Wert im logischen Teil eingetragen haben. Der Wert „often“ wird vergeben, falls für 70% der logischen Einheiten „always“ im logischen Teil dokumentiert ist, andernfalls „never“.

Für die automatische Erzeugung des Microsoft®-Excel-Formats gibt es verschiedene Lese- und Schreibschnittstellen aus der Programmiersprache Java. Das Paket JDBC-ODBC stellt eine Möglichkeit bereit, auf eine Excel-Arbeitsmappe mit SQL-Befehlen ähnlich wie auf eine relationale Datenbank zuzugreifen. Unterstützt wird mit der Select-Anweisung das Lesen des Arbeitsblatts. Allerdings wird die UPDATE-Anweisung und damit das Ausfüllen der QIP-Bewertung nur unzureichend unterstützt. Es können jedoch auch paketeigene API-Befehle zum Schreiben und Modifizieren verwendet werden [41]. Die Java Excel-API verspricht lesend und schreibend auf Excel-Arbeitsblätter zugreifen zu können [39], erweist sich aber in der Praxis als unbrauchbar. Das Apache POI.HSSF-Projekt hat in Kooperation mit dem OpenOffice-Projekt die brauchbarste API zum Lesen und Schreiben des Microsoft®-Excel-Formats entwickelt ([38], [40]). Allerdings unterstützt auch diese API nicht das komplette Excel-Format. Dadurch kommt es bei komplexen Arbeitsblättern, wie z. B. QIP, zu unbrauchbaren Zertifizierungsergebnissen. Makros und versteckte Berechnungen erzeugen schon häufig beim Einlesen der Arbeitsmappe Fehler.

Da für die Zertifizierung nach QIP lediglich das Ausfüllen der Spalten „Assessment“ und ggf. „Evaluator Comments“ relevant ist, ist eine Erzeugung dieser Spalten durch das IPQ-Framework ausreichend. Für zukünftige QIP-Versionen sollte überlegt werden die Metrik-Berechnung von den Eingabedaten zu trennen. Die Eingabedaten (Spalten „Assessment“ und ggf. „Evaluator Comments“) können dann automatisch und strukturiert erzeugt werden und einem Bewertungswerkzeug übergeben werden.

6.3.4.3 Experten für den IP-Export

Nach der erfolgreich abgeschlossenen Zertifizierung, kann ein IP-Modul ausgeliefert werden. Während der Exportphase werden die IPQ-Metadaten automatisch aus der IP-Struktur generiert und dem IPQ-Format hinzugefügt. Die automatische Generierung hängt von den firmenspezifischen Gegebenheiten ab, die in der Export-Expertenkonfigurationsdatei spezifiziert werden. Der Vorteil dieses Ansatzes ist, dass diese Konfigurationsdatei nur einmal geschrieben werden muss, und weitgehend unverändert in neuen Projekten wieder verwendet werden kann, solange sich die firmenspezifische IP-Struktur nicht ändert. Aufgrund der Tatsache, dass die firmenspezifischen Gegebenheiten außerhalb der Experten spezifiziert werden, kann eine schnelle Anpassung an Änderungen beziehungsweise andere Firmen erfolgen. Da ein IP-Anbieter immer in das gleiche IPQ-Integrationsdatenformat exportiert und ein IP-Nutzer immer das gleiche IPQ-Integrationsdatenformat importiert, ist durch die Automatisierung dieser Phasen eine nachhaltige Aufwandsreduzierung zu erwarten. Einen Ausschnitt einer Export-Expertenkonfigurationsdatei zeigt Quellcode 14. Dabei handelt es sich um eine erweiterte Regeldarstellung von Quellcode 6 auf Seite 84.

```
1 <Rules name="export" company="ip-anbieter" tool="IPQ-Framework">
2 …
3 <Rule name="RoleAssignment" logicalUnitType ="DCSH"
    pathExpression =".*/synopsys/run.scr"
    logicalRoleName ="Synthesis Setup Script">
    <!-Synopsys Design-Compiler top level setup script-->
  </Rule>
4 <Rule name="TypeRecognition" logicalUnitType ="ASCII"
    pathExpression =".*\.README">
    <!-README files have ASCII format-->
  </Rule>
5 <Rule name="Ignore"
    pathExpression =".*/CVS/.*">
    <!-Ignore CVS related stuff for delivery-->
  </Rule>
6 <Rule name="Ignore"
    pathExpression ="#.*#">
    <!-Ignore files generated by tool crash-->
  </Rule>
7 …
8 </Rules>
```

Quellcode 14: Beispielkonfigurationsdatei für Datei-/Verzeichnisstruktur

Diese Regeln konfigurieren die Export-Experten, auf die in den folgenden Abschnitten näher eingegangen wird. Da Bibliotheken und Zellen der IPQ-Metadaten im Dateisystem durch Verzeichnisse oberhalb der Hierarchieebene von physikalischen Einheiten dargestellt werden, ist für das Verständnis der Export–Experten die Erläuterung ab der Ebene der physikalischen Einheiten ausreichend. Denn physikalische Einheiten modellieren ebenfalls Verzeichnisse und zusätzlich Dateien.

Die generierten IPQ-Metadaten des Exportwerkzeugs werden zunächst in einem Java-Objektmodell gespeichert, das anschließend in ein XML-Do-

kument serialisiert wird. Dieses XML-Dokument wird mit den komprimierten IP-Integrationsdaten an den IP-Nutzer ausgeliefert.

Experte für physikalische Einheiten

Der Experte für die Erfassung der physikalischen Einheiten liefert die Grundlage für alle folgenden Export-Experten. Vom Wurzelverzeichnis der Zellen eines IP-Moduls ausgehend analysiert und erfasst er die IP-Struktur. Gefundene Verzeichnisse und Dateien, werden entsprechend ihrer Hierarchie in den IPQ-Metadaten als physikalische Einheiten vom Typ Verzeichnis oder Datei abgebildet.

Dabei werden die Regeln mit dem Attribut `name="Ignore"` aus der Experten-Konfigurationsdatei in Quellcode 6 auf Seite 84 beachtet. Pfade, die durch einen regulären Regelausdruck spezifiziert sind, werden bei der Erzeugung der IPQ-Metadaten ignoriert. Im Beispiel-Quellcode 14 auf Seite 97 werden Verzeichnisse, die von CVS erzeugt wurden, und durch einen Werkzeugabsturz generierte Backups beim Export ignoriert, also nicht ausgeliefert.

Experte für logische Einheiten

Jeder physikalischen Einheit ist in den IPQ-Metadaten genau eine logische Einheit untergeordnet. Logische Einheiten beschreiben Dateiformate, wie beispielsweise VHDL, Verilog, ASCII, C-Shell-Skript usw. Der Typ einer logischen Einheit im IPQ-Format entspricht also dem Dateityp. Der Dateityp ist meist durch die Dateinamenserweiterung gegeben. Der Experte für die Erzeugung der logischen Einheiten fügt die logische Einheit mit den notwendigen Attributen hinzu. Dabei werden die Regeln in Quellcode 14 auf Seite 97 mit dem Attributwert `name="TypeRecognition"` befolgt. Der Experte setzt in diesem Beispiel das Attribut `logicalUnitType` auf den Wert ASCII, falls es sich um eine README-Datei handelt. Eine README-Datei, wird an der Erfüllung des regulären Ausdrucks im Attribut `pathExpression` erkannt.

Experte für logische Rollen

Während des Exports werden anhand der Experten-Konfigurationsdatei Rollen vergeben, welche die Einordnung in die IP-Struktur unabhängig von firmenspezifischen Gegebenheiten beschreiben. Die Rollenzuweisung entspricht der semantischen Bedeutung der Datei oder des Verzeichnisses innerhalb der IP-Struktur. Dadurch wird der automatische Import in die IP-Struktur des IP-Nutzers ermöglicht.

Aufgrund der Regel (`<Rule name="RoleAssignment"…>`) in Quellcode 14 auf Seite 97 weist der Experte für die Generierung von logischen Rollen der logischen Einheit DCSH (Design-Compiler-Shellskript) die logischen Rollen: Synthese, Setup und Skript zu, falls ein Pfad dem regulären Ausdruck im XML-Attribut `pathExpression` entspricht.

Experte für logische Teile

Die Arbeitsweise eines „logischen Teile“-Analysators wird anhand eines Design-Compiler-Skriptausschnitts erklärt. Der Quellcode eines Design-Compiler-Skripts ist in Quellcode 15 gegeben. Für die besser Zuordnung der Beschreibung zu den Teilen in Quellcode 15, sind Beschreibung und Quellcode farblich gleich hinterlegt. Der Analysator durchsucht den Quellcode nach logischen Teilen die Abhängigkeiten definieren. Im Beispiel findet er netlist_path und define_design_lib. Die zugehörige Abhängigkeit (Pfadname) wird extrahiert. Enthaltene Variablen werden im Skriptkontext ausgewertet, bis der reine Pfadname vorliegt. Diese Information wird in den IPQ-Metadaten zur späteren Anpassung nach der Umstrukturierung beim Import gespeichert.

```
1   /* General variable settings */
2   SYN_SELECTOR = get_unix_variable("SYN_SELECTOR")
3   OS_SELECTOR = get_unix_variable("OS_SELECTOR")
4
5   /* General variable settings */
6   DEFAULT_WORK = "./work/" + SYN_SELECTOR + "/" + OS_SELECTOR
7
8   netlist_path = "./netlist/"
9
10  define_design_lib work -path DEFAULT_WORK
11  define_design_lib synopsys -path synopsys_root +
    _"/packages/synopsys/lib"
```

Quellcode 15: Ausschnitt aus einem Synopsys Design-Compiler-Skript

Die entsprechende Modellierung der logischen Teile als IPQ-Metadaten ist in Quellcode 16 dargestellt.

ID	logicalPartType	Name	xmlns:dcsh	xsi:type
id-9485555-60e	netlist_path	$IP_HOME/synthesis/netlist/	http//www.fzi.de/ipq	dcsh:netlist_path
id-9485555-61e	define_design_lib work	$IP_HOME/synthesis/work/synopsys/linux	http//www.fzi.de/ipq	dcsh:define_design_lib
id-9485555-62e	define_design_lib synopsys	$SYN_TOOL/packages/synopsys/lib	http//www.fzi.de/ipq	dcsh:define_design_lib

Quellcode 16: Darstellung logischer DCSH-Teile im IPQ-Format

Im Attribut `logicalPartType` wird das Schlüsselwort plus ggf. ein alleinstellendes Merkmal (z. B. der Name der Entwurfsbibliothek) gespeichert. Das Attribut `Name` speichert den zugehörigen absoluten Pfad. Dabei fällt auf, dass in dem aufgelösten Pfadnamen Variablen (`$IP_HOME` und `$SYN_TOOL`) enthalten sind. Dabei handelt es sich um firmenspezifische Werte, die erst beim IP-Nutzer bekannt werden. Der Experte für die logischen Teile analysiert zunächst einen Pfadnamen, bevor er ihn als Attributwert übernimmt. Enthält der Pfadname einen firmenspezifischen Anteil, wir der durch die entsprechende Variable ersetzt (z. B. `$SYN_TOOL` für das Installationsverzeichnis des Synthesewerkzeugs oder `$SIM_TOOL` für das Installationsverzeichnis des Simulators). Diese speziellen Variablen sind im IPQ-Framework registriert und werden beim Import anhand der Werkzeugkonfigurationsdatei (vgl. Abschnitt „Experte zur Auswertung der Entwurfsablaufdokumentation“ auf Seite 105) durch den firmenspezifischen Wert ersetzt.

Experte zur Dokumentation des Entwurfablaufs

Firmenspezifische Informationen über das verwendete beziehungsweise das generierende Werkzeug (Werkzeugname und -version) stellt der IP-Anbieter in den einführenden Kommentaren der Quellcodedateien beziehungsweise für binäre oder generierte Daten in der Framework-Konfigurationsdatei bereit. Diese Informationen werden vom Werkzeug-Experten genutzt, um die logischen Einheiten im IPQ-Integrationsdatenformat mit Werkzeuginformationen zu ergänzen. Auf diese Weise wird der Entwurfsablauf unabhängig von der firmenspezifischen Dokumentationsweise an den IP-Nutzer übertragen. Dort werden sie für die Integration des IP-Moduls in den Werkzeugablauf genutzt.

Quellcode 18 auf Seite 101 zeigt eine logische Einheit als Ausschnitt aus dem XML-Dokument des IPQ-Formats. Zu sehen ist das XML-Element `ipq:LogicalUnit`. Die logische Einheit ist in der physikalischen Einheit *ip_ci_rtl.vhd* enthalten und modelliert eine VHDL-`Architecture`. Erzeugt wurde diese Einheit mit dem Editor XEmacs in der Version 21.4.15. Für eine Netzliste wäre an dieser Stelle das Synthesewerkzeug dokumentiert. Die physikalische Einheit ist aus Gründen der Übersichtlichkeit nicht wiedergegeben. Quellcode 17 zeigt die dazugehörigen logischen Teile. Zu sehen ist, dass die logische Einheit zwei Bibliotheken benutzt und eine `Architecture` enthält. Sowohl die `Architecture` als auch die Bibliothek `work` verweisen auf unterschiedliche logische Einheiten (unterschiedliche IDs unter `referencedUnits`) außerhalb der eigenen physikalischen Einheit.

Die verfügbaren XML-Attribute physikalischer Einheiten wurden in Abschnitt 6.1.4, erklärt, die der logischen Einheiten in Abschnitt 6.1.5, die der logischen Rollen in Abschnitt 6.1.6 und die der logischen Teile in Abschnitt 6.1.7.

ID	logicalPart-Type	Name	referenced-Units	xmlns:vhdl	xsi:type
id-9489155-40e	VHDLUse	ieee.std_logic_1164.all		http//www.fzi.de/ipq	vhdl:VHDLUseType
id-9489155-37e	VHDLArchitecture	rtl	id-9489155-9e	http//www.fzi.de/ipq	vhdl:VHDLArchitectureType
id-9489155-39e	VHDLLibrary	work		http//www.fzi.de/ipq	vhdl:VHDLLibraryType
id-9489155-38e	VHDLLibrary	ieee		http//www.fzi.de/ipq	vhdl:VHDLLibraryType
id-9489155-41e	VHDLUse	ieee.std_logic_arith.all		http//www.fzi.de/ipq	vhdl:VHDLUseType
id-9489155-42e	VHDLUse	work.ip_pkg.all	id-9489155-0e	http//www.fzi.de/ipq	vhdl:VHDLUseType

Quellcode 17: Darstellung logischer Teile im IPQ-Format

ipq:LogicalUnit						
ID	id-9489155-2e					
author	ip-anbieter					
description	Submodule ip_ci (Command Interface).					
isCompiled	False					
isExecutable	False					
isTop	False					
logicalUnitType	VHDL					
name	ip_ci_rtl.vhd					
toolname	XEmacs					
toolversion	21.4.15					
version	1.1.1.1					
xmlns:vhdl	http//www.fzi.de/ipq					
xsi:type	vhdl:VHDLUnitType					
ipq:LogicalRole						
name	Design					
ipq:LogicalPart (6)						
ID	**logicalPartType**	**name**	**referencedUnits**	**xmlns:vhdl**	**xsi:type**	

Quellcode 18: Darstellung einer logischen Einheit im IPQ-Format

6.3.4.4 Experten für die Auslieferungszusammenstellung

Während der Exportphase werden die zur Auslieferung bestimmten Integrationsdaten automatisch zusammengestellt. Dafür steuern mehrere Experten, die im Folgenden beschrieben werden ihr Wissen bei. Die Vorgehensweise ist in Abbildung 20 auf Seite 102 dargestellt. Die folgenden Expertenbeschreibungen beziehen sich auf diese Abbildung.

Versionsexperte

Zunächst wird für den Export die korrekte Auslieferversion anhand des Kundennamens und des IP-Modulnamens bestimmt. Aufgrund von Verbesserungen und Korrekturen am IP-Modul können mehrere Releases eines IP-Moduls im CVS-Speicher enthalten sein. Für eine Auslieferung wird daher immer das aktuelle Release verwendet. Ist für den IP-Modulnamen eine kundenspezifische Markierung vorhanden, wird die neuste kundenspezifisch qualifizierte Version verwendet. Anhand der CVS-Markierung wird das gesamte IP-Modul in den Arbeitsbereich des Auslieferers kopiert (`cvs checkout -r <markierung> <ip-modul-name>`).

Vollständigkeitsexperte

Da das IP-Modul in der Regel über mehr Daten (Dateien) verfügt, als dem Kunden ausgeliefert werden, wird dem Auslieferer eine dem Vertragsmodell (Tabelle 4 auf Seite 60) entsprechende Vorauswahl der Integrationsdaten durch Markierung in einer Dateiliste bereitgestellt. Über eine grafische Benutzeroberfläche (engl. Graphical User Interface, kurz GUI) kann das Vertragsmodell gewählt werden. Die möglichen Vertragsmodelle eines

IP-Anbieters sind in einer Experten-Konfigurationsdatei definiert, die ebenfalls im Kommandozeilenmodus an das Werkzeug übergeben werden kann. Anhand des gewählten Vertragsmodells wird die Vollständigkeit des IP-Moduls sichergestellt.

Für kundenspezifische Anpassungen kann die Dateiauswahl manuell nachbearbeitet werden oder direkt übernommen werden. Ist die manuelle Nachbearbeitung abgeschlossen, wird die Konsistenz der Auslieferung automatisch geprüft. Details zur automatischen Konsistenzprüfung sind in Abschnitt 4.3.5 beschrieben. Die endgültige Dateiauswahl wird zu Reproduktionszwecken, mittels einer für diese Auslieferung eindeutigen Markierung, im CVS-System festgehalten (`cvs tag <markierung> <ausliefer-dir>/<ip-modul-name>`). Über diese Markierung kann genau zurückverfolgt werden an welchen Kunden welche Releaseversion des IP-Moduls und mit welchem Dateiumfang ausgeliefert wurde.

Nachdem der Auslieferumfang festgelegt ist, werden mit den Experten für den IP-Export aus Abschnitt 6.3.4.3 die IPQ-Metadaten erzeugt.

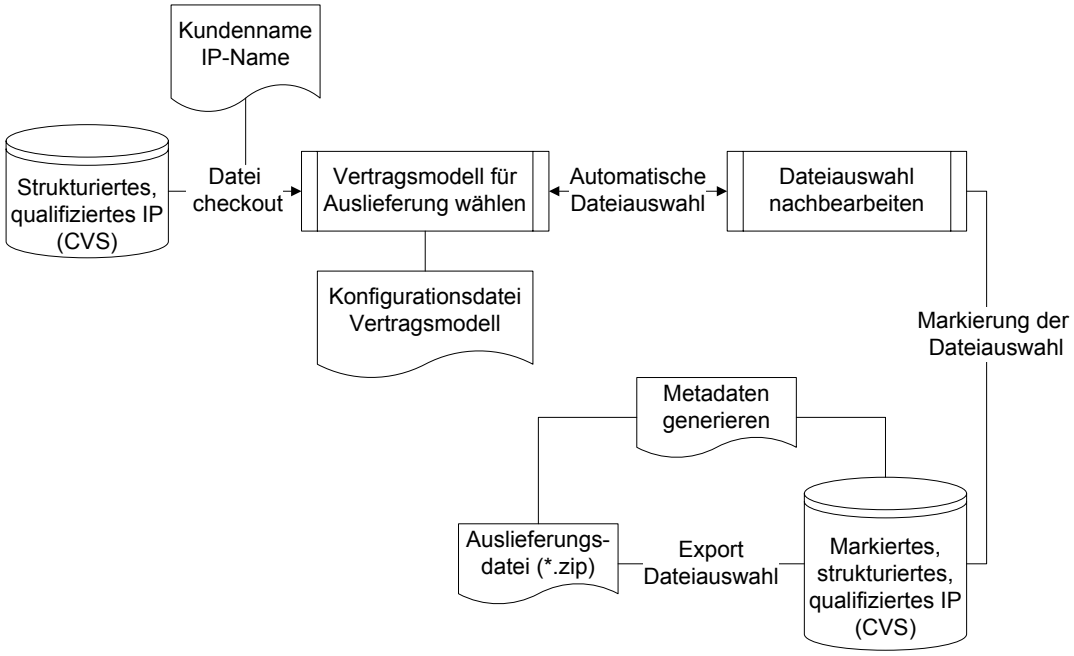


Abbildung 20: IP-Exportwerkzeug

Dokumentations-, Überprüfungs- und Verschlüsselungsexperte

Die in Abbildung 20 genanten Zusatzinformationen, wie grafischer Verzeichnisbaum als Dokumentation für den IP-Nutzer, Versionsinformationen und Prüfsummen aller Dateien, werden der komprimierten Auslieferung hinzugefügt.

Die IP-Struktur wird in strukturiertem XHTML dokumentiert, der mithilfe eines XSLT-Skripts aus den IPQ-Metadaten generiert wird. Durch das XSLT-Skript werden Bibliotheken, Zellen und physikalische Einheiten in einer hierarchischen XHTML-Liste ausgegeben und mit den Verzeichnissen beziehungsweise den Dateien verlinkt. Quellcode 19 auf Seite 103 zeigt einen Auszug aus dem entsprechenden XSLT-Skript. In Zeile 1 wer-

den die benötigten Namensräume definiert. In Zeile 2 wird HTML als Ausgabeformat gewählt, die Zeichencodierung auf UTF-8 und der Dokumententyp auf XHTML festgelegt. Nach den HTML-Header-Informationen in den Zeilen 5-7, folgt ab Zeile 10 der Aufbau der Liste, ausgehend von den `Library`-Elementen (XPath-Ausdruck: „`//ipq:Library`“) im XML-Metadatendokument. Dafür wird das `Template` verwendet, das `ipq-Library`-Elemente verarbeitet. Dieses erzeugt die XHTML-Listenelemente für IPQ-Bibliotheken und erzeugt XHTML-Links auf die entsprechenden Verzeichnisse der IP-Struktur. Dieses Vorgehen wiederholt sich für die Zellen und physikalischen Einheiten, also die Verzeichnisse und Dateien der IP-Struktur.

```
1  <xsl:stylesheet version="1.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:ipq="http://www.fzi.de/IPQ">
2  <xsl:output method="html" version="1.1" encoding="UTF-8"
   doctype-public="-//W3C//DTD XHTML 1.1 Strict//EN" doctype-
   system= "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd"/>
3  <xsl:template match="/">
4     <html>
5       <head>
6        <title>IP module structure documentation</title>
7       </head>
8       <body>
9        <h1>IP module structure documentation</h1>
10       <ul>
11        <xsl:apply-templates select="//ipq:Library"/>
12       </ul>
13      </body>
14    </html>
15 </xsl:template>
16 <xsl:template match="ipq:Library">
17    <li>
18      <a target="sourcecode">
19       <xsl:attribute name="href"><xsl:text>../iist/
         </xsl:text><xsl:value-of select="@name"/>
         </xsl:attribute>
20      <xsl:value-of select="@name"/>
21      </a>
22      <ul>
23       <xsl:apply-templates select="ipq:Cell"/>
24      </ul>
25    </li>
26 </xsl:template>
27 …
```

Quellcode 19: XSLT-Skript zur Generierung der XHTML-IP-Struktur (Auszug)

Der IP-Nutzer erhält durch die Prüfsummen, die für jede Datei generiert werden, die Möglichkeit das Auslieferungspaket auf Übertragungsfehler zu kontrollieren. Dafür wird die MD5-Hash-Funktion verwendet. In Quellcode 20 ist das Vorgehen im C-Shell-Quellcode dokumentiert.

```
1  find $IP_NAME -exec md5sum -b {} \; > chechsums.md5
```

Quellcode 20: MD5-Prüfsummenberechnung

Zeile 2 übergibt der Prüfsummenberechnung die einzelnen Dateien des Auslieferungsumfangs und speichert das Ergebnis in der Datei *checksum.md5*.

Schließlich wird der Auslieferungsumfang zusammen mit der Dokumentation der IP-Struktur, den Prüfsummen und den IPQ-Metadaten archiviert und komprimiert (Quellcode 21).

```
1  gtar cvfz $IP_NAME.tar.gz $IP_NAME
```

Quellcode 21: Archivierung und Komprimierung für die Übermittlung

Da für die Verschlüsselung nicht vorausgesetzt werden kann, dass der öffentliche Schlüssel des IP-Nutzers bekannt ist, wird zunächst automatisch ein Schlüsselpaar generiert.

```
1  gpg --gen-key --batch default-key-gen.txt
```

Quellcode 22: Schlüsselpaargenerierung

Für die Generierung im Batchbetrieb wird die folgende Datei benötigt. Die Variablen im Quellcode werden während der Exportphase automatisch ersetzt und beim IP-Anbieter dokumentiert.

```
1  %echo Generating an IPQ key
2  Key-Type: DAS
3  Key-Length: 1024
4  Subkey-Type: ELG-E
5  Subkey-Length: 1024
6  Name-Real: $IP_USER_NAME
7  Name-Comment: $IP_NAME Version: $VERSION
8  Name-Email: $IP_USER_EMAIL
9  Expire-Date: 0
10 Passphrase: $PASSPHRASE
11 %commit
12 %echo done!
```

Quellcode 23: default-key-gen.txt

Mit dem öffentlichen Schlüssel wird das Auslieferungspaket vor der Übermittlung verschlüsselt.

```
1  gpg --encrypt --recipient $IP_USER_NAME
    $IP_NAME.tar.gz
```

Quellcode 24: Verschlüsselung des Auslieferungspakets

Der private Schlüssel mit Kurzanleitung zum Entschlüsseln wird dem IP-Nutzer auf einem separaten Weg übermittelt.

6.3.4.5 Experten für den IP-Import

Dem IP-Nutzer stehen im IPQ-Framework Experten für den automatischen Import in den firmenspezifischen Entwurfsablauf zur Verfügung. Dazu zählt ein Experte zur Auswertung der Entwurfsablaufdokumentation, die während der Exportphase erzeugt wurde. Der IP-Nutzer kann sich während einer Eingangskontrolle von der Qualität des IP-Moduls überzeugen, bevor er es umstrukturiert und die Daten an die firmeninternen Änderungen anpasst. Die implementierten Experten werden im Folgenden vorgestellt.

Experte zur Auswertung der Entwurfsablaufdokumentation

Aus den beim IP-Anbieter generierten Entwurfsablaufinformationen kann der IP-Nutzer beim Import eine Liste aller für das IP-Modul verwendeten Werkzeuge erzeugen. Diese Liste konfiguriert einen Experten für die Werkzeugintegration. Dieser erfragt beim IP-Nutzer den Pfad zu der firmenspezifischen Werkzeug-Installationsdatei und dem firmenspezifischen Werkzeug-Installationsverzeichnis. Auf diese Weise wird der IP-Nutzer bereits beim Import auf nicht verfügbare Werkzeuge aufmerksam gemacht. Diese Werkzeuge können dann durch Werkzeuge gleicher Funktionalität ersetzt werden. In diesem Fall ist jedoch zu erwarten, dass beim IP-Nutzer nicht die gleichen Qualifizierungsergebnisse erzielt werden, wie sie der IP-Anbieter mit einem anderen Werkzeug erzielt hat.

Die Zuordnung von Werkzeugnamen und -versionen zu Installationsdateien wird in der Werkzeugkonfigurationsdatei im XML-Format gespeichert. Aufgrund dieser Zuordnung können ausführbare Skripte für die Synthese, die Simulation oder *Makefile*s externer Qualifizierungsexperten und Pfadnamen mit firmenspezifischen Werten (z. B. `$SIM_TOOL`, `$IP_HOME`) an den IP-Nutzer Entwurfsablauf angepasst werden.

Experten für die Eingangskontrolle

Während einer Eingangskontrolle kann der IP-Nutzer Qualitätsmessungen durchführen und sich auf diese Weise von der Qualität des IP-Moduls überzeugen. Für diese Qualitätsmessungen können die gleichen Experten verwendet werden, die in der entwurfsbegleitenden beziehungsweise abschließenden Qualifizierung verwendet werden. Daher wird an dieser Stelle auf die Beschreibung in Abschnitt 6.3.4.1 verwiesen.

Experte für die Umstrukturierung

Der Experte für die Umstrukturierung unterstützt den IP-Nutzer, wenn das IP-Modul nach einer erfolgreichen Eingangskontrolle automatisch an die firmeninterne IP-Struktur angepasst werden soll. Dafür erzeugt der Experte aus den IPQ-Metadaten das interne Objektmodell des IPQ-Frameworks. Dieses Objektmodell traversiert der Experte und entscheidet anhand der Regeln in der Expertenkonfigurationsdatei (Quellcode 25) in welches Zielverzeichnis eine Datei verschoben wird. Da die Zielverzeichnisse automatisch mit dem Verschieben der Dateien erzeugt werden, müssen nur physikalische Einheiten vom Typ Datei auf Übereinstimmung mit den Regelkriterien geprüft werden. Eine Regel wird auf eine Datei angewendet, wenn sie die Regelbewertung aus Abschnitt 5.5.4.2 gewinnt.

```
1  <Rules name="import" company="ip-nutzer" tool="IPQ-Framework">
2  …
3     <Rule name="import" logicalUnitType="ASCII"
        logicalRoleName="Lint LOG"
        targetDirectory="$IP_HOME/lint/log"/>
4  …
5  </Rules>
```

Quellcode 25: Import-Expertenkonfiguration

Wird eine Datei von keiner Regel abgedeckt und kann aufgrund dessen nicht in die firmeninterne IP-Struktur eingefügt werden, erzeugt der Experte einer Fehlermeldung in der Framework-Logdatei. Ist die Verschiebung einer Datei von mehreren Regeln abgedeckt wird in der Framework-Logdatei eine Warnung erzeugt. Kann eine Datei aufgrund einer eindeutigen Regelabdeckung verschoben werden, wird eine Information der Framework-Logdatei erzeugt (Quellcode 26).

```
1  if (applyRules.count(physicalUnit) == 1) {
2     Logger.infoLog("File has been unambiguously moved!");
3  else if (applyRules.count(physicalUnit) > 1) {
4     Logger.warnLog("File has been moved but more than one rule
        applied!");
5  }
6  else {
7     Logger.errorLog("File has not been moved, because no rule
        did apply!");
8  }
```

Quellcode 26: Erzeugung der Framework-Log Events

Warnungen sollten vom IP-Nutzer kontrolliert und Fehlermeldungen manuell behandelt werden.

Experten für Formatanpassungen

Bei allen Formaten, die Definitionen und Verwendung von (System)Variablen ermöglichen, können die Variablenwerte Verweise auf LogicalUnits enthalten. Pfad- und Dateinamen werden häufig am Anfang eines Skripts oder in einem externen Setup-Skript als Variable definiert und im weiteren Verlauf die Variable benutzt. Das IP-Qualifizierungsframework bestimmt die Abhängigkeiten statisch. Wird beispielsweise ein Syntheseskript analysiert, ohne die Installationsskripte für die Erzeugung der Umgebung ausgeführt zu haben, sind die Werte von Umgebungsvariablen, nicht ohne weiters zu bestimmen. Nur über die physikalische Analyse des gesamten Integrationsdatenumfangs und dem zusätzlichen Wissen, welche enthaltenen Dateiformate Variablen definieren können, ist es möglich, den Variablenwert statisch zu bestimmen. Daher müssen zunächst Konstrukte aufgedeckt werden, dann Variablen identifizieren und diese dann aufgelöst werden, bis der Variablenwert feststeht. Dafür werden reguläre Aus-

drücke verwendet. Ein Beispiel für die Erkennung wurde bereits im Abschnitt „Experte für logische Teile“ auf Seite 99 geliefert.

Nach der erfolgreichen Erkennung des Konstrukts kann anhand der logischen Teile in den IPQ-Metadaten, die beim IP-Anbieter erzeugt wurden, die Daten angepasst werden. Aus Quellcode 15 auf Seite 99 werden mit den IPQ-Metadaten aus Quellcode 16 auf Seite 99, den bei der Umstrukturierung erzeugten Informationen in Tabelle 14 und der firmenspezifischen Werkzeugkonfigurationsdatei in Tabelle 15 die angepassten Daten in Quellcode 27 generiert.

Quellverzeichnis	Zielverzeichnis
$IP_HOME/synthesis/netlist/	*/user/modules/ip/syn/net*
$IP_HOME/synthesis/work/synopsys/linux	*/user/modules/ip/syn/work*
$SYN_TOOL/packages/synopsys/lib	*/tools/synopsys-2002.05/packages/synopsys/lib*

Tabelle 14: Umstrukturierungsinformationen

registrierte Variable	Werkzeug		Installationsdatei	Installationsverzeichnis
	Name	Version		
$SYN_TOOL	synopsys	2002.05	*/tools/synopsys-2002.05/setup.csh*	*/tools/synopsys-2002.05*

Tabelle 15: Werkzeugkonfigurationsdatei

```
12 /* General variable settings */
13 SYN_SELECTOR = get_unix_variable("SYN_SELECTOR")
14 OS_SELECTOR = get_unix_variable("OS_SELECTOR")
15
16 /* General variable settings */
17 DEFAULT_WORK = "./work/" + SYN_SELECTOR + "/" + OS_SELECTOR
18
19 netlist_path = /user/modules/ip/syn/net
20
21 define_design_lib work -path /user/modules/ip/syn/work
22 define_design_lib synopsys -path
          /tools/synopsys-2002.05/packages/synopsys/lib
```

Quellcode 27: Angepasstes Synopsys Design-Compiler-Skript

7 Ergebnisse

Ergebnisse wurden in verschiedenen Bereichen erzielt, die im Folgenden beschrieben werden. Abschnitt 7.1 erläutert die Ergebnisse, die in Bezug auf die Standardisierung von Qualitätskriterien und deren -bewertung erreicht werden konnten. Die Leistungsfähigkeit der Qualifizierungsmethode wird in Abschnitt 7.2 dargestellt. Abschnitt 7.3 beschreibt die Aufwandsreduzierung während der IP-Auslieferungsphasen.

7.1 Standardisierung

Die Standardisierung von IP-Qualitätskriterien und deren Bewertung hat der Autor beeinflusst, indem er bis zum Erscheinen der offiziellen QIP-Version 1.11 aktiv in der Quality Development Working Group der VSIA mitgearbeitet hat. Ergebnisse der automatischen Qualifizierung, Kriterienvorschläge und das Konzept der nutzerspezifischen Bewertung sind dadurch in den QIP-Standard eingeflossen.

7.2 IP-Qualifizierung

Zur Messung der Leistungsfähigkeit der entwickelten Qualifizierungsmethode, wurde das IP-Qualifizierungsframework in einen realen Entwurfsablauf integriert. Das untersuchte IP-Modul hat eine Gesamtdateigröße von 13 MB in 260 Dateien. Der Flächenbedarf des synthetisierten IP-Moduls entspricht ca. 3.800 Gatteräquivalenten. Die Ziele dieser Arbeit, die Verbesserung der Wiederverwendbarkeit, die Reduzierung des Integrationsrisikos und die Verringerung des Qualifizierungsaufwands durch automatisierte Qualitätsmessungen, konnte erreicht werden, was die folgenden Ergebnisse bestätigen. Zudem konnte nachgewiesen werden, dass die Wiederverwendung von IP-Modulen, trotz des erhöhten Entwicklungs- und Qualifizierungsaufwands sowohl für den IP-Nutzer, als auch für den IP-Anbieter, profitabel ist.

Das entwickelte IP-Qualifizierungsframework kann an unterschiedliche Entwurfsabläufe angepasst werden. Für diese Untersuchung wurden kommerzielle Werkzeuge anbieterspezifisch angepasst und in Kooperation mit einem IP-Anbieter in seinen Entwurfsablauf integriert. In den Fällen, in denen keine Werkzeuge oder nur unzureichende kommerzielle Werkzeuge verfügbar waren, wurden eigene Werkzeuge entwickelt. Die Werkzeuge, inklusive aller benötigten Parameter, werden automatisch ausgeführt. Die Qualitätsanalysen werden durch generierte Makefiles gesteuert. Lediglich einige technologie-, entwurfs- und nutzerspezifischen Anforderungen müssen manuell bearbeitet werden. In den meisten Fällen werden die Qualitätsanalysen ohne Eingriffe des Benutzers durchgeführt. In Tabelle 16 auf Seite 110 werden die erreichten Ziele den entsprechenden Qualifizierungsmethoden zugeordnet.

7.2.1 IP-Qualifizierungsphasen

Bereits durch die Implementierung des Quellcodestandards können 68 % der anbietereigenen Codierrichtlinien mit Werkzeugunterstützung geprüft werden. Die Codierrichtlinien sind aus Quellcodekriterien von RMM und QIP abgeleitet. Es ist wichtig zu erwähnen, dass die Arbeiten am Qualifizierungsframework auf den QIP-Vorgängern RMM [4] und OpenMORE [7] und dem VSIA-Deliverable Document [15] basieren. Da diese Qualitätskriterien in QIP eingeflossen sind, sollte eine Übertragung der Ergebnisse auf die QIP-Qualifizierung erlaubt sein.

Änderungen an Zertifizierungsbestimmungen können aufgrund der datenbankgestützten Vorlagengenerierung sehr viel einfacher in den Qualifizierungsablauf integriert werden. Durch (teil-)automatisch generierte Dokumentation, konnte die Zeit, die zum manuellen Erstellen der Zertifizierungsberichte erforderlich ist, ebenfalls reduziert werden.

Qualitätskriterien	Erreichtes Ziel
Quellcodequalität	• Mit einem firmenspezifisch konfigurierten RuleChecker (Linter) können 68 % der relevanten Richtlinien automatisch überprüft werden.
Verifikationsqualität	• Verbesserung der Verifikationsabdeckung durch die Integration eines Code-Coverage-Werkzeugs in den Qualifizierungsablauf.
Entwurfsqualität	• Überprüfung der fehlerfreien Synthetisierbarkeit • Einhaltung des spezifizierten Zeitverhaltens
Dokumentationsqualität	• Datenbankgestützte Generierung standardkonformer Dokumentationsvorlagen
Versions- und Konfigurationsmanagement	• Integration des Versionskontrollsystems in die Exportphase
Vollständigkeits-, Integritäts- und Konsistenzprüfungen	• Anhand von Vertragsmodellen wird die Vollständigkeit gewährleistet. • Integritäts- und Konsistenzprüfungen werden durch Modellierung mit dem IPQ-Format ermöglicht.

Tabelle 16: Erreichte Qualifizierungsziele

7.2.2 Fallstudie

In einer Serie von Projekten eines IP-Anbieters wurden die Verbesserungen des Wiederverwendungsprozesses gemessen, die durch das Anwenden der beschriebenen Techniken ermöglicht werden. In diesen Projekten wurde ein peripheres Schnittstellen-IP-Modul integriert. Alle Daten wurden der Projektaufwandsdatenbank entnommen, in der alle Entwickler ihre aufgewendeten Arbeitsstunden eintragen.

7.2.2.1 Vergleichsprozess

Im allgemeinen IP-Wiederverwendungsprozess, bevor die beschriebene IP-Qualifizierungsmethode verwendet wurde, wurde ein IP-Modul, welches

in einem früheren Auftragsprojekt entwickelt wurde, dem Auftragsprojekt entnommen, in ein Wiederverwendungsprojekt importiert und integriert (Abbildung 21).

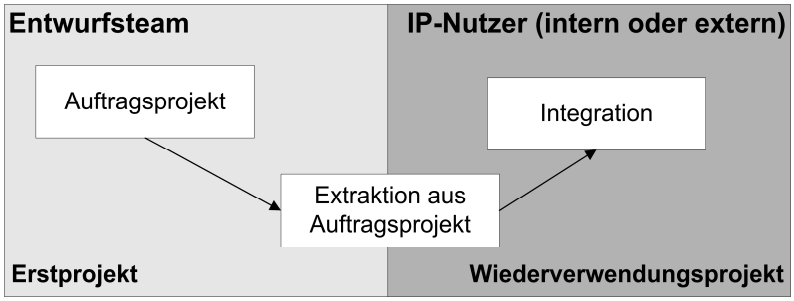


Abbildung 21: Unausgereifter Wiederverwendungsprozess

Tabelle 17 Spalte 1 zeigt den Nachteil dieses Wiederverwendungsprozesses. Der Aufwand für die Entnahme und Integration des IP-Moduls überschreitet den ursprünglichen Entwicklungsaufwand beträchtlich. Es wäre günstiger gewesen, das IP-Modul von Grund auf neu zu entwickeln. Sogar bei mehrfacher Wiederverwendung ist dieser Ansatz unwirtschaftlich, da das IP-Modul wiederholt entnommen und weiterbearbeitet werden muss.

Spaltennummer	1	2	3	4
Szenario	**Unausgereifte Methode: ab der ersten Wiederverwendung [Ph]**	**IPQ: erste Wiederverwendung [Ph]**	**IPQ: ab der zweiten Wiederverwendung [Ph]**	**IPQ: vollständig umgesetzt [Ph]**
Erstentwicklung	2340	2340		2500
Steigerung der Wiederverwendbarkeit		2300		1000
Qualifizierung		45	10	10
Auslieferung und Support		350	16	16
Integration	2450	120	100	100
Summe	4790	5155	126	3626

Tabelle 17: Aufwand für verschiedene Wiederverwendungsprozesse (Ph = Personenstunden)

7.2.2.2 Verbesserter Wiederverwendungsprozess

Im IPQ-Projekt [26] wurde seit 2002 vom Autor zusammen mit dem IP-Anbieter ein Wiederverwendungsprozess entwickelt und zunehmend beim IP-Anbieter eingesetzt (Abbildung 22 und Spalten 2-4 in Tabelle 17).

Folgt man diesem Wiederverwendungsprozess, wird nach einer positiven Bewertung des Marktpotentials ein IP-Modul weiterentwickelt, bis es definierten Qualitätskriterien entspricht (Abschnitt 2.5.1). Anhand der Techniken für Quellcode-, Validierungs- und Entwurfsqualitätsmessungen aus Abschnitt 4.3 wird das IP-Modul durch eine gesonderte Abteilung qualifi-

ziert. Das führt möglicherweise zu „kurzen“ Entwurfsiterationen, falls Qualitätsmängel gemessen werden. Ist das IP-Modul ausreichend qualifiziert, wird es in die Wiederverwendungsdatenbank eingepflegt. Aus der Wiederverwendungsdatenbank wird das IP-Modul an IP-Nutzer ausgeliefert.

Tauchen Qualitätsmängel erst beim IP-Nutzer auf, bietet die IP-Qualifizierungsabteilung ersten Support an und bearbeitet Kundenrückmeldungen und -ansprüche. Ist das nicht ausreichend, wird eine Entwurfsiteration eingeleitet. Da diese Iteration den IP-Nutzer einbindet, wird sie „lange“ Iteration genannt.

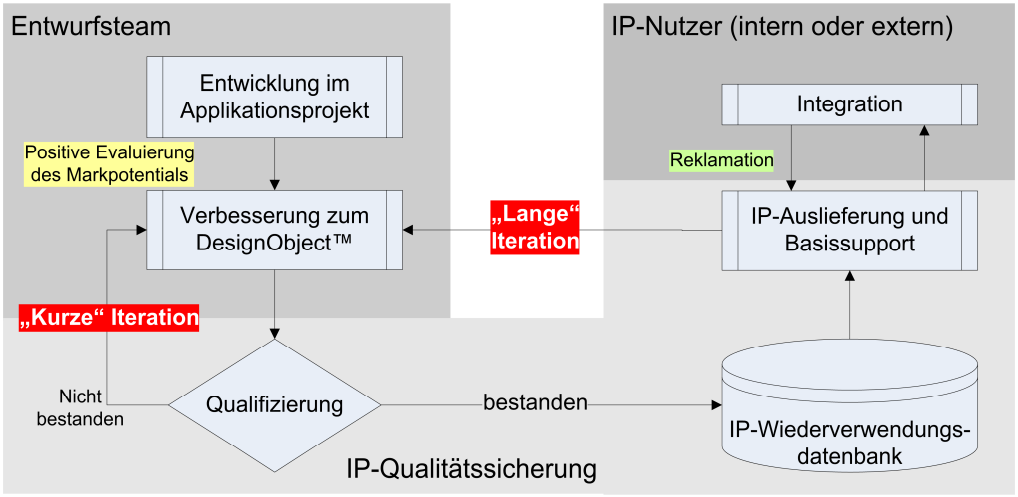


Abbildung 22: IP-Anbieter Wiederverwendungsprozess

Die Aufwandsdaten für die ursprüngliche Entwicklung, um den Entwurf entsprechend der Qualitätskriterien wiederverwendbar zu machen, zeigt Tabelle 17 Spalte 2 auf Seite 111. Offensichtlich ist ein erheblicher Aufwand nötig, um das IP-Modul wiederverwendbar zu machen und die Gesamtpersonenleistung für eine Wiederverwendung übersteigt sogar etwas den unreifen Wiederverwendungsprozess vor IPQ. Der Integrationsaufwand wurde jedoch um 95 % reduziert. Daher zieht der IP-Nutzer einen großen Vorteil aus der Anwendung der IP-Qualifizierungsmethoden beim IP-Anbieter.

Im Weiteren wird anhand der in Tabelle 17 auf Seite 111 gesammelten Daten über den Wiederverwendungsprozess in Abbildung 22 gezeigt, dass dieser Prozess auch für den IP-Anbieter wirtschaftlich ist. Dafür sind mehrere Wiederverwendungen nötig. Beim IP-Anbieter wurde das periphere IP-Modul ein zweites Mal wieder verwendet. Es wurde eine erneute Qualifizierung notwendig, um die Verbesserungen aus der Supportphase der ersten Wiederverwendung zu berücksichtigen, sowie eine erneute Auslieferung zum Zielprojekt und eine erneute Integration. Der Aufwand (Tabelle 17 Spalte 3 auf Seite 111) war gering, da keine erneute Entnahme aus dem Ursprungsprojekt, keine Entwurfsverbesserungen und kein Aufwand zur Schaffung der Wiederverwendbarkeit notwendig waren. Eine zweite Wiederverwendung, mit dem früheren, unausgereiften Wiederverwendungsprozess, hätte erneut 2.000 Personenstunden benötigt.

Der neue Wiederverwendungsprozess erlaubt es einem IP-Anbieter, wie folgt zu arbeiten, um das eigene Risiko zu reduzieren:

- Entwicklung eines IP-Moduls in einem Auftragsprojekt. Zu diesem Zeitpunkt wird nicht in die Wiederverwendbarkeit des Moduls investiert. Es entstehen keine Mehrkosten.
- Für einen potentiellen IP-Nutzer wird das IP-Modul durch Verbesserungen und Qualifizierung zu einem qualitativ hochwertigen IP-Modul aufbereitet. In diesem Stadium ist es unwahrscheinlich, dass der IP-Anbieter Gewinn macht, da der Erlös durch die zusätzlichen Entwicklungskosten aufgebraucht wird. Jedoch wird der Firmenwert durch das zur Wiederverwendung bereitstehende IP-Modul erhöht.
- Ab der zweiten Wiederverwendung des IP-Moduls kann das IP-Geschäft wirtschaftlich betrieben werden.

Es bleibt anzumerken, dass in dieser Untersuchung die ursprüngliche Entwicklung ohne einen Gedanken an Wiederverwendung durchgeführt wurde. Als die IPQ-Ergebnisse beim IP-Anbieter bekannt wurden, konnten die Entwickler überzeugt werden, die Wiederverwendbarkeit von Anfang an zu berücksichtigen. Automatisierte Qualifizierungswerkzeuge, die auf den in Abschnitt 4.3 vorgestellten Methoden basieren, sind für alle Entwickler verfügbar gemacht worden. Dadurch wird erwartet, dass mit geringfügig mehr Aufwand während der ursprünglichen Entwicklung im Auftragsprojekt, eine beträchtliche Verringerung des Aufwands in den Folgeschritten zum qualifizierten IP-Modul erreicht wird. Eine Schätzung ist in Tabelle 17 Spalte 4 dargestellt.

Ergänzend kann anhand der Reklamationenstatistik nachgewiesen werden, dass die IP-Qualifizierung das Integrationsrisiko aufseiten des IP-Nutzers reduziert. Die Reklamationenstatistik der IP-Nutzer ist in Abbildung 23 grafisch aufbereitet dargestellt. Sie zeigt, dass die IP-Qualifizierung dazu beigetragen hat, die „langen“ Entwurfsiterationen beträchtlich zu verringern. Die IP-Nutzer profitieren von einem geringeren Time-to-Market Risiko und der IP-Anbieter von geringerem Nacharbeitsaufwand.

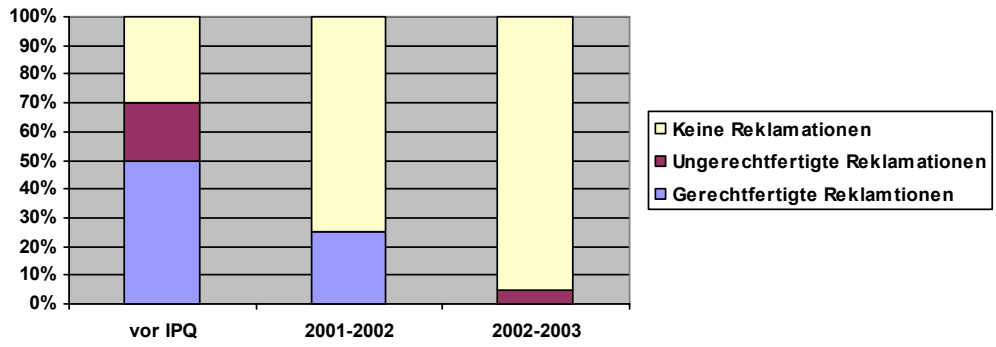


Abbildung 23: Prozentuale Verteilung der Reklamationen bei IP-Auslieferungen

Zum Schluss ist es noch wichtig anzumerken, dass dieser Untersuchung ein firmeninterner IP-Austausch zugrunde liegt. Die Techniken, die in Kapitel 5 beschrieben wurden, sind wesentlich, um zusätzlichen Aufwand

für den IP-Nutzer wegen notwendiger Anpassungen der Integrationsdaten an seine Entwicklungsumgebung zu vermeiden. Dadurch sollte auch eine Übertragung der erzielten Ergebnisse auf Wiederverwendungsprojekte, an denen verschiedene Firmen beteiligt sind, gestattet sein.

7.3 IP-Auslieferungsphasen

Verbesserungen konnten auch bei den IP-Auslieferungsphasen (Export, Übermittlung und Import) erreicht werden. Die erreichte Zeiteinsparung ist besonders erwähnenswert. Die Zeit, die für das Zusammenstellen und Ausliefern eines IP-Moduls benötigt wird, konnte durch den Export-Experten um 80 % verringert werden. Das entspricht einem Größenordnungsverhältnis von Tagen zu Stunden.

Während der Exportphase wird automatisch die Vollständigkeit des IP-Moduls gewährleistet. Durch die zusätzlich generierten Metadaten wird ein einheitliches IP-Format erzeugt. Dieses gestattet eine automatische Eingangskontrolle des IP-Moduls vor einem aufwendigen IP-Import, die unabhängig vom Format des IP-Anbieters ist.

Zusätzlich ermöglicht das einheitliche IP-Format einen automatischen Import eines IP-Moduls in die Wiederverwendungsdatenbank eines IP-Nutzers und die automatische Anpassung an dessen Entwurfsablauf. Der notwendige Zeitaufwand für den Import konnte durch den Import-Experten so ebenfalls von Tagen auf Stunden reduziert werden.

Bedingt durch die sehr guten Ergebnisse der automatischen Qualifizierung, werden Re-Imports, aufgrund von „langen“ Entwurfsiterationen, wesentlich reduziert, was ebenfalls zu einer positiven Aufwandsreduzierung während der Auslieferungsphasen beiträgt.

8 Zusammenfassung und Ausblick

Dieses Kapitel fasst die wichtigsten Ergebnisse der vorliegenden Arbeit zusammen. Dies schließt eine Erfolgskontrolle anhand der erreichten Ziele in Abschnitt 8.1 ein. Darüber hinaus wird in Abschnitt 8.2 ein Ausblick auf künftige Arbeiten gegeben, die auf den vorliegenden Ergebnissen aufbauen können.

Es wurde deutlich gemacht, dass IP-Wiederverwendung absolut notwendig ist, um aktuelle und zukünftige SoC-Entwürfe erfolgreich zu beenden. Da wiederverwendbare IP-Module definierte Qualitätskriterien einhalten müssen, ist es notwendig, auf der Seite des IP-Anbieters die IP-Qualität zu messen, um Qualitätsmängel frühzeitig beseitigen zu können. Für einen IP-Nutzer bedeutet die Möglichkeit, IP-Qualität zu messen, eine Gelegenheit das Integrationsrisiko zu reduzieren, das mit der Verwendung von externen IP-Modulen verbunden ist. Aufgrund der Tatsache, dass IP-Module zu komplex sind, um manuelle Qualifizierungen durchzuführen, müssen automatisierte Qualifizierungslösungen entwickelt werden.

In dieser Arbeit wurde das IP-Qualifizierungsframework vorgestellt. Bereits mit dem Prototyp konnten sehr gute Ergebnisse, durch automatische Qualitätsmessungen einer Teilmenge der standardisierten Qualitätskriterien, erzielt werden. Zudem beschleunigen die integrierten Werkzeuge zur Unterstützung der Auslieferungsphasen, den Export, die Übermittlung und den Import von IP-Modulen.

8.1 Erreichte Ziele

In Abschnitt 3.9 wurden ausgehend vom Stand der Technik die offenen Probleme zusammengefasst, die im Zusammenhang mit der Entwicklung und der Wiederverwendung von IP-Modulen bestehen. Anhand der in Abschnitt 3.9 aufgestellten Liste werden im Folgenden die erreichten Ziele zusammenfassend dargestellt.

Der Autor hat innerhalb der VSIA Beiträge zur Standardisierung von Qualitätskriterien und deren -bewertung geleistet (Abschnitt 3.2 und 7.1). Basierend auf diesen standardisierten Qualitätskriterien wurden automatische Qualitätsmessungen und Möglichkeiten zur Qualitätsbewertung in Abschnitt 4 vorgestellt.

Aufgrund der Standardisierung und Automatisierung von Qualitätsmessungen ist die Überprüfbarkeit der IP-Qualität durch IP-Anbieter und IP-Nutzer gewährleistet. Darauf wurde insbesondere in Kapitel 5 eingegangen. In den Abschnitten 5.5.1-5.5.4 wurde das Problem der Qualitätserhaltung während der IP-Auslieferung gelöst. Zusätzlich ermöglicht die Entwicklung des einheitlichen IPQ-Formats die automatisierte Integration von IP-Modulen in neue Entwurfsabläufe (Abschnitt 5.5.4). In Abschnitt 6.3.4.3 konnte auch dargestellt werden, dass sich dieses einheitliche Übertragungsformat unabhängig von der Ausgangs- und der Zielstruktur automatisch erzeugen lässt. Die einzigen Anforderungen für die Nutzung der Automatisierungsmöglichkeiten des IPQ-Formats sind die Verwendung

definierter, aber nicht zwingenderweise gleicher, IP-Strukturen durch IP-Anbieter und IP-Nutzer und deren Spezifikation in den Konfigurationsdateien des IP-Qualifizierungsframeworks.

In Kapitel 6 und anhand der Ergebnisse in Kapitel 7 konnte der Nutzen des Qualifizierungsframeworks bereits in der nicht vollständigen Ausbaustufe deutlich gemacht werden. Zudem wurde gezeigt, dass sich die entwickelten Qualitätsmessungen in bestehende Entwurfsabläufe integrieren lassen und zu einer effizienteren Wiederverwendung von IP-Modulen beitragen, die trotz des Mehraufwandes für Qualitätsmessungen wirtschaftlich ist. Gemessen wurde eine bis zu 95 prozentige Verringerung des Integrationsaufwands und ein deutlicher Rückgang der Reklamationen vonseiten der IP-Nutzer.

8.2 Ausblick

Im erweiterbaren IP-Qualifizierungsframework sind bereits verschiedene Qualifizierungsmethoden für die Vollständigkeits-, Integritäts-, Quellcode-, Verifikations- und Entwurfsqualitätsmessungen implementiert. Der Ausbau dieses Ansatzes wird es möglich machen, mehr als 50 % der für IP-Module standardisierten Qualitätskriterien (teil-)automatisiert zu überprüfen. Dieses Ergebnis lieferte die Analyse in Abschnitt 4.1.

Darüber hinaus dürfen durch die Nutzung der Qualifizierungsmethoden bereits im Auftragsprojekt und den weiteren Ausbau des Frameworks weitere Aufwandseinsparungen erwartet werden.

Eine mögliche Erweiterung des Qualifizierungsframeworks ist die applikationsspezifische Qualifizierung sowie die Qualifizierung parametrisierbarer IP-Module durch die Integration der entsprechenden Werkzeuge [80].

Durchaus denkbar ist es, den Einsatz des Qualifizierungsframework nicht auf Hardware-Entwürfe zu beschränken, sondern auf Softwareprojekte auszudehnen. Die Komplexität aktueller Software-Prokjekte und die Anforderungen an einzelne Software-Bausteine hinsichtlich ihrer späteren Systemintegration sind vergleichbar. Die Verwendung von Software-Bausteinen von unterschiedlichen, firmeninternen und fremden Quellen, erfordern ebenfalls die Einhaltung von definierten Qualitätskriterien. Für einen Einsatz des Frameworks in solchen Projekten, müssen die analysierbaren Datenformate durch Integration neuer Experten erweitert werden.

Für weiterführende Quellcodeanalysen können Grammatiken in das Framework integriert werden. Eine entsprechende, modulare Schnittstelle ist durch den Experten-Ansatz vorhanden.

Literaturverzeichnis

[1] E. Gamma, et al: „Entwurfsmuster“, Kapitel „Besucher“ aus dem amerikan. von Dirk Riehle, Addison-Wesley-Longman, 1996, S. 301-318

[2] A. Vörg, N.M. Madrid, W. Rosenstiel, R. Seepold: „IP qualification of reusable designs“ in Tagungsband des Forum on Design Languages (FDL), 2001

[3] Qualitätsnorm DIN EN ISO 9000:2001-01

[4] P. Bricaud, M. Keating: „Reuse Methodology Manual for System-on-a-Chip Designs“, 2. Auflage, Kluwer Academic Publishers, Dordrecht, 1999

[5] Synopsys, Mentor Graphics: „OpenMORE“, Webseite 1998, URL http://www.openmore.com

[6] Virtual Socket Interface Alliance (VSIA), Webseite 2004, URL http://www.vsi.org

[7] VSIA Quality IP (QIP) Bewertung – Microsoft®-Excel-Arbeitsblatt, Version 1.11; Juni 2004, Datei: VSIA QIP-v1.11.xls

[8] VSIA: „QIP Quality Metric – User Guide“, Version 1.11; Juni 2004

[9] VSIA, „VC/IP Quality Development Working Group“ jetzt „IP Quality Pillar“, Webseite 2004, URL http://www.vsi.org/pillars/IP_Quality_Pillar.htm

[10] L. Cooke: „Why we don’t have IP quality yet“ in EEDesign, Juli 2003, URL http://www.eedesign.com/article/printableArticle.jhtml?articleID=17408502

[11] J. Haase: „Design Methodology for IP Providers“ in Tagungsband der Design, Automation and Test in Europe (DATE), 1999

[12] Atrenta: „Spyglass“, Webseite, 2004, URL http://www.atrenta.com/products/spyglass.htm

[13] TransEDA: „VN-Cover“, Webseite 2004, URL http://www.transeda.com

[14] Synopsys: „Design-Compiler“, Webseite 2004, URL http://www.synopsys.com/products/logic/design_compiler.html

[15] Virtual Socket Interface Alliance: „VSIA Deliverables Document“, Version 2.6.0, Mai 2002, URL http://www.vsi.org/resources/TechnicalDocumentsList.htm

[16] R. Seepold, N. M. Madrid, A. Vörg, W. Rosenstiel, M. Radetzki, P. Neumann, J. Haase: „A qualification platform for design reuse“ in Tagungsband des International Symposium on Quality Electronic Design (ISQED), IEEE, New York, 2002

[17] Design & Reuse: „D & R Catalog“, Webseite Juli 2003, URL http://www.us.design-reuse.com/sip/

[18] Synopsys: „IP Directory“, Webseite Dezember 2003, URL http://www.synopsys.com/products/designware/ipdir/

[19] M. Schaaf, M. Visarius, R. Bergmann, R. Maximini, M. Spinelli, J. Lessmann, W. Hardt, S. Ihmor, W. Thronicke, C. Tautz, R. Traphoener: „IPCHL - a description language for semantic IP characterization“ in Tagungsband des Forum on Design Languages (FDL), 2002

[20] M. Schaaf, A. Fressmann, M. Spinelli, R. Maximini, R. Bergmann: „A knowledge representation format for virtual IP marketplaces“ in Tagungsband der International Conference on Case-Based Reasoning (ICCBR), Springer, Trondheim, Norway, 2003

[21] M. Visarius, J. Lessmann, W. Hardt, F. Kelso, W. Thronicke: „An XML format based integration infrastructure for IP based design“ in Tagungsband des Symposium on Integrated Circuits and Systems Design (SBCCI), 2003, Seiten 119–124

[22] Virtual Component Exchange, Webseite Juli 2003, URL http://www.thevcx.com

[23] Mentor Graphics, Synopsys, VSIA: „Mentor Graphics and Synopsys Donate OpenMORE Program to Virtual Socket Interface Alliance“ Presseerklärung am 18.6.2001, URL http://www.vsia.org/news/pressrelease/061801.pdf

[24] V. Jerinic, D. Müller: „Shrinking the parameter space of IP by utilizing parameter domains“ in Tagungsband des IP Based SOC Design Workshop, Grenoble, France, 2002

[25] Philips Semiconductors: „Coreuse standards book“, Technischer Bericht, Philips Semiconductors ReUse Technology Group, Eindhoven, Niederlande, 2001

[26] IPQ, BMBF IPQ Projektwebseite, November 2003, URL https://www.ip-qualifikation.de

[27] TOOLIP, MEDEA+ TOOLIP (A511) Projektwebseite, November 2003, URL http://toolip.fzi.de

[28] Semiconductor Industry Association: „International Technology Roadmap for Semiconductors“ (ITRS), Webseite 1999, URL http://public.itrs.net/files/1999 SIA Roadmap/Home.htm

[29] Semiconductor Industry Association: „International Technology Roadmap for Semiconductors“ (ITRS), Webseite 2001, URL http://public.itrs.net/Files/2001ITRS/Home.htm

[30] A. Reutter, W. Rosenstiel: „An efficient reuse system for digital circuit design“ in Tagungsband der Design, Automation and Test in Europe (DATE), 1999, S. 38

[31] N. Faulhaber, R. Seepold: „A Flexible Classification Model for Reuse of Virtual Components“ in „Reuse techniques for VLSI design“, Kluwer Academic Publishers, Dordrecht, 1999, Seiten 21–36

[32] U. Schlichtmann, B. Wurth: „An Integrated Approach towards a Corporate Design Reuse Strategy“ in „Reuse techniques for VLSI design“, Kluwer Academic Publishers, Dordrecht, 1999, Seiten 37-47

[33] L. Ghanmi, A. Ghrab, M. Hamdoun, B. Missaoui, G. Saucier, K. Skiba: „E-Design based on the reuse paradigm“ in Tagungsband der Design, Automation and Test in Europe (DATE), 2002, Seiten 214–220

[34] T. Zhang, L. Benini, G.D. Micheli: „Component selection and matching for IP-based design“ in Tagungsband der Design, Automation and Test in Europe (DATE), 2001, S. 40

[35] C. Barna, W. Rosenstiel: „Object-oriented reuse methodology for VHDL“ in Tagungsband der Design, Automation and Test in Europe (DATE), 1999, S. 689

[36] H. Lange, M. Radetzki: „IP configuration management with abstract parameterizations“ in Tagungsband des IP Based SOC Design Workshop, Grenoble, France, 2002

[37] D. Corkill: „Blackboard Systems“ in AI Expert, Jahrgang 6, Nummer 9, Januar 1991

[38] D. Rentz: „OpenOffice.org's Documentation of the Microsoft®-Excel File Format“, Version 1.37, 26.8.2004, URL http://sc.openoffice.org/excelfileformat.pdf

[39] A. Khan: „Java Excel API - A Java API to read, write and modify Excel spreadsheets“, Oktober 2004, URL http://www.andykhan.com/jexcelapi/

[40] Apache: „Apache POI-HSSF - Java API To Access Microsoft®-Excel-Format Files“, Oktober 2004, URL http://jakarta.apache.org/poi/hssf/

[41] K. S. Bhogal: „JDBC-ODBC Bridge Driver Enables Spreadsheet-as-database Interaction“, Oktober 2004, URL http://www.devx.com/Java/Article/17848

[42] VSIA: „VSIA-Mitgliederbereich“, Oktober 2004, URL http://members.vsi.org

[43] R. Goering, P. Clarke “Design reuse on a macro level”, 1998, URL http://www.eetimes.com/news/98/1020news/rmm2.html

[44] R. Seepold, N. M. Madrid: „Virtual Components Design and Reuse“ Kluwer Academic Publishers; Dezember 2000; ISBN 0-7923-7261-1

[45] Y. Torroja, F. Casado, F. Machado, T. Riesgo, E. de la Torre, J. Uceda, “Using a Simplified Hardware Model to Analyse the Quality of VHDL Based Designs” in User Forum der DATE, 2000

[46] J. Schneider, S. Rülke, A. Reutter: „Comparison of Different Analysis Approaches for Coding Guidelines and Implementation of a TCL-based Prototype“ in Tagungsband des Forum on Design Languages (FDL), 2000

[47] Tsu-Hua Wang, Chong Guan Tan, „Practical code coverage for Verilog“ in Tagungsband des 4th IEEE International Verilog HDL Conference, March 27 - 29, 1995, Santa Clara, CA, Page 99

[48] IEEE P1603, Advanced Library Format (ALF), URL http://www.eda.org/alf

[49] OpenAccess, 2004, URL http://www.si2.org/openaccess/

[50] R. Goering: „Verilog reader/writer boosts OpenAccess reach“ EETimes 18.3.2004, URL http://www.eedesign.com/showArticle.jhtml?articleID=18400806

[51] SPIRIT Consortium, 2004, URL http://www.spiritconsortium.com/

[52] Sergey Melnik, et al.: „Developing metadata intensive applications with Rondo“, in Journal of Web Semantics, Jahrgang 1, Ausgabe 1, Elsevier, Dezember 2003, Seiten 47-74

[53] D. Dempster, M. Stuart: „Verification Methodology Manual“, 3. Auflage, Teamwork International und TransEDA, Juni 2002

[54] „International Technology Roadmap for Semiconductors“, Edition Executive Summary, 2003, URL http://public.itrs.net/Files/2003ITRS/Home2003.htm

[55] H. Chang, et al.: „Surviving the SOC Revolution - A Guide to Platform-Based Design“, Kluwer Academic Publishers, 2000

[56] VSIA: „Architecture Document“, Version 1.0, 1997

[57] R. Goering: „Design reuse will cut costs in half, study finds“, EETimes, 28.9.1998, URL http://www.eet.com/news/98/1029news/design.html

[58] J. Zaidi „Different platform types are needed for SoC design“ in EEDesign, 31.1.2003 URL http://www.eedesign.com/story/OEG20030131S0057

[59] G. Martin: „Platform-Based Design: Breakthrough or Pipedream?“ DesignCon 2002 Panel, 29.1.2002

[60] OCP-IP Association, Webseite Oktober 2004, URL http://www.ocpip.org

[61] Altera: „Excalibur Entwurfsplattform, Webseite Oktober 2004, URL http://www.altera.com/products/devices/excalibur/exc-index.html

[62] ARM: „AMBA On-Chip Bus“, Webseite Oktober 2004, URL http://www.arm.com/products/solutions/AMBAHomePage.html

[63] ARM: „PrimeXsys“ Entwurfsplattform, Webseite Oktober 2004, URL http://www.arm.com/products/solutions/PrimeXsysPlatforms.html

[64] Freescale: „StarCore“ Entwurfsplattform, Webseite Oktober 2004, URL http://www.freescale.com/webapp/sps/site/taxonomy.jsp?nodeId=0127958594

[65] Philips Semiconductors: „Nexperia“ Entwurfsplattform, Webseite Oktober 2004, URL http://www.semiconductors.philips.com/products/nexperia/

[66] R. Walker, D. Thomas: „A Model of Design Representation and Synthesis“ in 22nd Design Automation Conference, Las Vegas, USA, 1985

[67] J. Bralla: „Design for Manufacturability Handbook“, McGraw-Hill Professional; 2. Auflage, August 1998

[68] B. Cohen: „VHDL Coding Styles and Methodologies“, Kluwer Academic Publishers; 2. Auflage, Februar 1999

[69] A. Crouch: „Design for Test for Digital IC's and Embedded Core Systems“, Prentice Hall PTR; Juni 1999

[70] M. Abramovici et al: „Digital Systems Testing and Testable Design“, Wiley-IEEE Press, September 1994

[71] F. Traverdians: „Zeit und Kosten sparen mit plattformbasiertem Design“, Design & Verification, Publish-Industry Verlag, Oktober 2004, URL http://www.duv24.net

[72] Mentor Graphics: „Platform Express“ Entwicklungsumgebung, November 2004, URL http://www.mentor.com/platform_ex/

[73] Mentor Graphics: „ModelSim“ Simulator, November 2004, URL http://www.model.com/

[74] Cadence: „Incisive“ Simulator, November 2004, URL http://www.cadence.com/products/functional_ver/incisive_unified_simulator/

[75] Mentor Graphics: „Seamless“ Hardware/Software Co-Verification, November 2004, URL http://www.mentor.com/seamless/

[76] D. Manners: „Two faces of intellectual property“ in ElectronicsWeekly am 28.10. 2004, URL http://www.electronicsweekly.com/articles/article.asp?liArticleID=37783

[77] H. Yeates: „Rising cost of IP hits vendors“ in ElectronicsWeekly am 26.10.2004, URL http://www.electronicsweekly.com/articles/article.asp?liArticleID=37788

[78] heise online: „AMD will AMD64-Prozessoren auch in Singapur fertigen lassen“, 9.11.2004, URL http://www.heise.de/newsticker/meldung/53057

[79] Synopsys: „Design-Compiler Command-Line Interface Guide“, Version 2002.05

[80] V. Jerinic, D. Müller: „Safe integration of parameterized IP“ in Integration, the VLSI journal, 37 (2004), ELSEVIER B. V., ISSN 0167-9260, pp.193-221

Anhang A

A.1 Implementierung

Tabelle 18: Typen für logische Teile von logischen VHDL-Einheiten

logicalPartType	Erklärung
VHDLArchitecture	Dokumentiert, dass die übergeordnete logische Einheit (VHDL-Entwurfseinheit) eine `architecture`-Anweisung enthält. Damit ist die logische Einheit eine Quelle für Verweise auf eine Zieleinheit vom Typ `entity`.
VHDLComponent	Dokumentiert, dass die übergeordnete logische Einheit (VHDL-Entwurfseinheit) eine `component`-Anweisung enthält. Dadurch ist die übergeordnete logische Einheit eine Quelle für Verweise auf Zieleinheiten, die eine entsprechende `entity`-Anweisung enthalten.
VHDLConfiguration	Dokumentiert, dass die übergeordnete logische Einheit (VHDL-Entwurfseinheit) eine `configuration`-Anweisung enthält. Damit ist die logische Einheit eine Quelle für Verweise auf Zieleinheiten der Typen `entity` und `architecture`.
VHDLEntity	Dokumentiert, dass die übergeordnete logische Einheit (VHDL-Entwurfseinheit) eine `entity`-Anweisung enthält. Damit ist die logische Einheit als Ziel für Verweise von VHDL `architecture`- und `configuration`-Anweisungen vorgemerkt.
VHDLFile	Dokumentiert, dass die übergeordnete logische Einheit (VHDL-Entwurfseinheit) eine `file`-Anweisung enthält. Die übergeordnete logische Einheit ist eine Quelle für einen Verweis auf eine Datei. Diese Datei wird durch eine logische Einheit mit dem Wert Dateinamen als Namensattribut modelliert.
VHDLFor	Dokumentiert, dass die übergeordnete logische Einheit (VHDL-Entwurfseinheit) eine `for`-Anweisung enthält. Die `for`-Anweisung spezifiziert die Konfiguration einer Komponente (`component`). In diesem Kontext wird eine weitere `use`-Anweisung verwendet, die verschieden von dem VHDLUse oben ist. Im `for`-Kontext verweist `use` auf das zu verwendende `entity`-/`architecture`-Paar beziehungsweise die `configuration`. Die übergeordnete logische Einheit ist demnach eine Quelle für Verweise auf Zieleinheiten, die `entity`- und `architecture`- beziehungsweise `configuration`-Anweisungen enthalten. Anmerkung: Die `for`-Anweisung wird ebenfalls im Schleifenkontext verwendet, dieser wird aber nicht durch den logischen Teil-Typ VHDLFor modelliert.

Tabelle 18: Typen für logische Teile von logischen VHDL-Einheiten (Fortsetzung auf der nächsten Seite)

logicalPartType	Erklärung
VHDLLibrary	Dokumentiert, dass die übergeordnete logische Einheit (VHDL-Entwurfseinheit) eine `library`-Anweisung enthält. Eine `library`-Anweisung verweist auf ein physikalisches Verzeichnis des Dateisystems. Die Zuweisung wird nicht im VHDL-Quellcode, sondern durch das verarbeitende VHDL-Werkzeug vorgenommen. Eine `library`-Anweisung modelliert demnach eine Quelle für einen Verweis auf ein durch das VHDL-Werkzeug beziehungsweise dessen Konfiguration festgelegtes Verzeichnis des Dateisystems.
VHDLPackage	Dokumentiert, dass die übergeordnete logische Einheit (VHDL-Entwurfseinheit) eine `package`-Anweisung enthält. Damit ist die übergeordnete logische Einheit ein Ziel für VHDL `use`–Anweisung und eine Quelle für Verweise auf `package body`-Anweisung.
VHDLPackageBody	Dokumentiert, dass die übergeordnete logische Einheit (VHDL-Entwurfseinheit) eine `package body`-Anweisung enthält. Dadurch ist die übergeordnet logische Einheit ein Ziel für Verweise von `use`- und `package`-Anweisungen. Ein `package body` implementiert die Funktionen und Prozeduren, die in einem `package` deklariert werden.
VHDLUse	Dokumentiert, dass die übergeordnete logische Einheit (VHDL-Entwurfseinheit) eine `use`-Anweisung enthält. Damit ist die logische Einheit eine Quelle für Verweise auf Zieleinheiten, die `package`– beziehungsweise `package body`-Anweisungen enthalten.

Tabelle 18: Typen für logische Teile von logischen VHDL-Einheiten

Tabelle 19: Tcl-s Kommandos

Tcl-s-Kommando	Erklärung
`alias`	Erzeugt ein Kommando, das in Wörter expandiert. Die Verwendung dieses Kommandos wird nicht empfohlen, da möglicherweise spezifische Aliase des IP-Anbieters verwendet werden, die beim IP-Nutzer keine Entsprechung haben. Da ein `alias` keine zusätzliche Funktion bietet, kann er grundsätzlich anders codiert werden.
`define_name_rules`	Definiert Namensregeln für die Netzliste. Da keine Abhängigkeiten auf der Abstraktionsebene von Soft-IP-Modulen modelliert werden, hat das Kommando keine Entsprechung in den IPQ-Metadaten
`exit`	Beendet den Design-Compiler. Da keine Abhängigkeiten beziehungsweise qualitätsrelevanten Informationen modelliert werden, hat das Kommando keine Entsprechung in den IPQ-Metadaten als logischer Teil.
`get_unix_variable`	Codiert eine Abhängigkeit von einem Unix-Shell-Skript beziehungsweise der Umgebung des IP-Anbieters. Das Kommando wird als logischer Teil modelliert und unter referencedUnits der ID-Wert der logischen Einheit, in der die Umgebungsvariable gesetzt wird, eingetragen.
`getenv`	Siehe `get_unix_variable`; ist nicht im dc-shell-Modus verfügbar.
`group_variable`	Fügt Variablen einer Gruppe hinzu. Da das Kommando nur verwandte Variablen zusammenfasst, aber weder Auswirkung auf die Werte der gruppierten Variablen hat noch qualitätsrelevant ist, wird das Kommando nicht in den IPQ-Metadaten modelliert.
`if`	Leitet eine Entscheidung im Programmablauf ein. Da das Kommando weder Abhängigkeiten modelliert noch qualitätsrelevant ist, hat das Kommando keine Entsprechung in den IPQ-Metadaten
`info`	Zeigt die Argumentliste oder die Implementierung einer Prozedur an. Da das Kommando weder Abhängigkeiten modelliert noch qualitätsrelevant ist, hat das Kommando keine Entsprechung in den IPQ-Metadaten.
`list`	Zeigt den Wert einer Variablen an oder listet die Variablen und ihre Werte innerhalb einer Variablengruppe auf. Da das Kommando weder Abhängigkeiten modelliert noch qualitätsrelevant ist, hat das Kommando keine Entsprechung in den IPQ-Metadaten.
`quit`	Beendet den Design-Compiler. Da keine Abhängigkeiten beziehungsweise qualitätsrelevanten Informationen modelliert werden, hat das Kommando keine Entsprechung in den IPQ-Metadaten als logischer Teil.

Tabelle 19: Tcl-s Kommandos
(Fortsetzung auf der nächsten Seite)

Tcl-s-Kommando	Erklärung
`redirect`	Verweist auf Ausgabedatei. Lenkt die Ausgabe eines Kommandos in eine Datei um redirect [-append] <Dateipfad> {Kommando}
`set`	Setzt Umgebungsvariablen für alle Child-Prozesse. Dieses Kommando wird als logischer Teil modelliert und merkt die übergeordnete logische Einheit (Tcl-s Format) als mögliches Ziel für Verweise von logischen Teilen vor.
`set_layer`	Erzeugt eine neue Ebene. Da keine Abhängigkeiten beziehungsweise qualitätsrelevanten Informationen modelliert werden, hat das Kommando keine Entsprechung in den IPQ-Metadaten als logischer Teil.
`set_unix_variable`	Setzt den Wert einer Umgebungsvariablen. Dieses Kommando wird als logischer Teil modelliert und merkt die übergeordnete logische Einheit als mögliches Ziel für Verweise von logischen Teilen vor.
`setenv`	Siehe `set_unix_variable`; ist nicht im dc-shell-Modus verfügbar.
`sh`	Führt Unix-Shell-Kommandos aus. Das nachfolgende Konstrukt muss für Abhängigkeitserkennungen und Qualitätsanalysen als logische Einheit des Formats Bourne-Shell ausgewertet werden.
`source`	Liest eine Datei und führt sie als Skript aus `source [-echo] [-verbose] <Dateipfad>` Das Kommando wird als logischer Teil modelliert und unter referencedUnits der ID-Wert der logischen Einheit eingetragen, auf die `<Dateipfad>` verweist.

Tabelle 19: Tcl-s Kommandos

Anhang B Verzeichnisse

B.1 Akronyme und Abkürzungen

In der folgenden Tabelle sind verwendete Akronyme und Abkürzungen mit ihrer Bedeutung zusammengestellt.

Abkürzung	Bedeutung
DFT	Design-for-Test
DWG	Development Working Group
EDA	Elektronische Entwurfsautomatisierung (engl. Electronic Design Automation)
GDSII	Binäres Stream Format zur Übertragung von EDA-Entwürfen auf physikalischer Abstraktionsebene
HDL	Hardware Description Language
HW	Hardware
IP	Intellectual Property
IPQ	IP-Qualifizierung
ITRS	International Technology Roadmap for Semiconductors
OCP-IP	Open Core Protocol-IP IP-Module mit einem Standardinterface zur Kapselung der Verbindungsfunktionalität mit verschiedenen On-Chip Bussen.
OpenMORE	Open Measure Of Reuse Excellence
Ph	Personenstunden
QDWG	Quality Development Working Group (jetzt: Quality Pillar)
QIP	Quality IP; Name der VSIA-IP-Qualitätsmetrik
RMM	Reuse Methodology Manual [4]
SoC	System-on-a-Chip
SW	Software
VC	Virtual Component™
VCT	Virtual Component Transfer
VCX	Virtual Component Exchange
VSIA	Virtual Socket Interface Alliance

B.2 Abbildungsverzeichnis

B.3 Tabellenverzeichnis

B.4 Quellcode-Verzeichnis