

Referenzprofil

Component Developer

Josh C. Beier

Dieses Referenzprofil wurde im Rahmen des bmb+f-geförderten Projekts „Arbeitsprozess-orientierte Weiterbildung in der IT-Branche“ erarbeitet von:



Fraunhofer ISST



IT-Akademie
Oldenburg

Bildungspartner



Offis e. V.

Unternehmenspartner

Danksagung

Diese Profilbeschreibung entstand auf Basis eines Praxisprojekts des Offis e.V. Oldenburg. Felix Bauer danken wir für die ausführliche Darstellung der projektbezogenen und fachlichen Zusammenhänge, insbesondere für die Weitergabe seines Wissens zum Designflow im Rahmen einer ASIC-Entwicklung und die sehr illustrativen Darstellungen aus dem Praxisprojekt. An Guido Kuhlmann von der Tigris GmbH geht Dank für die vielfältigen Anregungen aus der Perspektive von Entwicklern diskreter Elektronik. Ohne diese Beiträge hätte das Dokument nicht entstehen können.

Inhalt

1 EINFÜHRUNG: REFERENZPROZESSE ALS CURRICULA.....	4
1.1 EREIGNIS-PROZESS-KETTEN: SYMBOLIK	4
1.2 REFERENZPROZESS UND TEILPROZESSE.....	6
2 DAS PROFIL: COMPONENT DEVELOPER (KOMPONENTEN-ENTWICKLER/IN).....	9
2.1 TÄTIGKEITSBESCHREIBUNG.....	9
2.2 PROFILTYPISCHE ARBEITSPROZESSE	9
2.3 PROFILPRÄGENDE KOMPETENZFELDER.....	10
2.4 QUALIFIKATIONSERFORDERNISSE.....	11
2.5 EINORDNUNG INS SYSTEM UND KARRIEREPFADE	11
3 REFERENZPROZESS	13
3.1 KOMPONENTENENTWICKLUNG.....	13
3.1.1 Referenzprozess Komponentenentwicklung	14
3.1.2 Das Beispielprojekt: Emulation eines ASIC 80C32a18	16
3.1.3 Prozesskompass Component Development	18
3.1.3.1 Verhandeln mit Kunden	19
3.1.3.2 Erarbeiten der Anforderungsdefinitionen.....	23
3.1.3.3 Beschreiben der Schaltung	28
3.1.3.4 Erstellen einer Testspezifikation	33
3.1.3.5 Prüfen und Optimieren des Entwurfs	37
3.1.3.6 Layouting und Routing.....	41
3.1.3.7 Herstellen der Platine für den Prototypen	44
3.1.3.8 Beschaffen der Bauteile.....	47
3.1.3.9 Erstellen der systemnahen und der Test-Software.....	49
3.1.3.10 Bauen der Testhardware	52
3.1.3.11 Programmieren der Bauteile	55
3.1.3.12 Iteratives Inbetriebnehmen und Testen der Baugruppen	58
3.1.3.13 Unterstützen bei Integration und Test im Zielsystem.....	61
3.1.3.14 Erstellen der Nutzer- und Produktionsunterlagen	63
3.1.3.15 Übergeben der Komponente	65

1 Einführung: Referenzprozesse als Curricula

Das Referenzprojekt des Component Developer verdeutlicht paradigmatisch die diesem Tätigkeitsfeld zugrunde liegenden Arbeitsprozesse, die mit ihnen verbundenen Ansprüche sowie die daraus resultierenden Anforderungen an Inhalt und Durchführung einer qualitativ hochwertigen Weiterbildung.

Das Referenzprojekt erfüllt mehrere Funktionen:

Aus der Praxis für die Praxis

Als Abstraktion tatsächlich stattgefundener Projekte und Prozesse bietet der Referenzprozess eine realistische und leicht nachvollziehbare Abbildung dessen, was die Tätigkeiten eines Component Developer sind.

Prozessorientierung als innovatives „Curriculum“

Als vollständige Darstellung aller wichtigen Arbeitsprozesse sowie der dazugehörigen Qualifikationen, Tätigkeiten und Werkzeuge bieten die Referenzprozesse die Grundlage für die Weiterbildung zum Component Developer. All diese Prozesse müssen – entsprechend den Vorgaben – einmal oder mehrfach durchlaufen werden und ermöglichen dadurch den Weiterzubildenden den arbeitsplatznahen, integrativen Erwerb von relevanten Kompetenzen. Durch den Verbleib im Arbeitsprozess wird nicht nur für die Weiterzubildenden eine hohe Motivation (Arbeit an echten Projekten/Aufgaben) und Nachhaltigkeit erreicht, sondern auch – aus Sicht des Unternehmens – die Kontinuität und Qualität der laufenden Arbeiten gesichert (keine Ausfallzeit durch Seminartage, kein mühsamer Transfer).

Qualitätsstandard für die Weiterbildung

Als Referenz bieten insbesondere die Teilprozesse und die mit ihnen verbundenen Tätigkeits- und Qualifikationsziele einen Qualitätsmaßstab für die arbeitsprozessorientierte Weiterbildung und die resultierenden Abschlüsse. Vollständige Transparenz und klare Zielvorgaben ermöglichen die qualitativ hochwertige Absicherung auch komplexer Kompetenzen sowie den systematischen Erwerb des notwendigen Erfahrungswissens.

Transferprozesse

Die Generalisierung des Referenzprojekts aus der Praxis und seine didaktische Anreicherung ermöglichen eine leichte Auswahl angemessener Transferprozesse, deren Bearbeitung die Grundlage der Weiterbildung ist. Transferprozesse sind reale Prozesse, die Referenzprojekte in einer lernförderlichen Umgebung abbilden. Abgeschlossene Transferprozesse auf Basis der hier dargestellten Anforderungen und Qualitätsmaßstäbe sind nicht nur Qualifikationsnachweis des Einzelnen, sondern bilden auch die Basis eines angemesseneren und zielgerichteteren Umgangs mit Geschäfts- und Arbeitsprozessen im Unternehmen.

1.1 Ereignis-Prozess-Ketten: Symbolik

Die Darstellung des Referenzprozesses in Form von Ereignis-Prozess-Ketten¹ ermöglicht einen schnellen Überblick. Vollständigkeit kann leicht überprüft werden, Anpassungen und Modifikationen in Hinblick auf das eigene Unternehmen sind problemlos möglich und Anknüpfungspunkte an andere Prozesse, aber auch zu weiterführenden Informationen ergeben sich automatisch.

Die bei der Darstellung der Referenz- und Teilprozesse verwendete Modellierungssprache stellt eine Anpassung und Weiterentwicklung der klassischen EPK-Modellierung dar:

¹ Vgl. A.-W. Scheer, *Wirtschaftsinformatik*, Springer 1998.

Referenz- wie Teilprozesse sind aus der Sicht des jeweiligen Spezialisten, also als Arbeitsprozesse einer Person dargestellt.

Referenz- wie Teilprozesse stellen in der Regel keinen Geschäftsprozess dar.

Die EPK-Symbole werden hier wie folgt verwendet:

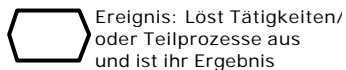
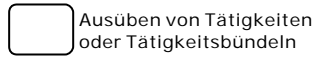
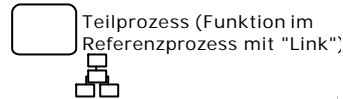


Abbildung 1: Grundlegende Symbole der Referenz- und Teilprozessmodelle.

Die wichtigsten Symbole sind:

- die Tätigkeiten bzw. Tätigkeitsbündel oder Teilprozesse, die mit dem Funktionssymbol dargestellt werden
- die Ereignisse, die Tätigkeiten bzw. Teilprozesse auslösen und Ergebnisse von Teilprozessen sind

Grundsätzlich gilt: Auf ein Ereignis folgt immer ein Teilprozess bzw. eine Tätigkeit.

Ergebnisse von Tätigkeiten sind sehr oft Dokumente; diese werden dann zusätzlich durch das Dokumentsymbol dargestellt.

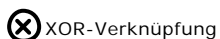


Abbildung 2: Konnektoren.

Wenn Alternativmöglichkeiten bestehen, werden Ereignisse und Teilprozesse/Tätigkeiten über Konnektoren (AND, OR, XOR) verbunden. Dabei steht AND für ein verbindendes „Und“, OR für ein „Oder“, das alle Möglichkeiten offen lässt, und XOR für ein „ausschließendes Oder“, welches nur einen der angegebenen Pfade ermöglicht.

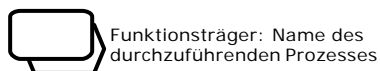


Abbildung 3: Schnittstelle.

Da die Prozesse aus der Sicht des jeweiligen Spezialisten formuliert werden, sind Schnittstellen zu Prozessen anderer Spezialisten oder zu Entscheidungsprozessen auf höherer Ebene notwendig. Dazu wird das Schnittstellensymbol verwendet. Es steht für Prozesse, die der Spezialist nicht selber durchführt, auf deren Durchführung er aber angewiesen ist. Parallel zu jeder Schnittstelle wird die Tätigkeit dargestellt, die der Spezialist selbst in diesem Zusammenhang ausübt, wie „Beraten bei ...“, „Unterstützen bei ...“ oder „Informieren des ...“.

Alle Prozesse werden durch die Verwendung dieser Symbole klar und einfach strukturiert dargestellt und sind offen für die Übertragung in konkrete Transferprozesse.

1.2 Referenzprozess und Teilprozesse

Der hier vorgestellte Referenzprozess und seine Teilprozesse stellen das Curriculum des Spezialistenprofils Component Developer dar.

Der Referenzprozess erhebt nicht den Anspruch eines Vorgehensmodells, sondern bildet beispielhaft den möglichen Arbeitsprozess und Verlauf eines Projekts auf Spezialistenebene ab.

Er bildet die Grundlage für Weiterbildungen und damit einen Qualitäts-, Niveau- und Komplexitätsmaßstab. Die zugehörigen Teilprozesse sind hier beispielhaft modelliert und stellen eine Möglichkeit der Durchführung dar. Einzelheiten zu den unverzichtbaren Prozessen und Kompetenzfeldern sind im Referenzprojekt festgelegt. Die Reihenfolge und die Inhalte der Teilprozesse sind abhängig vom jeweils auszuwählenden Transferprojekt und werden in diesem Zusammenhang festgelegt.

Die Darstellung der Prozesse erfolgt systematisch:

Jeder Prozess wird mithilfe von Ereignis-Prozess-Ketten dargestellt. Einem auslösenden Ereignis folgt eine Funktion, die wiederum ein oder mehrere Ereignisse als Ergebnis hat. Ereignisse und Funktionen können mit AND, OR oder XOR, den Konnektoren verbunden sein.

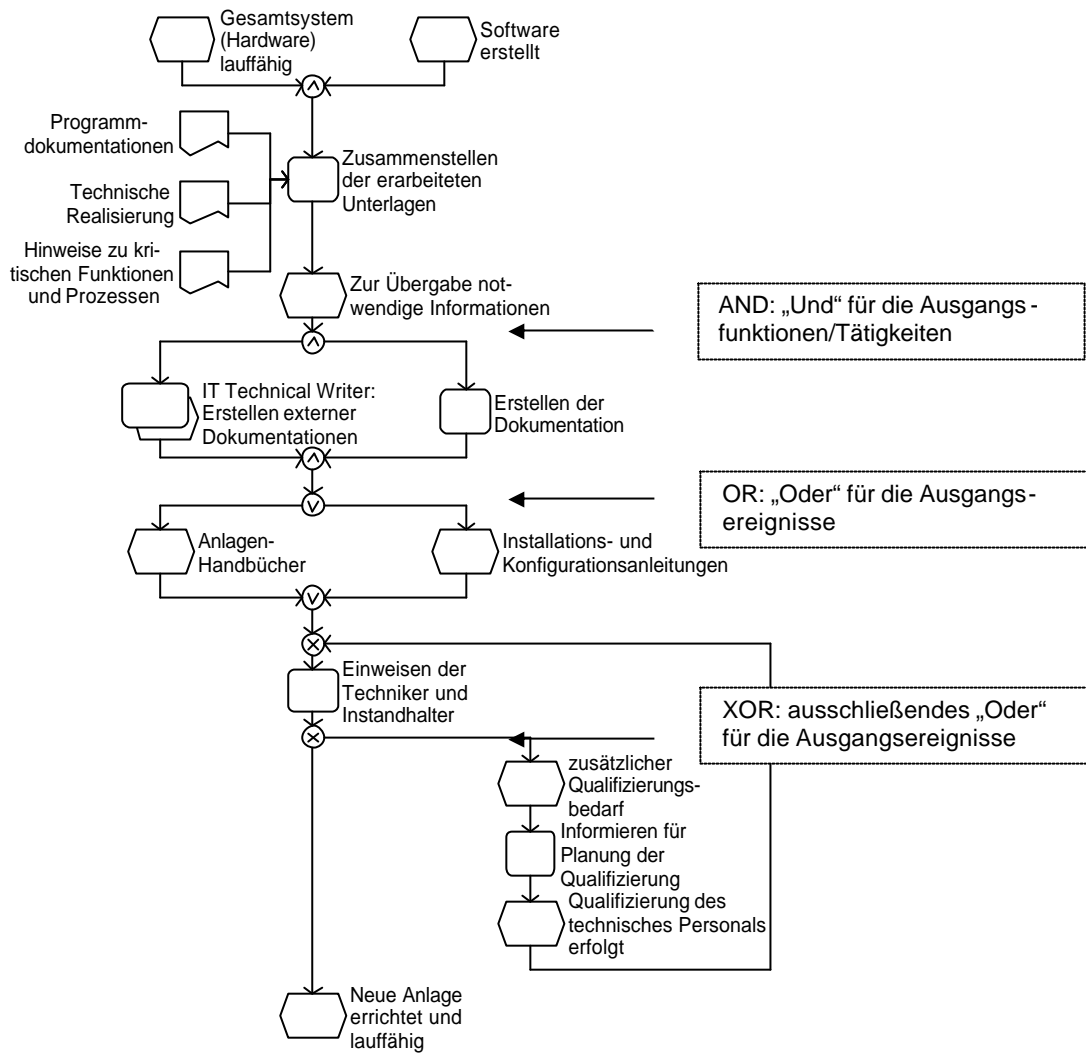


Abbildung 4: Beispielprozess (Teilprozess "Einweisen der Instandhalter" aus dem Profil Industrial IT Systems Technician) mit unterschiedlicher Verwendung von Konnektoren.

Die Verbindung von Referenzprozess und Teilprozessen erfolgt über die Funktionen des Referenzprozesses:

Jede Funktion im Referenzprozess steht für einen Teilprozess.

Ereignisse, die dem jeweiligen Teilprozess direkt vor- oder nachgeordnet sind, sind Anfangs- und Endereignisse der jeweiligen Teilprozesse. Damit stellen die Teilprozesse die Funktionen des Referenzprozesses ausführlich dar und ein Hin- und Herbewegen zwischen Referenz- und Teilprozessen ist jederzeit problemlos möglich.

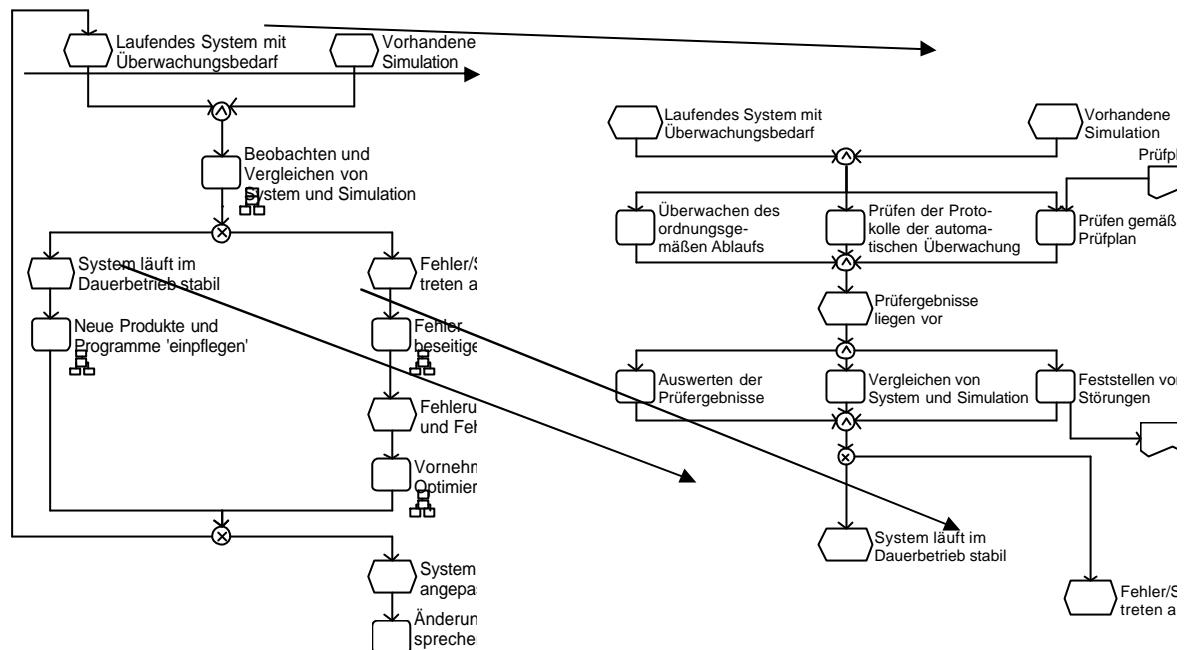


Abbildung 5: Ausschnitt aus dem Referenzprozess des Industrial IT Systems Technician (links) und Teilprozess des Industrial IT Systems Technician "Beobachten und Vergleichen von System und Simulation" (rechts).

Die Teilprozesse stellen so die wesentlichen Teile eines Projekts dar und lassen sich entsprechend auf Transferprojekte übertragen. Den Teilprozessen sind die jeweils wesentlichen Tätigkeiten und Kompetenzfelder zugeordnet.

2 Das Profil: Component Developer (Komponentenentwickler/in)

Component Developer² entwickeln und realisieren Hardwarekomponenten und Geräte.

2.1 Tätigkeitsbeschreibung

Component Developer analysieren geforderte Funktionalitäten für Hardwarekomponenten und Geräte, erfassen und bewerten technische Bedingungen und Standards sowie technische Umgebungen. Sie überprüfen technische Voraussetzungen für Systeme, beraten betriebsinterne und externe Kunden hinsichtlich der technischen Realisierbarkeit der Konzepte und verständigen sich über technische Lösungen. Zu ihren Aufgaben gehört die Projektplanung der einzelnen Projektschritte des Entwicklungsprojekts. Sie arbeiten kooperativ in heterogenen Teams.

Component Developer konzipieren und realisieren Hardwarelösungen sowohl für diskrete als auch für eingebettete Systeme und erstellen hardwarenahe Software. Sie lösen Schnittstellenprobleme, programmieren Schnittstellen und binden diese in Systeme ein. Sie testen Hard- und integrierte Softwarekomponenten im Labor und unterstützen bei Integration und Test im jeweiligen Zielsystem, analysieren und strukturieren dabei auftretende technische Probleme.

Component Developer erstellen technische Dokumentationen und Betriebsanleitungen. Sie wirken bei der Erstellung von Produktionsunterlagen für die Serienproduktion der entwickelten Hardwarekomponenten mit. Sie analysieren und strukturieren technische Probleme und leisten Support.

2.2 Profiltypische Arbeitsprozesse

Die im Folgenden beschriebenen Teilprozesse dokumentieren den gesamten profiltypischen Arbeitsprozess des Component Developer. Die Beherrschung dieses Arbeitsprozesses in Verbindung mit den Kompetenzen in den jeweiligen Kompetenzfeldern und der Berufserfahrung bilden die Grundlage für die berufliche Handlungskompetenz.

1. Verhandeln und Erstellen von Angeboten, Festhalten aller wesentlichen Daten für die Entwicklung, den Zeit- und Kostenrahmen
2. Erarbeiten der Spezifikation der Hardwarekomponente zusammen mit dem Kunden
3. Ausarbeiten einer Testspezifikation, die es ermöglicht, die Hardwarekomponente in allen Phasen der Entwicklung (Entwurf, Prototyp, Kleinserie) gegen die Spezifikation zu testen
4. Entwerfen der Schaltung und Teilsimulation von Funktionseinheiten, bis die Schaltung der Spezifikation entspricht
5. Programmieren von Bauteilen
6. Routen der Verbindungen und Layouten von Schaltungsträgern
7. Programmieren der systemnahen Software und der Testsoftware
8. Beschaffen der für die Herstellung des Prototyps notwendigen Bauteile, Programmieren der programmierbaren Bauteile
9. Bauen der Testhardware für den Prototyp

² Das Kapitel 2: gibt – mit Ausnahme des Abschnitts 2.5 „Einordnung in das System und Karrierepfade“ – den offiziellen Text der „Vereinbarung über die Spezialistenprofile im Rahmen des Verfahrens zur Ordnung der IT-Weiterbildung“ vom 25.05.2002 (Bundesanzeiger 105, ausgegeben am 12.06.2002) wieder.

10. Integrieren der Softwarekomponenten
11. Bestücken der Bauteile, Inbetriebnehmen einzelner Teile und Testen der Funktionsgruppen des Prototyps
12. Testen des Prototyps im Labor
13. Unterstützen bei Integration und Test im Zielsystem des Kunden
14. Erstellen der Schaltungs- und Nutzerdokumentation
15. Erstellen der Produktionsunterlagen (Layout-Daten, Maskendaten, Bestückungsliste, Bestückungsplan)

2.3 Profilprägende Kompetenzfelder

Die Beherrschung der profiltypischen Arbeitsprozesse setzt Kompetenzen unterschiedlicher Reichweite in den nachstehend aufgeführten beruflichen Kompetenzfeldern³ voraus. Den Kompetenzfeldern sind Wissen und Fähigkeiten sowie typische Methoden und Werkzeuge unterschiedlicher Breite und Tiefe zugeordnet.

Grundlegend zu beherrschende, gemeinsame Kompetenzfelder⁴:

- Unternehmensziele und Kundeninteressen
- Problemanalyse, -lösung
- Kommunikation, Präsentation
- Konflikterkennung, -lösung
- fremdsprachliche Kommunikation (englisch)
- Projektorganisation, -kooperation
- Zeitmanagement, Aufgabenplanung und -priorisierung
- wirtschaftliches Handeln
- Selbstlernen, Lernorganisation
- Innovationspotenziale
- Datenschutz, -sicherheit
- Dokumentation, -standards
- Qualitätssicherung

Fundiert zu beherrschende, gruppenspezifische Kompetenzfelder:

- Engineering-Prozesse
- Systemanalyse, -modellierung, -entwicklung
- Entwicklungsstandards (Leistungsfähigkeit, Sicherheit, Verfügbarkeit, Innovation)
- Qualitätsstandards
- Bussysteme, Protokolle und Schnittstellen
- Hardwareanalysen und Analysewerkzeuge
- Wirtschaftlichkeitsanalysen

³ Die Kompetenzfelder werden in der nachfolgenden Auflistung jeweils durch ein zusammenfassendes Stichwort benannt. Da die Weiterbildung zum Spezialisten auf die erfolgreiche Bewältigung zunehmend offener beruflicher Handlungssituationen sowie ganzheitlichen Kompetenzerwerb abzielt, bildet der Kompetenzerwerb einen integralen Bestandteil der Arbeits- und Weiterbildungsprozesse und lässt sich nur im Zusammenhang mit diesen operationalisieren (vgl. dazu die Abschnitte „Kompetenzfelder“ in den Kapiteln 3.1.3.1.2ff.).

⁴ Jeder Spezialist muss in den in diesem Abschnitt genannten „weichen“ Kompetenzfeldern wie „Kommunikation, Präsentation“, „Konflikterkennung, -lösung“ usw. ein Niveau erreichen, das über dem einer Fachkraft liegt. Das heißt, er muss auch in diesen Feldern zu eigenständigem Handeln in der Lage sein und zum Erreichen des Ziels in dem jeweiligen Feld gegebenenfalls über den Rahmen bekannter Verfahren und Lösungen hinausgehen können.

- Marktüberblick

Routiniert zu beherrschende, profilspezifische Kompetenzfelder:

- Logikentwurf, Schaltungssimulation
- analoge und digitale Schaltungstechnik
- kundenspezifische Schaltkreise
- Mikrosysteme
- Bauteilmontagetechniken (Dünn- und Dickfilm, SMD)
- Methoden und Werkzeuge der Softwareentwicklung
- CAD
- Prototypenfertigung

2.4 Qualifikationserfordernisse

Im Regelfall wird ein hinreichendes Qualifikationsniveau auf der Basis einschlägiger Berufsausbildung oder Berufserfahrung vorausgesetzt.

2.5 Einordnung ins System und Karrierepfade

Das neue IT-Weiterbildungssystem gibt auf Basis der vier neuen IT-Ausbildungsberufe drei Ebenen für die Weiterqualifizierung vor: Spezialisten, wie auch der Component Developer einer ist, operative und strategische Professionals.

Der wesentliche Unterschied zu den Profilen aus der Gruppe der Developer – und gleichzeitig der Grund für die Einordnung in die Gruppe der Technician – ist, dass der Component Developer Hardwarekomponenten entwickelt und nicht Software- oder IT-Lösungen. Sein Aufgabenfeld ist ursächlich ein technisches/elektronisches, in dem dezidiertes und sehr spezielles Know-how im Umgang mit Software- und IT-Tools zur Umsetzung der Zielvorgaben zur Anwendung gebracht wird.

Verwandte Profile

Obwohl das Profil des Component Developer wegen seiner sehr spezifischen Kenntnisse aus den Bereichen digitale und analoge Schaltungstechnik recht klar abgegrenzt ist, gibt es eine Reihe verwandter Profile, die sich in folgende Gruppen einteilen lassen:

Industrial IT Systems Technicians: Eine typische Zielumgebung zu entwickelnder Komponenten sind Steuerungen von Industrieanlagen. Industrial IT Systems Technicians nehmen mehr Aufgaben aus den Bereichen Wartung, Betrieb, Anpassung wahr und haben breitere Kenntnisse in den Bereichen Netzwerk-/Verbindungstechnik und Aktorik/Sensorik. Aufgrund seiner detaillierten Kenntnisse im Programmierungs-, Schaltungs- und Signalverarbeitungsbereich wird ein Component Developer eher an Anpassungen, Neuentwürfen, Simulationen oder Programmierungen von Industrieanlagen beteiligt sein als an Betrieb und Wartung.

Software Developer: Kenntnisse aus dem Bereich Programmierung von Steuerungen und Systemkomponenten schaffen eine Nähe zum Software Developer. Moderner Schaltungsentwurf kann auf einer reinen Softwarebasis unter Verwendung von Hochsprachen erfolgen. Umfangreiche Kenntnisse im Bereich Softwareentwicklung sind die Folge. Die Abgrenzung ergibt sich daraus, dass selbst wenn der Schaltungsentwurf auf einer reinen Softwarebasis erfolgt, umfassende Kenntnisse aus den Bereichen Elektronik, Schaltungsentwurf, Signalverarbeitung etc. nötig sind und einen spezifischen Kern des Kompetenzspektrums des Component Developer bilden. Für Component Developer sind Softwarekomponenten kein „Selbstzweck“, sondern ein Mittel zum Zweck, Funktionalität von Hardwarekomponenten umzusetzen.

Übergang in das Elektrotechnik-Weiterbildungssystem

Die Bereiche Industriesysteme und Komponentenentwicklung stellen eine Brücke dar zwischen dem IT-Weiterbildungssystem und einem äquivalenten Weiterbildungssystem der Elektrotechnik, welche z. B. die Bereiche Mechatronik oder Nachrichtentechnik umfasst.

Technische Spezialisten, deren Tätigkeitsschwerpunkte nicht in der softwarebasierten Entwicklung, sondern in Einsatz, Integration, Wartung und Betrieb von Komponenten in den unterschiedlichen Zielumgebungen der Elektrotechnik liegen, sind dort eingeordnet.

Aufstiegsqualifizierung

Aufstiegsqualifizierungen aus dem Tätigkeitsfeld des Component Developer sind möglich, wobei wegen der ausgeprägten technischen und technologischen Expertise insbesondere die technisch orientierten Professional-Profile im Fokus stehen dürften. Beim IT Systems Manager sind als Schwerpunkte Technologie- und Lösungsanalysen sowie das Planen und Umsetzen komplexer Entwicklungsprozesse denkbar. Bei weiter gehender Qualifizierung bietet sich die Entwicklung zum IT Technical Engineer an, wobei in der Praxis aufbauende kaufmännische Qualifizierungen und ein Ingenieurstudiengang üblich sind.

3 Referenzprozess

Der Referenzprozess gibt die jeweiligen Abläufe auf hohem Abstraktionsniveau wieder und ermöglicht so einen Überblick.

Mit den Teilprozessen wird in den Referenzprozess hineingezoomt. Die Teilprozesse entsprechen damit in etwa der Abbildung von Arbeitsprozessen, sie stellen einen konkreten Tätigkeitsverlauf, einschließlich auslösendem Ereignis und Ergebnis, dar.

Die zur Durchführung der Teilprozesse notwendigen Tätigkeiten und Kompetenzfelder werden jeweils in einem separaten Abschnitt aufgelistet.

Das Praxisprojekt dient als Beispiel zur Konkretisierung und Veranschaulichung. Es ist ein echtes, bereits durchgeführtes Projekt, auf dessen Grundlage die hier dargestellten Referenz- und Teilprozesse entwickelt wurden.

3.1 Komponentenentwicklung

Die Entwicklung einer Komponente als Referenzprozess des Component Developer besteht – kurz zusammengefasst – aus folgenden ineinander greifenden Teilen:

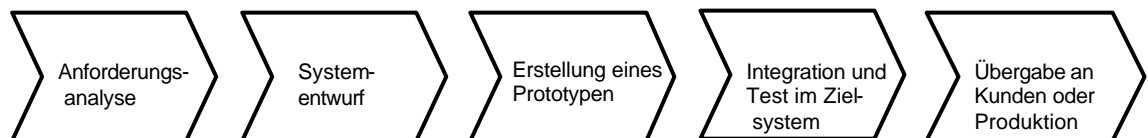


Abbildung 6: Zusammenfassung des Referenzprozesses Komponentenentwicklung.

Der hier dargestellte Referenzprozess bezieht sich sowohl auf die Erstellung von Systemen unter Verwendung von diskreten Komponenten (wie z. B. klassische Bauelemente oder Mikroprozessoren, Small-bis-Large-Scale-Integration-Komponenten) als auch die rein softwarebasierte Realisierung aller Funktionen in Form von (re-)konfigurierbaren Systems-On-a-Chip.

In den letzten Jahren hat sich die Grenze zwischen Hard- und Softwareentwicklung zunehmend verwischt. Software- und nun auch Hardwaredesigns werden auf einem Hochsprachniveau dokumentiert, kompiliert und in eine Hardwarekomponente geladen: Komponentenentwickler, deren Ziel nicht die Entwicklung rein diskreter Elektronik ist, beschreiben ihre Hardware unter Verwendung moderner Synthesetools mit Programmiersprachen wie VHDL (Hardware Description Language) oder Verilog oder gar klassischen Sprachen wie C; Hardwaresysteme werden so auf der Grundlage ihrer logischen Strukturen oder ihres Verhaltens beschrieben und in (re-)konfigurierbaren programmierbaren Bauteilen komplett implementiert. Durch die enorme Zunahme von Leistungsfähigkeit, Dichte logischer Strukturen und preiswerter Verfügbarkeit bei konfigurierbaren logischen Bauelementen können gesamte Hardwaredesigns als System-On-A-Chip zunehmend auch in (re-)konfigurierbaren programmierbaren Bauteilen realisiert werden.

Auch wenn sich die Herstellung z. B. eines anwendungsspezifischen Schaltkreises von der einer diskreten, platinenbasierten Komponente im Detail unterscheidet, so sind die nötigen Kompetenzen und Prozessschritte bei Entwurf und Prototyping strukturell dieselben. Die Modellierung des Prozesses „Component Development“ bezieht sich auf eine gemischte Hard-/Softwaremethode und trägt dem durch eine angemessene Abstrahierung der einzelnen Prozessschritte Rechnung.

3.1.1 Referenzprozess Komponentenentwicklung

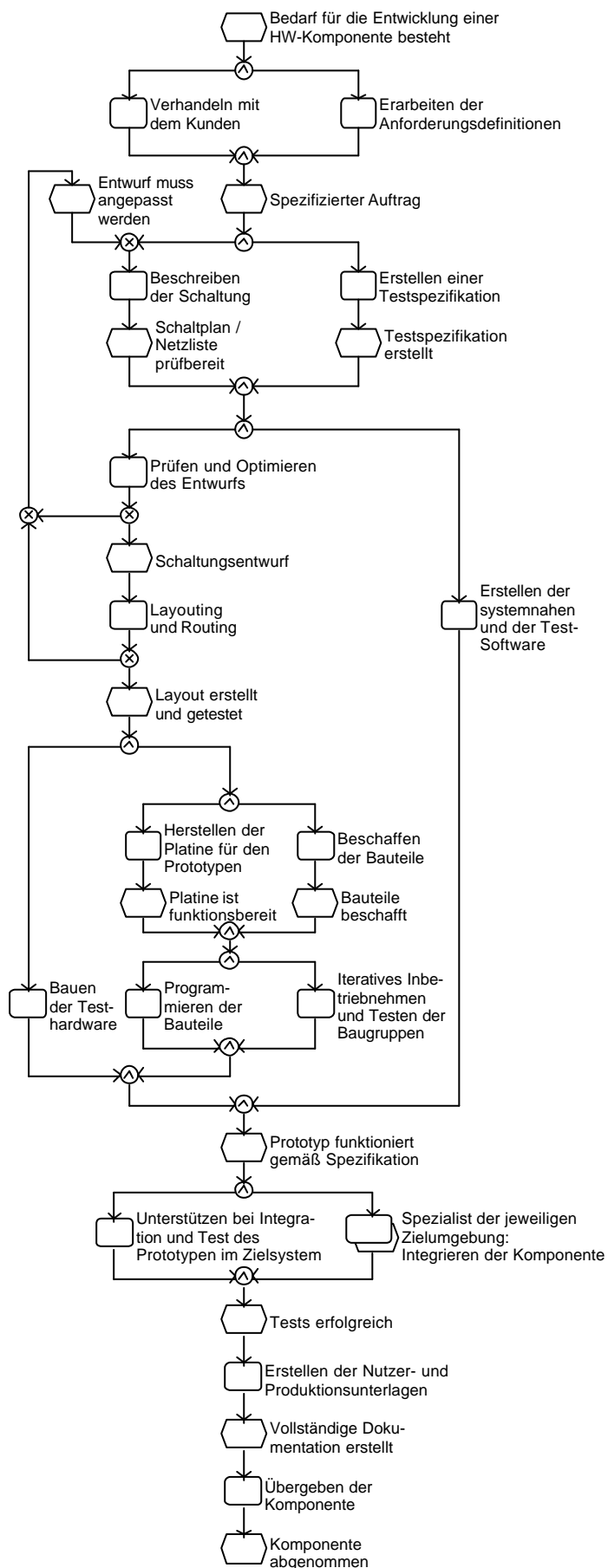


Abbildung 7: Referenzprozess Komponentenentwicklung.

Der dargestellte Prozess umfasst das gesamte Spektrum von Kenntnissen und Kompetenzen des Component Developer: von der Anforderungsanalyse unter Berücksichtigung technologischer Aspekte, über Hard- und Softwarepartitionierung und -entwurf, Layout, Herstellung und Test bis zur Integration und Übergabe. Am Ende des Prozesses steht eine Komponente, die die den festgelegten Spezifikationen in vollen Umfang genügt. Der eigentliche Design-Flow umfasst die Teilprozesse „Beschreiben der Schaltung“, „Prüfen und Optimieren des Entwurfs“ sowie „Layouting und Routing“, in denen es zu einem hohen Maß an Iterationen kommen kann.

Nicht in jedem Projekt wird jeder Teilprozess den gleichen Umfang und die gleiche Komplexität haben, insbesondere hängt das Ausmaß, in dem Simulationen durchgeführt werden, stark von der Komplexität des vorhandenen Systems sowie von der Zieltechnologie ab. Kleine Projekte werden vom Component Developer üblicherweise alleine betreut und durchgeführt. In großen Projekten hingegen werden mehrere Component Developer mit unterschiedlichen Arbeitsschwerpunkten zusammenarbeiten – u. U. auch mit entsprechend qualifizierten Entwicklungsingenieuren. Die Komplexität der zu verwendenden Tools führt in der Praxis gewöhnlich zu Fokussierungen in den Bereichen Hardwarebeschreibung, Schaltungsentwurf, Layout (Place And Route) sowie Simulation.

Abhängig vom Kontext der jeweiligen Zielumgebung der Komponente erfolgen die Integration und der Test unter Mitwirkung entsprechender Spezialisten. Der Component Developer ist hier eher als Spezialist vor Ort zu verstehen, der seine Erfahrungen aus der Konzeption und mit den verwendeten Technologien einbringt, um Funktion und Integration unter Produktionsbedingungen erstellter Komponenten zu gewährleisten.

Üblicherweise wird der Entwurf einer Komponente eine Kombination aus Software- und Hardwaremethoden sein. Neben der Auswahl geeigneter – auch (re-)konfigurierbarer – Hardware-Elemente gilt es, die nötige Software zu entwickeln. Zur Minimierung der einmaligen Entwicklungskosten sind Schaltungen nur in unbedingt nötigem Umfang neu zu entwerfen. Die Tendenz, ein komplettes System per Software-Methoden zu beschreiben und zu implementieren hat natürlich Einfluss auf den hier vorgestellten Referenzprozess. Da eine Unterscheidung zwischen Hardware- und Softwareentwicklung heute zunehmend schwieriger wird, wurden beide Implementierungsstrategien – der kombinierte Einsatz diskreter und (re-)konfigurierbarer Bauelemente sowie die Realisierung des gesamten Systems als (konfigurierbares) integriertes Bauteil – im hier dargestellten Referenzprozess berücksichtigt.

3.1.2 Das Beispielprojekt: Emulation eines ASIC 80C32a18

Das Projekt ist Teil des größeren Projekts „PickToLight“, das bei OFFIS durchgeführt wird. Ziel des PickToLight-Projekts ist die Entwicklung des ASICs (Application Specific Integrated Circuit) 80C32a18. Der ASIC ist zentraler Bestandteil von Displaymodulen für ein neu zu entwickelndes Lagerhaltungssystem.

Jedes Fach eines Lagers, das mit diesem System ausgestattet ist, erhält ein Displaymodul, das Informationen für den Betrieb des Lagers auf einer LED-Anzeige darstellt. Über Tasten können das Fach betreffende Vorgänge manuell bestätigt werden. Mehrere dieser Module werden über einen seriellen Bus mit einem Steuerrechner verbunden.

Eine schematische Abbildung der Architektur des ASIC zeigt die folgende Abbildung:

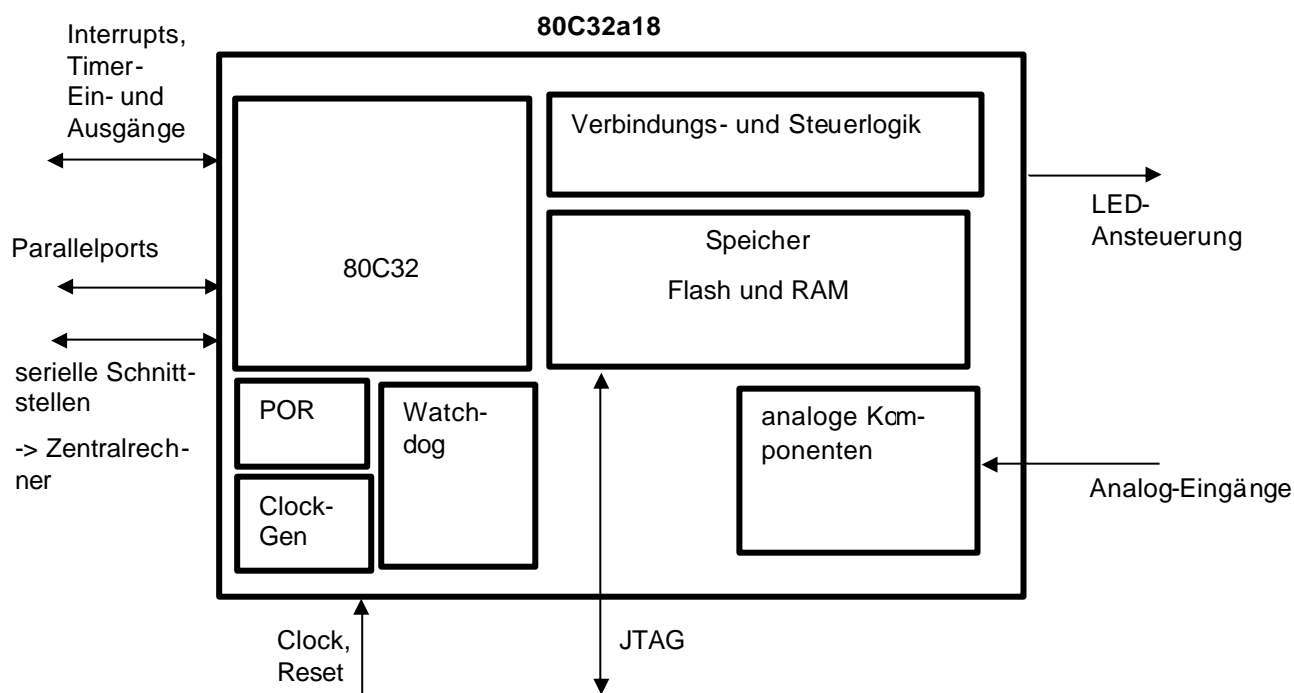


Abbildung 8: Struktur des ASIC 80C32a18

Um den ASIC-Entwurf vor der Fertigung unter möglichst realistischen Bedingungen und mit leicht erfassbaren Ergebnissen zu testen, wird die Schaltung des ASICs auf einem FPGA (Field Programmable Gate Array) nachgebildet und emuliert. Inhalt dieser 80C32a18-Emulation ist die Entwicklung eines zum Simulieren und Debuggen geeigneten Umgebungsmodells und das Einbinden des ASIC-Modells in das Umgebungsmodell.

Die dazu notwendigen Arbeiten umfassen:

- Spezifizieren, wie die Umgebung des ASICs modelliert wird; insbesondere Spezifikation der Kommunikation FPGA – Umgebung
- Aufbauen der Schaltung zum Modellieren der ASIC-Umgebung; insbesondere Aufbauen bzw. Anbinden eines LED-Displays
- Modellieren des ASICs als FPGA-Design; insbesondere Nachbilden der IP-Komponenten
- Modifizieren eines schon existierenden FPGA-Designs, in das beliebige andere Designs zum Testen eingebunden werden können. Das Design muss so angepasst werden, dass sich das Modell des 80C32a18-ASICs einbinden und debuggen lässt.

- Vervollständigen der Software, die zu einem Teil auf dem PC läuft und die Schnittstelle zum Benutzer bildet und zum anderen Teil als Embedded Software auf dem FPGA
- Funktionstest der Simulations- und Debug-Umgebung
- Dokumentieren

Die entwickelte Komponente besteht aus einem in einem PC eingebauten FPGA-Entwicklungsboard, einem extern angeschlossenen LED-Display und der zugehörigen Software und Dokumentation. Auf dem FPGA befinden sich das Modell des ASICs, Modelle der ASIC-Umgebung und Schaltungen zur Steuerung der Datenflüsse.

Die folgende Abbildung zeigt den groben Aufbau der Hardware für die 80C32a18-Emulation.

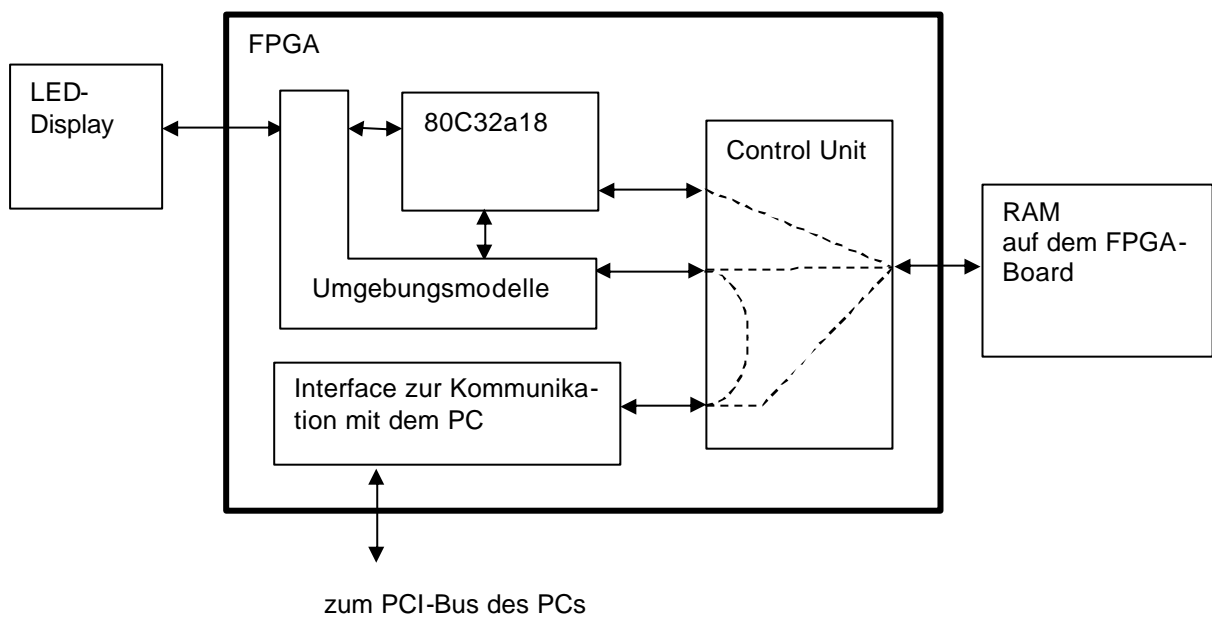


Abbildung 8a: Struktur der Hardware für die 80C32a18-Emulation

3.1.3 Prozesskompass Component Development

1. Verhandeln mit Kunden
2. Erarbeiten der Anforderungsdefinitionen
3. Beschreiben der Schaltung
4. Erstellen einer Testspezifikation
5. Prüfen und Optimieren des Entwurfs
6. Layouting und Routing
7. Herstellen der Platine für den Prototypen
8. Beschaffen der Bauteile
9. Erstellen der systemnahen und der Test-Software
10. Bauen der Testhardware
11. Programmieren der Bauteile
12. Iteratives Inbetriebnehmen und Testen der Baugruppen
13. Unterstützen bei Integration und Test des Prototypen im Zielsystem
14. Erstellen der Nutzer- und Produktionsunterlagen
15. Übergeben der Komponente

Die Teilprozesse geben im Folgenden die Entwicklung einer Hardwarekomponente ausführlich und detailliert wieder. Sie entsprechen einem realen Kundenprojekt, welches als Grundlage für den Referenz- und die Teilprozesse gedient hat und als Beispiel zur Veranschaulichung beschrieben wird.

Nicht alle hier dargestellten Teilprozesse werden in jedem Projekt vorkommen, alle jedoch müssen einem Component Developer auf Spezialistenebene vertraut sein.

3.1.3.1 Verhandeln mit Kunden

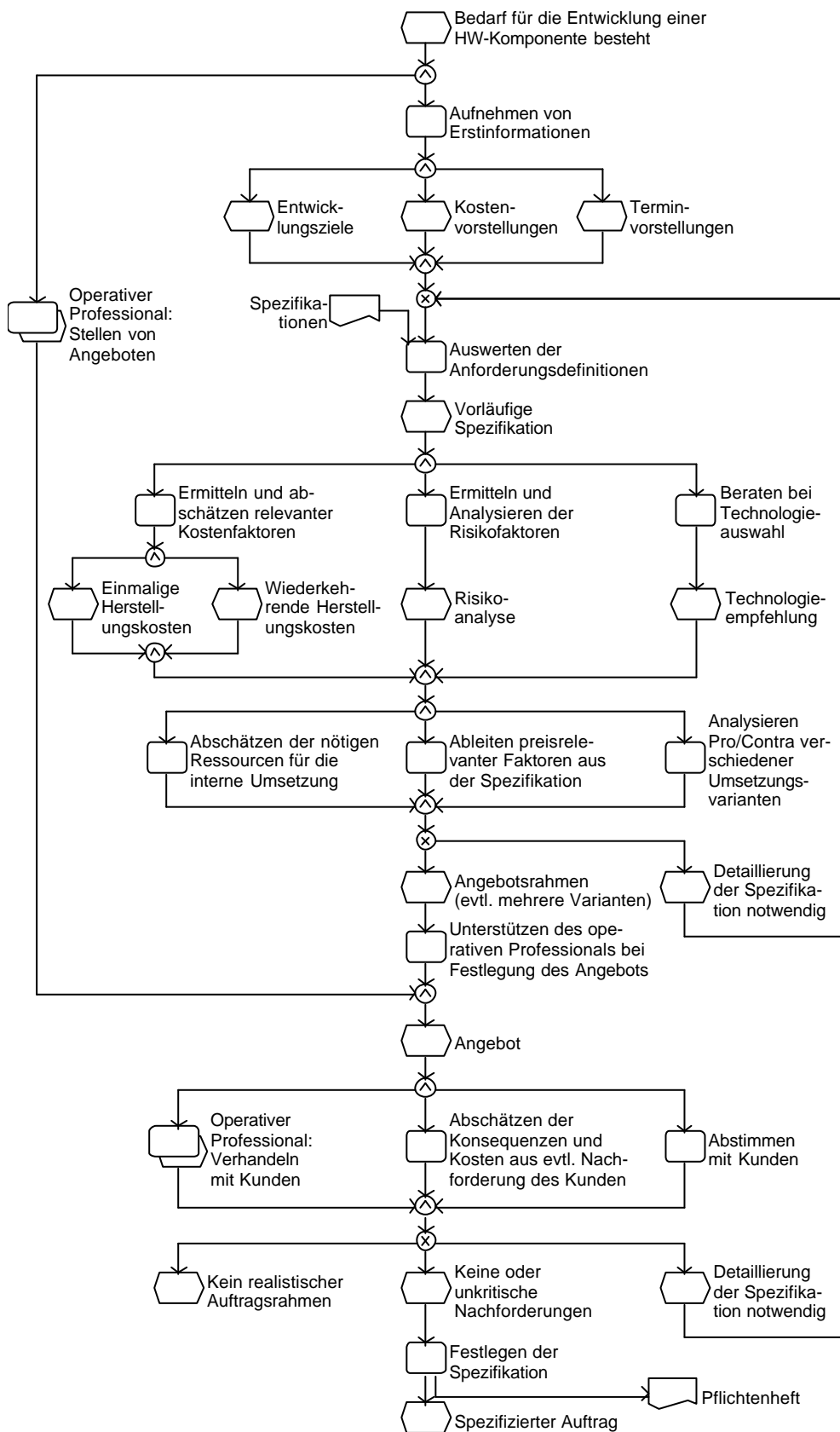


Abbildung 9: Verhandeln mit Kunden.

Diesem Teilprozess gingen bereits Akquisitionsmaßnahmen oder Gespräche seitens eines operativen Professionals voraus, in denen ein realer Bedarf für die Entwicklung einer Hardwarekomponente festgestellt wurde. In diesem Stadium ist eine Zusammenarbeit mit einem operativen Professional typisch: Dieser verantwortet die Preisgestaltung, den Rechtsrahmen und die Bereitstellung interner Ressourcen zur Auftragsabwicklung, während der Component Developer mit seinem Know-how bei der Identifikation von technologie-, preis- und qualitätsentscheidenden Rahmenbedingungen hilft.

Parallel zu diesem Prozess werden auf der Ebene der technischen Experten die Anforderungsdefinitionen erarbeitet und immer wieder mit dem Kunden abgestimmt. Umgekehrt ergeben sich aus den Verhandlungen Anstöße für die Spezifikation. Die Erarbeitung detaillierter Anforderungsdefinitionen kann – insbesondere bei komplexen Zielsetzungen – Bestandteil des Auftrags sein. Unabhängig davon müssen jedoch vor jeder Auftragserteilung technologische und kaufmännische Rahmenbedingungen geklärt sein.

Das Angebot an den Kunden und besonders die intern erarbeitete Umsetzungsempfehlung kann zunächst mehrere Realisierungsmöglichkeiten umfassen. Aufgrund der unterschiedlichen Kosten und des erwarteten Nutzens bei einzelnen dieser Möglichkeiten entscheidet sich der Kunde für eine der angebotenen Lösungen.

Der Component Developer erstellt nicht detaillierte Wirtschaftlichkeits-, Kosten- oder Kosten/Nutzen-Analysen. Dies erfolgt nötigenfalls durch darauf spezialisierte Fachleute. Technische Machbarkeitsanalysen können durch den Component Developer – je nach Vorkenntnissen und Projektspezifika – erstellt oder aber in Zusammenarbeit durchgeführt werden. Der Component Developer ist jedoch in der Lage, solche Analysen auszuwerten und in seinen Empfehlungen und Umsetzungsplanungen zu berücksichtigen.

3.1.3.1.1 Tätigkeiten: Verhandeln mit Kunden

Alle Ableitungen und Analysen erfolgen auf der Basis der aktuell vorliegenden Spezifikationen, die in einem iterativen Prozess ausgearbeitet werden. Die Tätigkeiten des Component Developer konzentrieren sich auf das Ableiten und Abschätzen technologie- und implementierungsbezogener Risiken und Kosten, die die Basis bilden für die Verhandlungen seitens des operativen Professionals.

- Aufnehmen von Erstinformationen zum Abstecken der Rahmenbedingungen: entweder aus Auftaktgesprächen mit dem Kunden oder aus Einweisungen durch einen operativen Professional
- Auswerten der soweit beim „Erarbeiten der Anforderungsdefinitionen“ erstellten Spezifikationen zu Klärung der vorläufigen Verhandlungsbasis
- Beraten bei der Technologieauswahl (auf der Grundlage in der Spezifikation vereinbarter Kriterien wie z. B. Stückzahl, -kosten, Konfigurierbarkeit ...)
- Ableiten preisrelevanter Faktoren aus der Spezifikation (entwurfs- und implementierungsbedingte, z. B. Bauelemente, Herstellungstechnologie)
- Ableiten relevanter Kostenfaktoren unter Berücksichtigung technologischer und auftragsbezogener Rahmenbedingungen sowie Unterscheiden von einmaligen und wiederholten Herstellungskosten (Non-, Recurring Engineering Costs)
- Ermitteln und Analysieren von Risikofaktoren (Auswahl zu verwendender Komponenten, Entwurfsrisiken, Technologierisiken etc.)
- Analysieren der Vor- und Nachteile verschiedener Umsetzungsvarianten – unter Berücksichtigung von wirtschaftlichen Faktoren und Aspekten technischer Machbarkeit
- Abschätzen der nötigen Ressourcen für die interne Umsetzung
- auf der Basis dieser Einschätzungen: Unterstützen des operativen Professionals mit technischen Argumenten bei der Verhandlung
- Abschätzen von Konsequenzen und Kosten aus Nachforderungen des Kunden
- Nachforderungen unter Darstellung der Konsequenzen für den Auftragsrahmen mit Kunden abstimmen
- Erstellen des Pflichtenhefts auf der Grundlage der Verhandlungsergebnisse und der erarbeiteten Anforderungsdefinitionen

3.1.3.1.2 **Kompetenzfelder: Verhandeln mit Kunden**

Fähigkeiten/Fertigkeiten

- technische und vertragliche Informationen aus Gesprächen erfassen, bewerten und klassifizieren können
- technische Dokumentationen, Spezifikationen, Lastenhefte auswerten können
- technik- und technologiebezogene Sachverhalte präzise und kontextbezogen dem Kunden oder Vorgesetzten vermitteln können
- Risiken (in Bezug auf die Planung zu verwendender Komponenten, auf den Einsatz von Technologien in Entwurfs- und Herstellungsprozess etc.) erkennen, abschätzen und priorisieren können, insbesondere in Bezug auf Preis, Termin und Qualität
- Aufwendungen zur Risikobewältigung abschätzen können
- kostenrelevante Faktoren aus der Spezifikation und aus geplanter Umsetzung erkennen und bewerten sowie Kosten/Nutzen-Erwägungen anstellen können
- Analysen (Machbarkeit, Kosten/Nutzen) auswerten und aufgabenbezogen berücksichtigen können
- Kostenfaktoren erkennen und abschätzen können, insbesondere in Bezug auf Entwicklungs- und Produktionskosten – das Durchführen einer exakten Kostenanalyse ist nicht Aufgabe des Component Developer
- auf der Grundlage der Rahmenbedingungen und Entwicklungsziele technische Machbarkeit abschätzen sowie Technologieempfehlungen erarbeiten und nachvollziehbar darstellen können
- auf der Basis von Erfahrungswerten und Technologiekenntnissen Umsetzungsempfehlungen ableiten und benötigte Ressourcen abschätzen können
- mit Lasten- und Pflichtenheften umgehen bzw. Pflichtenheft erstellen können
- Umsetzungsempfehlungen ableiten können unter Berücksichtigung fester Vorgaben des Kunden und wirtschaftlicher Überlegungen
- mit formellen und informellen Hierarchien umgehen können
- zur Abstimmung mit Kunden den eigenen Standpunkt konstruktiv vermitteln können

Wissen

- Entwurfs-, Herstellungstechnologien und -verfahren
- Kostenbegriff, relevante Kostenarten, unterschieden nach einmaligen (Entwicklungs-) und wiederholten (Produktions-) Kosten, Kostenstrukturen
- Grundlagen Kalkulation, Kosten-, Machbarkeits-, Wirtschaftlichkeits- und Nutzwertanalyse
- Verhandlungsprozess, -phasen
- Datenschutz, Umgang mit vertraulichen Informationen
- Dokumentationspflichten
- Lastenheft, Pflichtenheft

Werkzeuge/Methoden

- Befragung, strukturiertes Interview
- Checklisten
- Benchmarking
- Präsentation

3.1.3.1.3 Beispiel: Verhandeln mit Kunden

Da die 80C32a18-Emulation ein Teilprojekt des PickToLight-Projekts ist, trat an die Stelle des Kunden der Projektleiter von PickToLight.

Zunächst ging aus informellen Gesprächen hervor, dass eine Emulation mit dem uns zur Verfügung stehenden FPGA-Board sinnvoll ist und durchgeführt werden soll („Aufnehmen von Erstinformationen“). Daraus ergab sich als vorläufige Spezifikation – die nur in unseren Köpfen bestand –, dass der ASIC mehr oder weniger genau aufs FPGA abgebildet werden soll, die Ausgänge zur Display-Ansteuerung durch ein reales oder virtuelles Display visualisiert werden sollen und die Emulation über den PCI-Bus des PCs, in dem sich das FPGA-Board befindet, steuerbar sein soll. Größere Kosten für das Material würden dabei nicht entstehen, da die wesentlichen Komponenten bereits bei OFFIS vorhanden waren.

Für die endgültige Entscheidung über das Projekt wurden die noch offenen Fragen und der zu erwartende Arbeitsaufwand zusammengestellt („Angebot“). In einem Gespräch mit allen am Projekt Beteiligten wurde dann entschieden, dass alle Komponenten des ASICs so genau wie möglich auf dem FPGA modelliert werden sollen. Ein LED-Display sollte extern angeschlossen werden. Der Benutzer sollte Zugriff auf den Speicher des ASICs haben. Die Arbeitszeit wurde auf drei Monate veranschlagt.

3.1.3.2 Erarbeiten der Anforderungsdefinitionen

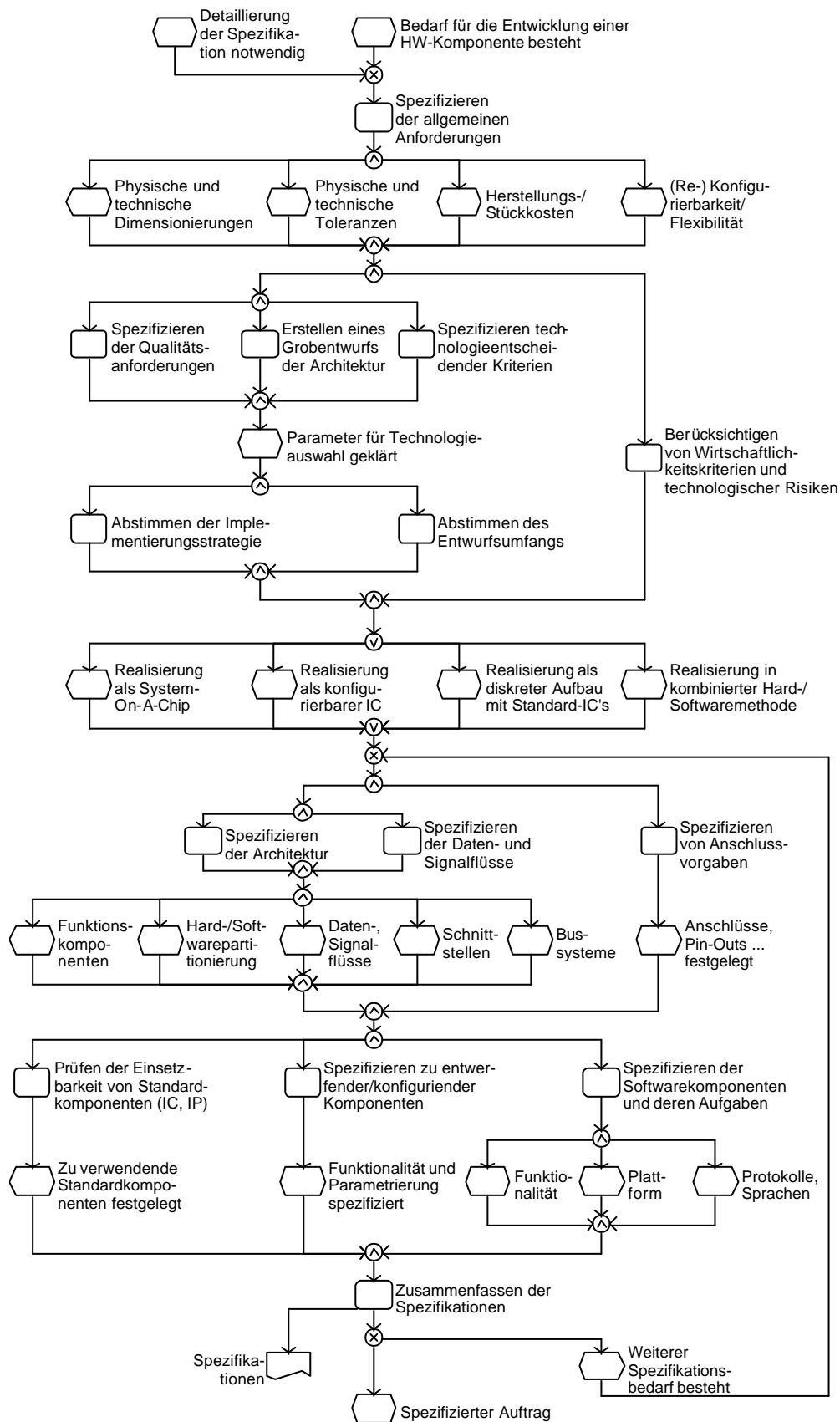


Abbildung 10: Erarbeiten der Anforderungsdefinitionen.

Während des gesamten Teilprozesses sind Informationsbedarfe zu identifizieren und ein enger Kontakt zum Kunden zu halten, um die nötigen Abstimmungen durchführen zu können.

Ein Auftrag wird üblicherweise nur auf der Basis einer klaren Anforderungsspezifikation erteilt. Der Detaillierungsgrad der Spezifikation ist von Projekt zu Projekt verschieden. Je nach Umfang und Art des Projekts kann die Erarbeitung einer genauen Spezifikation auch Bestandteil des Auftrags sein. Diese Erarbeitung ist üblicherweise ein iterativer Prozess, in dem ausgehend vom Kundenwunsch in einem Top-Down-Verfahren die Anforderungen erarbeitet werden. Die Spezifikation erfolgt unter Berücksichtigung von Wirtschaftlichkeitskriterien. Der Component Developer versucht dabei, kostenaufwändige Schaltungsentwürfe durch die Auswahl geeigneter Hard- und Softwarekomponenten zu vermeiden. Dabei ist zu beachten, dass die Software kein „Selbstzweck“ ist, sondern die Funktionalität der zu entwickelnden Komponente ermöglichen soll.

Die Zieltechnologie zur Realisierung ergibt sich aus den auftragsbezogenen Daten und kann von Entscheidungen über die Architektur, über HW/SW-Partitionierung oder auch von der Verfügbarkeit bestimmter Standardkomponenten (z. B. Prozessoren) abhängen. In der Praxis ist durch interne Vorgänge beim Kunden bzw. durch Vorgespräche ein grober Rahmen bereits festgelegt, durch den eine Vorauswahl getroffen wird von Komponentenentwicklung anbietenden Firmen je nach Kompetenzen in den Bereichen Entwurf, diskrete oder Systems-on-a-Chip-Realisierungen sowie digitale oder gemischte Signalverarbeitung (DSP/MSP). Component Developer setzen innerhalb ihrer Spezialisierung für diese Parameter klare Schwerpunkte. Das damit verbundene Know-how ist gefordert für die Erarbeitung der Spezifikation als Basis für die Auftragsstellung und -umsetzung. Zentrale Technologieentscheidungen (Realisierung des System-on-a-Chip oder diskret) sind weniger von Grund auf zu treffen (das ist gewöhnlich ja schon mit der Wahl des Auftragnehmers vorentschieden worden und spiegelt sich im Kompetenzfokus des beauftragten Component Developer), sondern deren Für und Wider sowie deren Ausgestaltung im Rahmen des konkreten Auftrags abzuwägen.

Der Übergang von der Spezifikation zum Schaltungsentwurf ist fließend. Beim Schaltungsentwurf finden die gleichen Tätigkeiten statt wie in der zweiten Hälfte dieses Teilprozesses. Die Grenze wird durch die endgültige Spezifikation markiert, die der Kunde bei Beauftragung unterschreibt.

3.1.3.2.1 Tätigkeiten: Erarbeiten der Anforderungsdefinitionen

- Spezifizieren der allgemeinen Anforderungen, insbesondere Dimensionierungen und Toleranzen (Größe, Leistungsaufnahme, Geschwindigkeit/Performanz, Wärmeverluste etc.), Herstellungs-/Stückkosten und Anforderungen in Bezug auf die (Re-)Konfigurierbarkeit (Notwendigkeit von Updates, mögliche Fehlerbehandlungen etc.)
- Identifizieren von Kosten- und Qualitätsfaktoren (Ausfallsicherheiten, Toleranzen, notwendige Komponenten/Bauteile und deren Preisentwicklung, Einsatzbereiche, Lebensdauer, Anpassbarkeit, Größe, nötige Neuentwicklungen ...)
- Erstellen eines Architektur-Grobentwurfs mit dem Ziel, Technologieentscheidungen zu unterstützen
- Spezifizieren technologieentscheidender Kriterien (Integrationsdichte, Stückzahl, Geschwindigkeit, Konfigurierbarkeit, Größe, Leistungsaufnahme, Wärmeverluste u. a.)
- Abstimmen der Implementierungsstrategie und des Entwurfsumfangs (Optimierungsproblem zwischen funktionalen Zielvorstellungen und Wirtschaftlichkeit/Risiko) – an der Abstimmung können je nach Kompetenzschwerpunkt unterschiedliche Vertreter seitens des Kunden oder des eigenen Entwicklungsteams beteiligt sein
- iteratives Spezifizieren der Architektur/logischen Struktur sowie der Daten- und Signalflüsse, insbesondere: Identifizieren und Beschreiben von Funktionskomponenten, Partitionieren des Gesamtsystems in Hard- und Softwarekomponenten, Identifizieren und Beschreiben von Schnittstellen, Festlegen von Bussystemen
- Spezifizieren und Dokumentieren von Anschlussvorgaben wie Steckverbindungen, Pin-Outs

- Prüfen der Einsetzbarkeit von Standardkomponenten (Katalogbauteilen wie Standard-ICs, Intellectual Property (IP-)Cores, Zielhardware für „Pre-Silicon-Prototypen“) sowie Festlegen von deren Einsatz
- Spezifizieren notwendig zu entwerfender anwendungsspezifischer Komponenten – das beinhaltet die Entwurfsziele, Parametrierung, Konfigurierbarkeit etc.; bei reinen System-on-a-Chip-Entwürfen bildet diese Tätigkeit einen Schwerpunkt und kann sehr umfangreich sein
- Spezifizieren der Softwarekomponenten, insbesondere das Beschreiben der in programmierbaren Bauteilen umzusetzenden Funktionalität und der systemnahen Software (wie Treiber, Schnittstellenverkehr) unter Berücksichtigung von Plattformen, Protokollen, Sprachen und Sprachumfang
- Zusammenfassen und Dokumentieren der bisherigen Ergebnisse und Einschätzen, ob weiterer Spezifikationsbedarf besteht – nicht jedes Detail ist in dieser Phase vollständig spezifizierbar; eine Nachbearbeitung oder Festlegung gewisser Anforderungsdefinitionen während des Entwurfsprozesses ist durchaus möglich

3.1.3.2.2 Kompetenzfelder: Erarbeiten der Anforderungsdefinitionen

Fähigkeiten/Fertigkeiten

- allgemeine Anforderungsdefinitionen systematisch und vollständig erarbeiten sowie notwendige Informationen durch gezieltes Nachfragen oder Recherchieren aufnehmen können
- Vertraulichkeit und Datensicherheit herstellen können
- mit Kunden konstruktiv zusammenarbeiten können und das eigene Vorgehen kontextbezogen motivieren können
- rechtliche Grundlagen, Produktstandards und technische Normen bei der Ausarbeitung der Spezifikation berücksichtigen können
- auftragsbezogen Spezifikationsparameter priorisieren können (z. B. die Bedeutung von Wärmeverlusten oder Abmessungen etc.)
- Qualitätsfaktoren erkennen, in Hinblick auf ihre Relevanz für den Entwurfs- und Produktionsprozess bewerten und genau spezifizieren können
- über Implementierung von Funktionalität als Hard- oder Softwarekomponente unter Berücksichtigung wirtschaftlicher und funktionaler Kriterien entscheiden können (z. B. Kosten durch Verifikation und Test, Wiederverwendbarkeit, Plattformabhängigkeiten etc.)
- aufgrund der allgemeinen Anforderungen technologieentscheidende Kriterien erkennen, priorisieren und bewerten können; dabei preisliche Konsequenzen, Wirtschaftlichkeitskriterien und technologische Risiken einschätzen und berücksichtigen können
- mit Komplexität umgehen können, insbesondere wechselseitige (technische und technologische) Abhängigkeiten der verschiedenen Zielparameter erfassen können
- Optimierungsbetrachtungen anstellen können (abwägen können zwischen Entwurfsvorteilen/-risiken und dem Einsatz von Standardkomponenten, Konfigurierbarkeit vs. Geschwindigkeit, Stückzahl und Herstellungspreis etc.)
- technik- und technologiebezogene Sachverhalte präzise und kontextbezogen dem Kunden oder Vorgesetzten vermitteln können
- Implementierungsstrategie und Entwurfsumfang mit dem Kunden bzw. den eigenen Vorgesetzten abstimmen können; dafür – unabhängig vom eigenen Technologiefokus – Vor- und Nachteile eines anderen technologischen Vorgehens einschätzen und darstellen können
- technologieabhängige Spezifikationsbedarfe systematisch abklären können
- Architekturspezifikationen, -entwürfe erstellen und darstellen können

- Zielumgebung der zu entwickelnden Komponente, eigene Erfahrungswerte und Branchenkenntnisse bei Architekturspezifikation und Hard-/Softwarepartitionierung einbringen können
- Anforderungen an Daten- und Signalflüsse (Arten, Übertragungssicherheit etc.) erfassen und dokumentieren können
- Einsetzbarkeit von Standardkomponenten prüfen, abwägen und spezifizieren können unter Gewährleistung von Rechts- und Industrienormen sowie Investitionssicherheit
- funktionale Anforderungen, Parameterraum, Kontext von zu entwerfenden oder zu konfigurierenden Komponenten erfragen, abstimmen und verbindlich festlegen können
- Plattform-, Sprach- und Sprachumfangsentscheidungen treffen und abstimmen können
- Anforderungen an die Funktionalität der konfigurierbaren Komponenten erarbeiten können
- Informationen zur Integrationsumgebung der Komponente erfragen und daraus Anforderungen an systemnahe Software ableiten können
- Datenblätter/Produktangaben auswerten bzw. nötige Herstellerinformationen recherchieren können
- zum gegenwärtigen Zeitpunkt noch nicht festlegbare Details erkennen und explizit dokumentieren können
- erarbeitete Anforderungsdefinitionen vollständig und korrekt dokumentieren können
- mit Methoden des Dokumentenmanagements (z. B. Versionierung, Freigabe) umgehen können

Wissen

- Spezifikationsstandards
- Produktstandards, relevante Industrienormen (z. B. Sicherheitszulassung nach EN 60950, CE-Konformität)
- Branchen- und Technologietrends
- Architekturen und Komponenten in (konfigurierbaren) Hardware/Software-Systemen (insbesondere feldprogrammierbare Bauteile wie Speicher, FPGA, CPLD und dazugehörige Standardarchitekturen)
- Hardware/Softwarepartitionierung (Richtlinien, Vorgehen, Überblick über Methoden)
- Modul-, Benutzer-, Kommunikationsschnittstellen
- Halbleiter-, Speichertechnologien
- Technologien der Oberflächenmontage
- Klassifizierungen diskreter und programmierbarer Bauteile
- Einsatzbereiche von diskreten und integrierten (Medium Scale, Large Scale, Mikroprozessoren u. a.) Komponenten
- Standard-IC-Familien, IP-Cores sowie Rechtsrahmen für deren Einsatz (Lizenzen, Entwicklungsrechte etc.)
- Produktdatenbanken, -bibliotheken
- Bussysteme
- Übertragungsprotokolle
- branchenübliche Plattformen und Betriebssysteme sowie Programmiersprachen und möglicher Sprachumfang
- Datenschutz, Datensicherheit
- Dokumentenmanagement
- technisches Englisch

Werkzeuge/Methoden

- Fragenliste, strukturiertes Interview
- Tools zur Darstellung von Architekturentwürfen

3.1.3.2.3 Beispiel: Erarbeiten der Anforderungsdefinitionen

Da ein PC mit eingebautem FPGA-Entwicklungsboard schon vorhanden war und die nötige Flexibilität für die Schaltung garantierte, stand die Entscheidung für diese Technologie von vornherein fest. Zum Visualisieren der Ausgänge des ASIC-Modells für die Display-Ansteuerung bestand die Wahl zwischen der Verwendung eines diskret aufgebauten LED-Displays oder einer Softwaresimulation des Displays.

Die Architektur der Schaltung ist durch ihren Zweck schon teilweise festgelegt. Sie besteht zum einen aus einem Modell des ASICs 80C32a18 und zum anderen aus Komponenten, die die Umgebung des ASIC modellieren und Kommunikation mit dem Benutzer ermöglichen. Aus früheren Projekten konnten einige Komponenten wieder verwendet werden. Diese Komponenten erlauben dem Benutzer, über den PCI-Bus des PCs mit einem beliebigen Design-under-Test (DUT), das sich neben den wieder verwendeten Komponenten auf dem FPGA befindet, und dem vom DUT verwendeten Speicher zu kommunizieren.

Folgende Fragen zum Entwurfsumfang waren noch zu klären: Da der schon vorhandene Prototyp des Displays Komponenten enthielt, die dann auf den ASIC integriert werden sollten, musste entschieden werden, ob diese Komponenten auf dem FPGA modelliert werden sollen und das Display eine neue Schnittstelle erhält, oder ob das FPGA-Design die Komponenten nicht enthalten soll und das ASIC-Modell auf das FPGA und den Display-Prototypen aufgeteilt wird. Es musste entschieden werden, wie analoge Schaltungsteile und IP-Blöcke auf das FPGA abgebildet werden sollen. Anschließend war noch zu klären, wie das Modell auf dem FPGA debuggt werden kann.

Um festzustellen, ob mit schon vorhandener Software das ASIC-Modell über seine serielle Schnittstelle debuggt werden kann, fand ein Gespräch mit einem Mitarbeiter für Softwareentwicklung statt. Daraus ging hervor, dass diese Debug-Möglichkeit für die Register des verwendeten Prozessors realisierbar ist, aber mit einem sehr unsicher abschätzbaren Arbeitsaufwand.

In einem verbindlichen Gespräch mit den am Projekt Beteiligten wurde dann entschieden, dass alle Komponenten des ASICs so genau wie möglich auf dem FPGA modelliert werden sollen. Ein LED-Display sollte extern angeschlossen werden. Der Benutzer sollte Zugriff auf den Speicher des ASICs haben.

Alle weiteren Details wurden während des Schaltungsentwurfs spezifiziert.

Der Component Developer verfügt über umfangreiche Kenntnisse in den Bereichen diskreter sowie digitaler Schaltungsentwurf. Durch diese – insbesondere in seinem Spezialbereich – können selbstständig Entwurfs- bzw. Entwicklungsaufgaben wahrgenommen werden. Bei sehr komplexen Entwurfs-/Entwicklungszielen ist jedoch eine ggf. sehr enge Zusammenarbeit mit entsprechend spezialisierten Entwicklungsingenieuren üblich. Insbesondere ist auch wegen der Komplexität der zu verwendenden Tools eine starke Spezialisierung gängig.

Auf der Basis der Entwurfsstrategie wird der Entwurf schrittweise verfeinert. Die Anzahl der nötigen Rekursionen und Überarbeitungen kann sehr hoch sein.

3.1.3.3.1 Tätigkeiten: Beschreiben der Schaltung

- Auswerten der Spezifikation und Strukturieren sowie Priorisieren aller die Struktur und den Ablauf des Entwurfs entscheidenden Komponenten
- Planen eines effizienten Vorgehens/Auswählen bzw. Abstimmen der Entwurfsmethoden
- Strukturieren und Priorisieren nach physischen, funktionalen und architektonischen Kriterien und den daraus abgeleiteten Entwurfskomponenten
- Analysieren von Funktionalität und Verhalten und Umsetzen in formale Beschreibungen
- Entwerfen von Schaltungen unter Verwendung spezifischer CAE-Tools (Computer Aided Engineering)
- Verwenden spezifischer CAE-Tools (Synthesetools) zum Entwerfen von Verhaltensbeschreibungen und zum Durchführen von Struktursynthese und Technologieabbildung (beinhaltet auch einfache Testläufe, ob die Verhaltensbeschreibung korrekt und hinreichend vollständig für eine Verifikation gemäß Testspezifikation ist) – dieser Vorgang kann extrem komplex und Schwerpunkt des gesamten Referenzprozesses sein; dabei kommt es zu vielfachen Iterationen zwischen Beschreibung und Test der Beschreibung
- Prüfen und Optimieren des Entwurfs (siehe den entsprechenden Teilprozess!)
- Aufbauen der Testfälle (z. B. Zusammenfassung von Stimuli und Responses innerhalb einer Testbench)
- Einpassen digitaler Schaltungsentwürfe in die (konfigurierbaren) Zielbauelemente (Anpassung an Struktur und Anzahl logischer Blöcke) – in diesem Entwurfsschritt werden üblicherweise auch bereits die Programmierdateien für die konfigurierbaren logischen Bauelemente (durch die entsprechende Firmware) mit erstellt
- Einbeziehen von Entwurfsvorlagen beim Entwurf, dazu Beschaffen und Einbinden entsprechender Bibliotheken/Vorlagen
- Integrieren von Teilentwürfen in den Gesamtentwurf (dabei sind u. U. die Kommunikationssysteme oder deren Schnittstellen zu entwerfen bzw. zu implementieren)
- Einschätzen des Entwurfsstandes: in Bezug auf nötige Anpassungen der Test-/Spezifikation; in Bezug auf hinreichenden Entwurfsstand für einen Prüflauf
- Prüfen der Verfügbarkeit/Umsetzbarkeit kritischer Komponenten (um eine reibungsfreie Beschaffung bzw. Herstellung zu gewährleisten)

3.1.3.3.2 Kompetenzfelder: Beschreiben der Schaltung

Fähigkeiten/Fertigkeiten

- entwurfsrelevante Kriterien vollständig, in ihrem systemischen Zusammenhang und ihrer Bedeutung aus der Spezifikation ableiten und priorisieren können
- Struktur des Entwurfs auf der Grundlage von aus der Spezifikation abgeleiteten Kriterien erstellen können
- Ablauf des Entwurfs unter Berücksichtigung vielfältiger Kriterien (wirtschaftlicher, zeitlicher, architektonischer, funktionaler) planen und systematisch verfolgen können
- dabei Entwurfsmethoden auswählen bzw. mit abstimmen können

- technologie- und herstellungsprozessbezogene Risiken erkennen und beim Entwurf berücksichtigen können
- mit CAE-Tools (Firmware, Schaltplaneditoren, Entwicklungsumgebungen etc.) arbeiten können und für diese Tools Skripte zur Teilautomatisierung von Entwurfsschritten (z. B. Syntheseläufen) erstellen können
- inhaltliche, in der Spezifikation beschriebene Anforderungen an Funktionalität und Verhalten analysieren und in formale Schaltungs- bzw. Verhaltensbeschreibungen umsetzen können
- mithilfe von Schaltplaneditoren Schaltungen entwerfen können (Umfang abhängig vom Arbeitsschwerpunkt)
- in einer Hardware-Beschreibungssprache das Verhalten eines Systems beschreiben können (Umfang abhängig vom Arbeitsschwerpunkt)
- Struktursynthese und Technologieabbildung unter Verwendung von CAE-Tools/Firmware durchführen können (Umfang abhängig vom Arbeitsschwerpunkt)
- benötigten Logikumfang („Logic Cells“) abschätzen sowie Programmlogik/digitale Schaltungen auf Zielbauelemente abbilden („fitten“) können
- Entwurfsvorlagen (allgemeine, tool- oder herstellerspezifische Bibliotheken) einbeziehen können und dabei allgemeine Rechtsbestimmungen (Copyrights, Lizenzbestimmungen u. Ä.) bei Beschaffung und Einsatz berücksichtigen können
- mit zum Herstellungsprozess gehörenden, vom jeweiligen Hersteller bereitgestellten Bibliotheken arbeiten können
- Entwurf iterativ und in Abstimmung mit dem aktuellen Entwurfsstand verfeinern können
- Teilentwürfe/Entwurfsvorlagen in Gesamtentwurf integrieren können
- nach Datenblättern, technischen Dokumentationen recherchieren und diese auswerten können
- Herstellerangaben, technische Kataloge etc. auswerten und mit Herstellern Verfügbarkeit und Umsetzbarkeit für den Entwurf kritischer Komponenten abklären können

Wissen

- Halbleiter- und Schaltungstechnologien und -trends
- digitale und analoge Schaltungstechnik
- analoge, digitale und gemischte Signalverarbeitung
- Hoch- und Höchstfrequenzschaltungen
- (Mikro-)Prozessorsysteme
- konfigurierbare und rekonfigurierbare Systemarchitekturen
- funktionales Chip-Design
- (computergestützter) Schaltungsentwurf für elektronische Komponenten, PLD-Design
- Vorgehen und Modelle zur Partitionierung in Hard- und Softwaresysteme
- Logiksynthese
- Entwurfsmethoden (z. B. Top-Down-Design, strukturierte Analyse)
- Board-Entwicklung (PCB – Printed Circuit Board –, Layout & Design)
- Beziehungen zwischen Entwurf-Test-Diagnose
- prüfgerechter Entwurf (z. B. Partitionierung, Prüfpfade, testorientierte Layout-Generierung)
- Modul-, Kommunikations-, Benutzerschnittstellen
- Betriebssysteme, Bussysteme, Protokolle
- Skript- und Hardware-Beschreibungssprachen (z. B. VHDL, Verilog)

- Simulation und Emulation von Softwaremodellen elektronischer Schaltungen
- Angebot, Möglichkeiten und Einsatzbereiche von Standardkomponenten (Standard-ICs, IP-Cores, Gatter-, Softwarebibliotheken)
- Einsatz, Aufbau und Einsatzbereiche programmierbarer Logik – z. B. CPLDs (Complex Programming Logic Devices), FPGAs, Speicherkomponenten
- Simulations-, Emulations-, Verifikationsmodelle und -methoden (→ Testbarkeit des Designs)
- Industrienormen und -standards

Werkzeuge/Methoden

- Entwurfsdatenbanken
- Firmware/CAE-Tools zum Schaltungsentwurf (analogen und/oder digitalen): Schaltplan-editoren, Simulatoren
- Entwicklungssuites für Hardware-Beschreibungssprachen einschließlich Tools zur Struktursynthese/Technologieabbildung und deren Verifikation
- Tools zum Erstellen von Testbenches und Testpatterns
- Tools für PCB Layout & Design (Editing, Polygon Pushing)
- Entwurfswerkzeuge für Hoch- und Höchstfrequenzschaltungen
- Verifikationsmethoden (z. B. Model-Checking, Equivalence-Checking, Timing-Analyse)
- Prüfmethode für das Layout (z. B. Design Rule Checks, Layout vs. Schematic Check)

3.1.3.3.3 Beispiel: Beschreiben der Schaltung

Die Hardware für die Emulation besteht hauptsächlich aus dem FPGA-Board und dem externen LED-Display. Ein entsprechendes LED-Display war schon vorher vom Projektpartner des PickToLight-Projekts hergestellt worden und konnte ohne Änderungen verwendet werden.

Aus den funktionalen Anforderungen an die Schaltung ergaben sich die Teilkomponenten, aus denen das FPGA-Design bestehen soll. Die wichtigsten Komponenten sind das Modell des ASICs (**D**evice **u**nder **T**est, DUT), ein Scan-Controller, der die Register des DUT seriell auslesen und beschreiben kann, einem programmierbaren digitalen Modell des LED-Displays und einer Kontrollschaltung, über die der Benutzer auf alle RAMs und programmierbaren Komponenten zugreifen kann. Diese Kontrollschaltung konnte aus früheren Projekten teilweise wieder verwendet werden. Für einige Bestandteile des ASICs, insbesondere die analogen Komponenten und die Speicher, mussten spezielle digitale Modelle entworfen werden.

Grundlage für das Modell des ASICs ist die synthetisierte Netzliste des ASICs. Diese Netzliste musste an die funktionalen Anforderungen der Emulation und die technischen Möglichkeiten des FPGAs angepasst werden. Dieser Vorgang wurde mit einem Skript automatisiert, damit verschiedene Versionen des ASIC-Entwurfs mit minimalem Aufwand emuliert werden können.

Um die Anforderungen an die übrigen Teilkomponenten festzulegen, wurden zunächst die Schnittstellen des DUT analysiert und für die Emulation festgelegt. Dann wurde ein Grobentwurf der Schaltung erstellt, in dem die Schnittstellen aller Komponenten definiert sind. Dazu mussten unter anderem Ein- und Ausgabemöglichkeiten für den Benutzer festgelegt werden und Zugriffe der Teilkomponenten auf gemeinsame Ressourcen (z. B. RAMs) geregelt werden. Nachdem der grobe Entwurf feststand, konnte jede Teilkomponente für sich entworfen werden. Bei Bedarf wurden die geforderte Funktionalität und die innere Architektur einer Komponente genauer spezifiziert, bis sie in VHDL implementiert werden konnte.

Die Schritte von der Festlegung der Schnittstellen bis zum Schreiben des VHDL-Codes wurden mehrfach iteriert, da sich die Anforderungen an die einzelnen Komponenten ge-

genseitig beeinflussen und manche wichtige Details oder Fehler erst während der Implementierung oder nach der Simulation bekannt wurden.

Nachdem alle VHDL-Dateien fehlerfrei simuliert werden konnten, wurde eine erste Synthese schrittweise durchgeführt. Die Befehle für das Synthesetool wurden gleichzeitig in einem Skript festgehalten. Auch bei diesem Schritt waren noch kleine Korrekturen am Quellcode nötig, weil einige VHDL-Konstrukte nicht vom Synthesetool akzeptiert wurden und weil Fehler im Timing erkannt wurden, die durch die Verwendung verschiedener Clocks auftreten.

3.1.3.4 Erstellen einer Testspezifikation

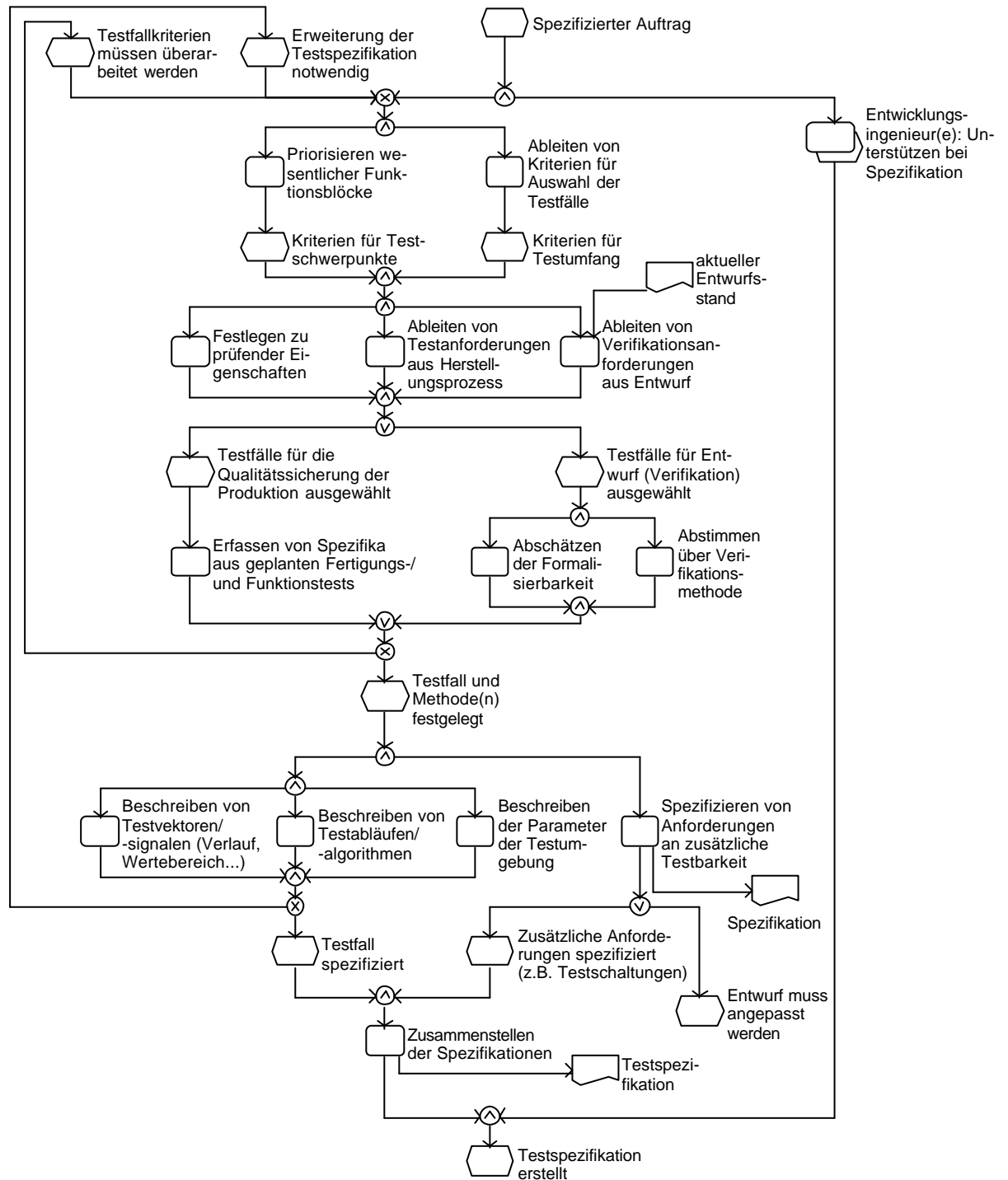


Abbildung 12: Erstellen einer Testspezifikation.

Für die Prüfung der Übereinstimmung von Spezifikation und Implementierung und die Verifikation der Entwurfsschritte sind konkrete Testfälle zu spezifizieren, ebenso für Tests zum Erkennen von Fertigungsfehlern (Produktionstests). Basis ist die Spezifikation. Die Menge aller möglichen Testfälle ist für jede Schaltung mit einer realen Komplexität zu groß, als dass diese alle geprüft werden könnten. Unter Berücksichtigung unterschiedlicher Kriterien (wie z. B. Priorität einzelner Funktionsblöcke für das Gesamtsystem, Testbarkeit und Effizienz, Formalisierbarkeit, Test- vs. Produktionskosten) ist eine funktional und ökonomisch sinnvolle Auswahl zu treffen. Für diese sind die Testbereiche und die dabei zu verwendenden Methoden zu spezifizieren. Dabei ist zu berücksichtigen, dass viele Details

der Produktionstests erst nach Vollendung des Entwurfs spezifiziert werden können, Verifikationsanforderungen jedoch schon vorher klar dargestellt sein müssen.

Dieser Prozess ist iterativ und erfolgt in enger Abstimmung mit beteiligten Entwicklungsingenieuren. Abhängig vom Einsatz diskreter bzw. konfigurierbarer Elektronik und deren Entwurfsumfang unterscheiden sich die zu verwendenden Testmethoden deutlich.

3.1.3.4.1 Tätigkeiten: Erstellen einer Testspezifikation

Die hier genannten Tätigkeiten können eine enge Zusammenarbeit mit ingenieurtechnischen Experten beinhalten. Gegebenenfalls wird darauf gesondert verwiesen.

- Priorisieren wesentlicher Funktionsblöcke auf der Basis der Spezifikation: Identifizieren Qualität und Erfolg sichernder Faktoren unter Berücksichtigung funktionaler und technologischer Gesichtspunkte und Priorisieren dieser unter Berücksichtigung des Gesamtzusammenhangs zum Festlegen von Testschwerpunkten
- Ableiten von Kriterien zur Bestimmung des Testumfangs (je nach Anteil der Eigenentwicklung, Möglichkeiten zur Formalisierbarkeit, Qualitätsanforderungen, theoretische und ökonomisch sinnvolle Testbarkeit u. a.)
- Festlegen der zu prüfenden Eigenschaften (funktionale und strukturelle Testüberlegungen)
- Ableiten von Testanforderungen aus herstellungs-/technologiebezogenen Spezifika (sowohl z. B. Halbleitertechnologien als auch Bestückungs-/Montagetechniken etc.)
- Ableiten von Verifikationsanforderungen aus dem Entwurf, insbesondere unter Berücksichtigung der jeweiligen Abstraktionsstufen Verhalten, Register-Transfer-Level, Netzliste) ; dabei Berücksichtigen von Entwurfsumfang, -schritten und Zieltechnologien
- Abstimmen der Details der Fertigungs- und Funktionstests mit Experten aus Produktion und Qualitätssicherung, diese bei der Erstellung von Testspezifikationen berücksichtigen
- Abschätzen der Formalisierbarkeit von Verifikationsschritten und dafür nötiger Aufwände, Ableiten dafür notwendiger Spezifikationen; Abstimmen der Verifikationsmethoden sowie Ableiten/Ermitteln von für deren Auswahl notwendigen Informationen (dies kann u. U. eine hoch spezialisierte Tätigkeit darstellen, die dann in Zusammenarbeit mit darauf spezialisierten Experten durchgeführt wird)
- Ermitteln aller für einen Test- bzw. Verifikationsfall relevanten Informationen, Spezifizieren derselben – die Details dieses Vorgangs hängen stark von Entwurfsumfang, verwendeten Technologien und den ausgewählten Methoden ab; dabei kann es auch hier zu einer Zusammenarbeit entsprechender Experten kommen
- Spezifizieren zusätzlicher Anforderungen, die sich aus der Ermöglichung von Testbarkeit ergeben (z. B. interne/externe Testschaltungen, explizites Zurverfügungstellen interner Signale/Zustände), Ableiten von Konsequenzen für den Entwurf sowie Dokumentieren dieser („Nachspezifikation“)
- Zusammenstellen und Dokumentieren der (Test-)Spezifikationsergebnisse

3.1.3.4.2 Kompetenzfelder: Erstellen einer Testspezifikation

Fähigkeiten/Fertigkeiten

- komplexe System- und Wirkungszusammenhänge erkennen und berücksichtigen können
- technologie- und funktionsbedingte Risikofaktoren erkennen und priorisieren können; dabei Erfahrungen aus früheren Entwurfs- und Produktionsprojekten/-phasen einbeziehen können
- eigenen Informationsbedarf erkennen, nötiges – insbesondere entwicklungs- und produktionsbezogenes – Know-how durch Austausch und Befragen beteiligter Experten (z. B. Entwicklungs-, Verfahreningenieuren) zusammentragen können
- eigenen Standpunkt konstruktiv und systematisch darstellen und vertreten können

- Kriterien für Testumfang zur Qualitätssicherung bei Entwurf und Produktion unter Abwägung wirtschaftlicher und projektbezogener Rahmenbedingungen ableiten können; dazu Optimierungsüberlegungen anstellen können
- zu prüfende Eigenschaften festlegen können: sowohl im Hinblick auf spezifizierte Funktionsparameter als auch unter Berücksichtigung struktureller Überlegungen (Verwendung einfacher Fehlermodelle)
- technologie- und herstellungsprozessbezogene Risiken erkennen, ein- und abschätzen können
- auf der Basis der Entwicklungsziele Verifikationsetappen zur Sicherung des Entwurfs-/Entwicklungsprozesses ableiten können
- Zusammenspiel von Entwurf/Entwicklung und Verifikation/Test erfassen und gestalten können
- sinnvolle Testfälle auswählen und in Abstimmung mit Dokumentationsanforderungen sowie Qualitätssicherung darstellen können
- Formalisierbarkeit/Automatisierbarkeit von Verifikations- und Testabläufen einschätzen und ggf. in Zusammenarbeit mit Experten nötige Spezifikationen erarbeiten können
- Verifikationsmethoden abstimmen und für die Auswahl entscheidende Informationen ableiten bzw. ermitteln können (nötigenfalls in Zusammenarbeit mit Fachexperten)
- Vor-, Nachteile sowie Grenzen verschiedener Test- und Verifikationsmethoden kennen und berücksichtigen können
- alle für einen Test- bzw. Verifikationsfall relevanten Informationen ermitteln und spezifizieren können (insbesondere Verläufe, Wertebereiche, Constraints für Testsignale/-vektoren, Abläufe, zu verwendende Algorithmen und Fehlermodelle, Anforderungen an Parameter der Testumgebung)
- zusätzliche Entwurfsanforderungen, die sich aus der Forderung nach Testbarkeit ergeben, erkennen, abschätzen und in Spezifikation einfließen lassen können
- weiteren Spezifikationsbedarf einschätzen und formulieren können
- Testspezifikationen entsprechend den geltenden Dokumentationsstandards und Vorgaben der Qualitätssicherung dokumentieren können

Wissen

- Spezifikation von Testdesigns, -fällen, -verfahren
- Teststrategien, Testalgorithmen, Testsequenzen
- Möglichkeiten zur Verhaltens-, Funktionsbeschreibung
- Spezifikation und Dokumentation elektronischer Schaltungen/Komponenten
- Entwurfsprozess integrierter digitaler Schaltungen
- Entwicklungsprozess elektronischer Komponenten
- Fertigungsprozess und -abläufe für verwendete Technologien
- Simulationsmethoden (z. B. ereignisgesteuerte Simulation, Signalsimulation in Test-Suites, taktgenaue und taktübergreifende Simulationen)
- Analysemethoden (z. B. statische Timing-Analyse)
- Emulationsmethoden und Möglichkeiten der Umsetzung (z. B. In-Circuit-Emulation, FPGAs)
- formale Verifikationsmethoden (Equivalence-, Model-Checker)
- Fehlermodelle, strukturelle Testmustererzeugung, Fehlersimulation
- Technologie- und Produktionsrisiken
- Signalarten und Wertebereiche
- digitale und analoge Signalverarbeitung

- digitale und analoge Messgerätetechnik
- Prüf-/Testautomaten, Teststände
- grundlegende Abläufe und Verfahren der Qualitätssicherung (z. B. Belastungstests, Leiterplattentests)
- Dokumentationsstandards

Werkzeuge/Methoden

- Checklisten

3.1.3.4.3 Beispiel: Erstellen einer Testspezifikation

Verifikationsbedarf bestand für den VHDL-Code, für das Synthese- bzw. Layout-Ergebnis und für die fertig im Zielsystem (PC) integrierte Komponente. Weil alle benutzte Hardware wiederverwendet wurde und nur ein Einzelstück angefertigt wurde, mussten Produktionstests, wie sie für eine ASIC-Produktion oder Platinenherstellung mit diskreten Bauteilen nötig sind, nicht durchgeführt werden.

Alle neu entworfenen Komponenten wurden als VHDL-Entwurf in Simulationen umfangreichen Funktionstests unterworfen. Für die wieder verwendeten Komponenten wurden die zugehörigen Simulationsprogramme um die Szenarien, die sich durch Änderungen der Komponente ergeben haben, erweitert.

Da die Synthese- und Layout-Tools üblicherweise logisch korrekt arbeiten, aber Timing-Vorgaben nicht immer einhalten, war nach dem Layout eine Timinganalyse notwendig.

Beim Schreiben der Simulationsprogramme (Testbenches) wurden für jede Komponente alle zu prüfenden Szenarien festgelegt und dann die zugehörigen Eingangsdaten und die erwarteten Ausgangsdaten in einer VHDL-Testbench festgehalten. Die Entwicklung der Testbenches fand parallel zum Schaltungsentwurf statt. Da die Testbenches das Verhalten der zu testenden Komponente taktgenau überprüfen, war für das Schreiben der Testbenches eine genaue Kenntnis des von außen sichtbaren Verhaltens einer Komponente notwendig. Die Testbenches wurden teilweise von einem Mitarbeiter geschrieben, nachdem man sich über das spezifizierte Verhalten der Komponente und den Umfang der Testbench abgestimmt hatte.

Da ebenso wie die Schaltungsbeschreibungen auch die Testbenches Fehler enthielten, mussten sie nach einer Simulation bei Bedarf überarbeitet werden. Die fertige Testspezifikation lag dann als VHDL-Testbench vor.

3.1.3.5 Prüfen und Optimieren des Entwurfs

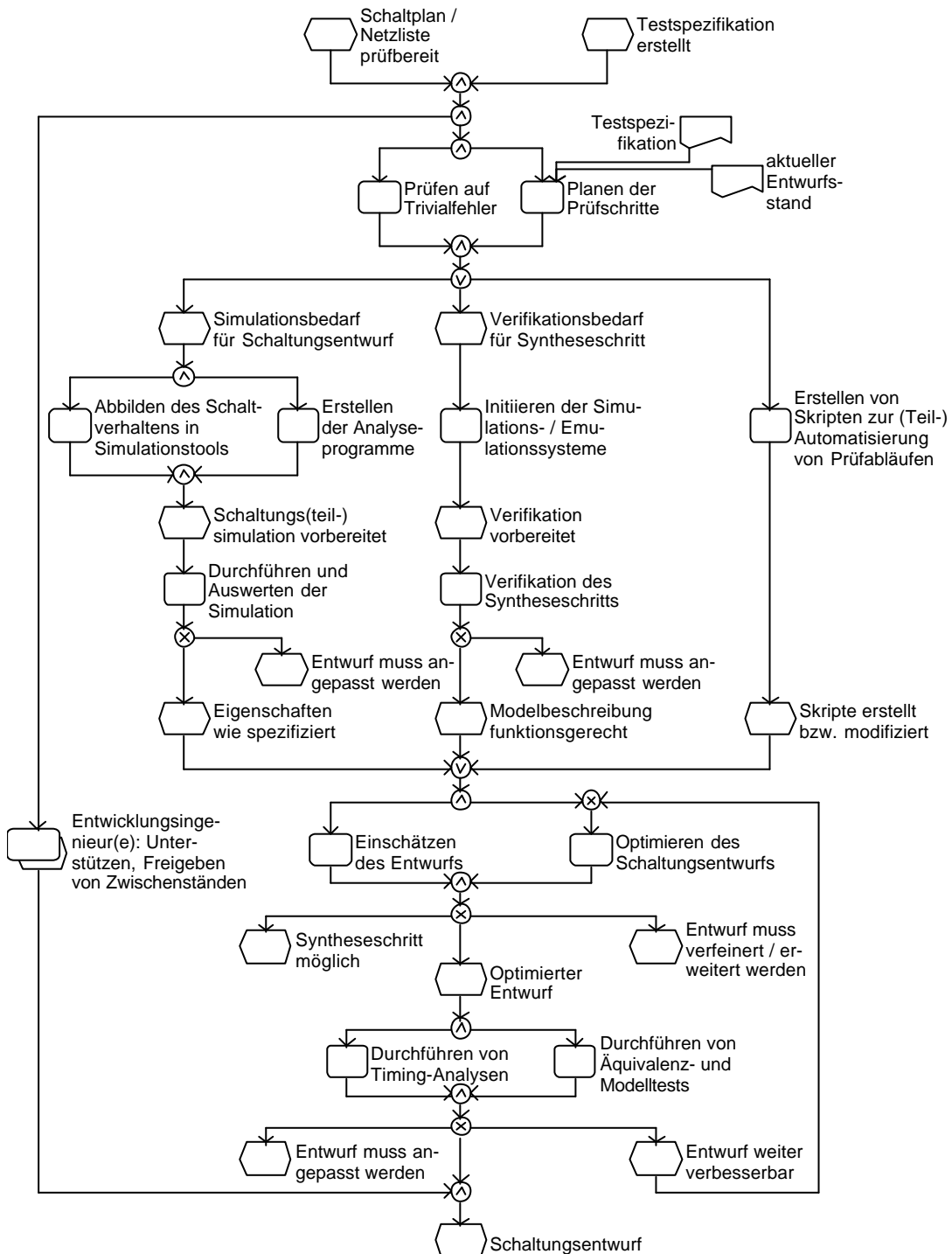


Abbildung 13: Prüfen und Optimieren des Entwurfs.

Die iterativen und rekursiven Entwurfsschritte werden hier immer wieder abgesichert, bevor innerhalb einer Funktionskomponente weiter verfeinert oder zu einer anderen übergegangen wird. Je nach Entwicklungsstand werden unterschiedliche Aktivitäten dieses Teilprozesses einen besonderen Schwerpunkt bilden. So ist es z. B. möglich, dass es für das Testen des Softwaremodells nur zu einer Simulation kommt und dann an der Beschreibung der Schaltung weitergearbeitet wird.

Einzelne hier aufgeführte Prozessschritte können – insbesondere bei einem hohen Entwicklungsumfang – sehr kosten- und ressourcenintensiv sein und ein hohes Maß an

speziellem Know-how voraussetzen. In solchen Fällen ist die Zusammenarbeit mit Entwicklungsingenieuren üblich.

3.1.3.5.1 Tätigkeiten: Prüfen und Optimieren des Entwurfs

- Prüfen auf Trivialfehler (d. h. Quellbeschreibung gegenlesen, auf Bezeichnungen, Zahlendreher, Zusammenfassungen bei Busleitungen u. Ä. achten)
- Planen der Prüfschritte – dabei sind die Vorgaben der Testspezifikation genauso zu berücksichtigen wie der aktuelle Entwurfsstand; die bei der Erarbeitung der Testspezifikation festgelegten Prüfmethoden sind anzuwenden; ggf. können erst jetzt mit den Detailinformationen aus dem Entwurf konkrete Prüfabläufe spezifiziert werden
- Abbilden des Schaltverhaltens zu simulierender Schaltungsfunktionen und Komponenten in geeigneten Simulationstools (unter Verwendung entsprechender Bibliotheken) und Erstellen der Analyseprogramme unter Verwendung tool- oder fachspezifischer (Skript-) Sprachen
- Durchführen und Auswerten der Simulation (durch Abfahren der Analyseprogramme sowie Simulation oder tatsächlicher Bereitstellung von Eingangssignalen)
- Initiieren der Simulations-/Emulationssysteme (ggf. in Zusammenarbeit mit Ingenieuren, die hier ihren Arbeitsschwerpunkt haben; insbesondere geht es um das Bereitstellen und Konfigurieren der Systeme und evtl. benötigter Schnittstellenmodule)
- Erstellen von Skripten (von einfachen MAKE-Skripten bis hin zu Skripten, die die Prüfsysteme selbst steuern)
- Verifikation des Syntheseschritts – dies bedeutet u. U. eine Zusammenfassung vielfältiger und umfangreicher Arbeitsschritte, die insbesondere das Kompetenzspektrum der auf die Entwicklung diskreter Elektronik ausgelegten Component Developer deutlich übersteigen kann
- Einschätzen der Prüfergebnisse und des Entwurfs sowie Vornehmen von Optimierungen des Entwurfs auf dieser Grundlage
- Durchführen von Timing-Analysen – der Test, ob die Komponenten die geforderten Geschwindigkeiten/Frequenzen haben, ist von besonderer Bedeutung; nach Festlegung der Verdrahtung wird der Test erneut durchgeführt (siehe „Die Entwürfe der Einzelkomponenten wurden mit Hilfe der zugehörigen Testbenches simuliert. Da die Schaltungsbeschreibungen und die Testspezifikationen als VHDL-Dateien vorlagen, mussten sie nur mit dem Simulationstool kompiliert werden und konnten dann simuliert werden. Nach einer fehlerfreien Simulation konnte der Entwurf synthetisiert werden.“)

Das Syntheseergebnis wurde nicht verifiziert, da man davon ausgehen konnte, dass das Syntheseergebnis logisch korrekt war. Eine genaue Timinganalyse sollte nur nach dem Layout stattfinden, weil die Timing-Informationen des Synthesetools sehr ungenau waren.

Im PickToLight-Projekt, dem Rahmenprojekt, fand allerdings eine umfangreiche Verifikation des Syntheseschritts statt. Innerhalb von PickToLight stellt die gesamte 80C32a18-Emulation die Vorbereitung dar, um das Syntheseergebnis zu emulieren. Zusätzlich wurde die Netzliste des ASICs mit den vorhandenen Testbenches für die HDL-Dateien simuliert. Dazu wurden zusätzlich zur Netzliste die Verhaltensmodelle der Technologiebibliothek in den Simulator geladen und dann die Simulationen anhand der erstellten Testbenches durchgeführt.

- Layouting und Routing“)
- Durchführen von Äquivalenz- und Modelltests

3.1.3.5.2 **Kompetenzfelder: Prüfen und Optimieren des Entwurfs**

Fähigkeiten/Fertigkeiten

- Prüfschritte systematisieren können (sowohl zur Berücksichtigung der Vorgaben aus der Testspezifikation und des aktuellen Entwurfsstands als auch für das eigene effiziente Vorgehen)
- Vorgaben aus der Testspezifikation und der Qualitätssicherung berücksichtigen und umsetzen können
- komplexe Zusammenhänge erkennen, systemisch denken und Wirkungsbeziehungen erfassen können
- mit Tools zur Schaltungssimulation arbeiten können – insbesondere Modellbibliotheken einsetzen und geeignete Modelle aus Bibliotheken zur Darstellung des Schaltverhaltens auswählen und einsetzen können sowie Simulationsanweisungen erstellen und ablaufen lassen können
- nötige Vorbereitungen – ggf. in Zusammenarbeit – für Konfiguration und Ablauf von Simulations- und Emulationsmethoden treffen und an entsprechend initiierten Simulations- und Emulationssystemen Syntheseschritte des Entwurfs verifizieren können
- Skripte – in Abstimmung mit verwendeten Betriebs- und Verifikationssystemen – erstellen und laufen lassen können
- Simulations- bzw. Prüfergebnisse dokumentieren, auswerten und einschätzen können
- Entwurfsstand einschätzen können, insbesondere bzgl. nötiger Anpassungen, möglicher Verbesserungen, und weiteres Vorgehen (Entwurfsreihenfolge, erweiterter Testbedarf ...)
- Möglichkeiten zur Optimierung des Entwurfs erkennen, Aufwand/Nutzen einschätzen und Verbesserungen umsetzen können
- einzelne Schaltungseigenschaften, insbesondere Frequenz-/Geschwindigkeitsverhalten (durch Timing-Analysen) testen sowie funktionale Übereinstimmung formaler Schaltungsbeschreibungen feststellen können (wichtig für Optimierungs- und Struktursyntheseschritte)

Wissen

Neben dem bereits unter 3.1.3.3.2 „Beschreiben der Schaltung“ und 3.1.3.4 „Erstellen einer Testspezifikation“ aufgeführten Wissen:

- Test-, Testprogrammentwicklung
- Skriptsprachen (sowohl auf Betriebssystem-/Shell-Ebene wie auch toolspezifische)
- System für IC-Test und -Analyse
- (ereignisorientierte) Signalerzeugung, -verfolgung und -analyse, insbesondere Timing-Analysen
- Simulationsmodelle für elektronische Komponenten, Modellbibliotheken
- Vorgehen bei und Methoden zur Schaltungsoptimierung

Werkzeuge/Methoden

- Firmware
- CAD/CAE-Tools, insbesondere Schaltplaneditoren und Simulationstools für analoge und digitale Schaltungen
- Emulationstools (z. B. „In-Circuit-Emulation“)
- Teststände, Prüfautomaten

3.1.3.5.3 Beispiel: Prüfen und Optimieren des Entwurfs

Die Entwürfe der Einzelkomponenten wurden mit Hilfe der zugehörigen Testbenches simuliert. Da die Schaltungsbeschreibungen und die Testspezifikationen als VHDL-Dateien vorlagen, mussten sie nur mit dem Simulationstool kompiliert werden und konnten dann simuliert werden. Nach einer fehlerfreien Simulation konnte der Entwurf synthetisiert werden.

Das Syntheseergebnis wurde nicht verifiziert, da man davon ausgehen konnte, dass das Syntheseergebnis logisch korrekt war. Eine genaue Timinganalyse sollte nur nach dem Layout stattfinden, weil die Timing-Informationen des Synthesetools sehr ungenau waren.

Im PickToLight-Projekt, dem Rahmenprojekt, fand allerdings eine umfangreiche Verifikation des Syntheseschrittes statt. Innerhalb von PickToLight stellt die gesamte 80C32a18-Emulation die Vorbereitung dar, um das Syntheseergebnis zu emulieren. Zusätzlich wurde die Netzliste des ASICs mit den vorhandenen Testbenches für die HDL-Dateien simuliert. Dazu wurden zusätzlich zur Netzliste die Verhaltensmodelle der Technologiebibliothek in den Simulator geladen und dann die Simulationen anhand der erstellten Testbenches durchgeführt.

3.1.3.6 Layouting und Routing

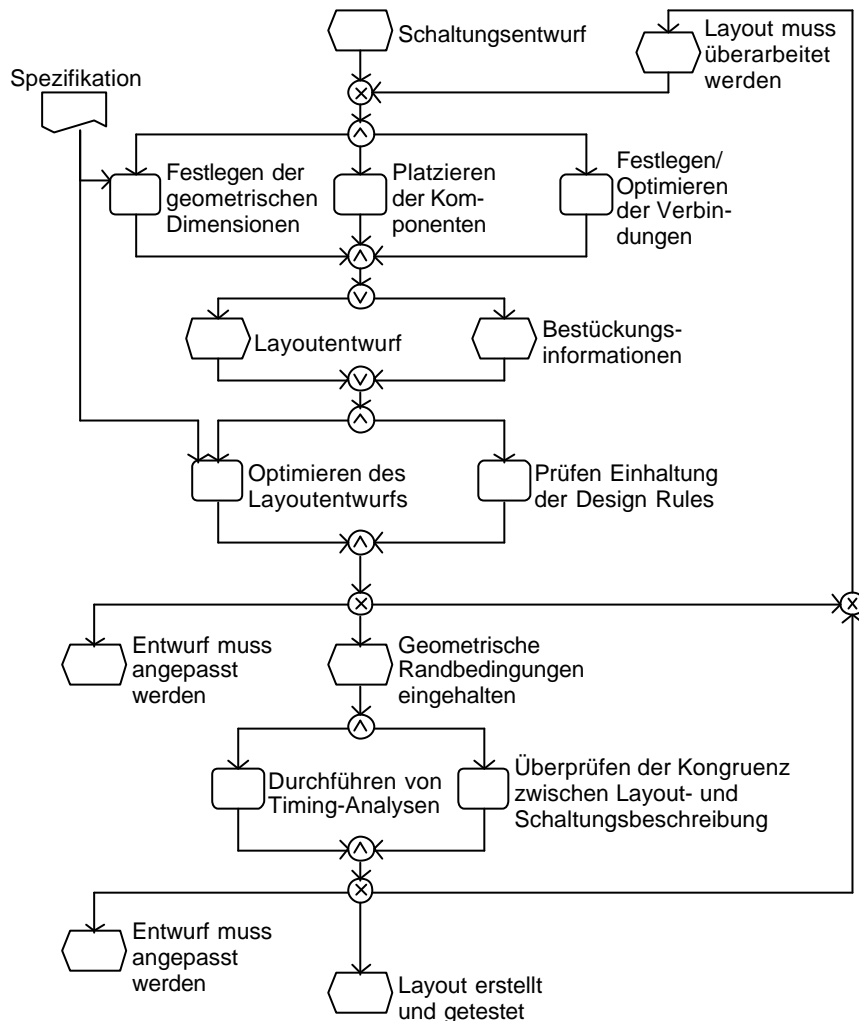


Abbildung 14: Layouting und Routing.

Der hier dargestellte Prozess gilt auf diesem Abstraktionsniveau für das Layout von Leiterplatten, das von Halbleiterchips und ebenso für die Abbildung von Schaltungsaspekten in programmierbare Bauteile (z. B. Implementierung der Schaltung in einem FPGA).

3.1.3.6.1 Tätigkeiten: Layouting und Routing

- Festlegen der geometrischen Dimensionen unter Berücksichtigung der Vorgaben aus der Spezifikation
- Platzieren der Komponenten und Festlegen/Optimieren der Verbindungen unter Berücksichtigung äußerer Randbedingungen (z. B. Pin-Outs) – diese Vorgänge sind nur teilautomatisierbar und erfordern neben Erfahrung immer wieder gezieltes Modifizieren und anschließendes Überprüfen; die Komponenten können Bauteile sein (diskreter Entwurf) genauso wie Gatter (digitaler Entwurf)
- Optimieren des Entwurfs (Minimierung der Fläche, Minimierung der Länge der Verbindungsbahnen zur Erhöhung der möglichen Arbeitsfrequenz)
- Prüfen auf Einhaltung der Entwurfsregeln (solche Design Rule Checks werden durch die Layout-Tools bereitgestellt)
- Durchführen von Timing-Analysen (das ist das zweite Mal: erst jetzt stehen mit den Verbindungslängen alle benötigten Informationen zur Berechnung der Signalverzögerungen zur Verfügung und kann die maximale Betriebsfrequenz realistisch abgeschätzt werden)

- Anwenden von Tests, ob die Schaltungsbeschreibung auf Layout-Ebene noch der Schaltungsbeschreibung auf der Entwurfsebene entspricht (u. a. sind Fehler, die auf der Layout-Ebene deutlich werden, optimalerweise auch gleich auf der Beschreibungs-/Entwurfsebene zu beheben)

3.1.3.6.2 Kompetenzfelder: Layouting und Routing

Fähigkeiten/Fertigkeiten

- mit entsprechenden CAE-Tools zur Layout- und Verbindungsfestlegung sowie Layout-Tests arbeiten können
- mit zum Herstellungsprozess gehörenden, vom jeweiligen Hersteller bereitgestellten Bibliotheken arbeiten können
- Spezifikation bzgl. Vorgaben zu Abmessungen und Platzierungen auswerten und diese Informationen im Layout umsetzen können
- dabei weitere äußere Randbedingungen (Gesamtfläche, Frequenzvorgaben) berücksichtigen und diese Parameter durch gezieltes, systematisches und erfahrungsgeleitetes Modifizieren optimieren können
- Design Rules berücksichtigen und in Abstimmung mit durch die Herstellungstechnologie und Qualitätssicherung bedingten Vorgaben umsetzen können
- zeitliche Randbedingungen/Signalverzögerungen simulieren bzw. darstellen können und auf dieser Grundlage durch Timing-Analysen Betriebsfrequenzen der Schaltung realistisch einschätzen können
- Layout auf funktionale Übereinstimmung mit Schaltungsbeschreibung/Gatternetzliste überprüfen können

Wissen

- Materialien, Technologien, Einsatzbereiche für/von Leiterplatten
- Herstellungsprozess für Leiterplatten und integrierte Schaltungen (Prozessschritte, Maskentechniken, Arbeit mit Schablonen etc.)
- Design und Layout von Leiterplatten (Printed Circuit Boards)
- Place & Route sowohl für Halbleiterfertigung von Chips als auch für programmierbare Logik
- Leiterplattentechnologien
- Halbleitertechnologien
- Bauformen von ICs, Standardprodukte und Logikfamilien relevanter Hersteller
- programmierbare Bauteile (Speichertechnologien, programmierbare Logik wie FPGA, Generic Array Logic (GAL), Complex Programmable Logic Devices (CPLD))
- Abbildung der Schaltung in Bauteile mit programmierbarer Logik (z. B. FPGA, PLD), Debugging
- Bestückungs- und Montagetechniken
- digitale, analoge und gemischte Signalverarbeitung (DSP, ASP, MSP – Digital, Analogous, Mixed Signal Processing)
- Datenformate und Standards zur Übertragung von Layout-Informationen

Werkzeuge/Methoden

- Layout-Editoren und (Auto-)Router (separat oder als Bestandteil komplexer CAE-Tools)
- Tools zum Testen auf Einhaltung der Entwurfsregeln und auf Kongruenz der Schaltungs- und Layout-Beschreibungen (gewöhnlich ebenfalls Bestandteil von den gesamten Entwurfsprozess abbildenden CAE-Tools/Entwicklungsumgebungen)

3.1.3.6.3 Beispiel: Layouting und Routing

Das Layout für das FPGA wurde vom Place-and-Route-Tool des FPGA-Herstellers durchgeführt. Zur Vorbereitung mussten die Pinbelegung des FPGAs und die Timing-Vorgaben dem Tool angegeben werden. Design-Rules wurden vom Tool automatisch geprüft.

Da die Timing-Vorgaben wegen der Verwendung mehrerer Clocks relativ komplex waren, musste jede Vorgabe im Anschluss an das Layout gesondert verifiziert werden.

Um das vorhandene LED-Display mit seinen eigenen Steckern mit dem FPGA-Board zu verbinden, musste noch eine Platine als Adapter entworfen werden. Deren Layout richtete sich nach der Form und Pinbelegung der Stecker des Displays.

3.1.3.7 Herstellen der Platine für den Prototypen

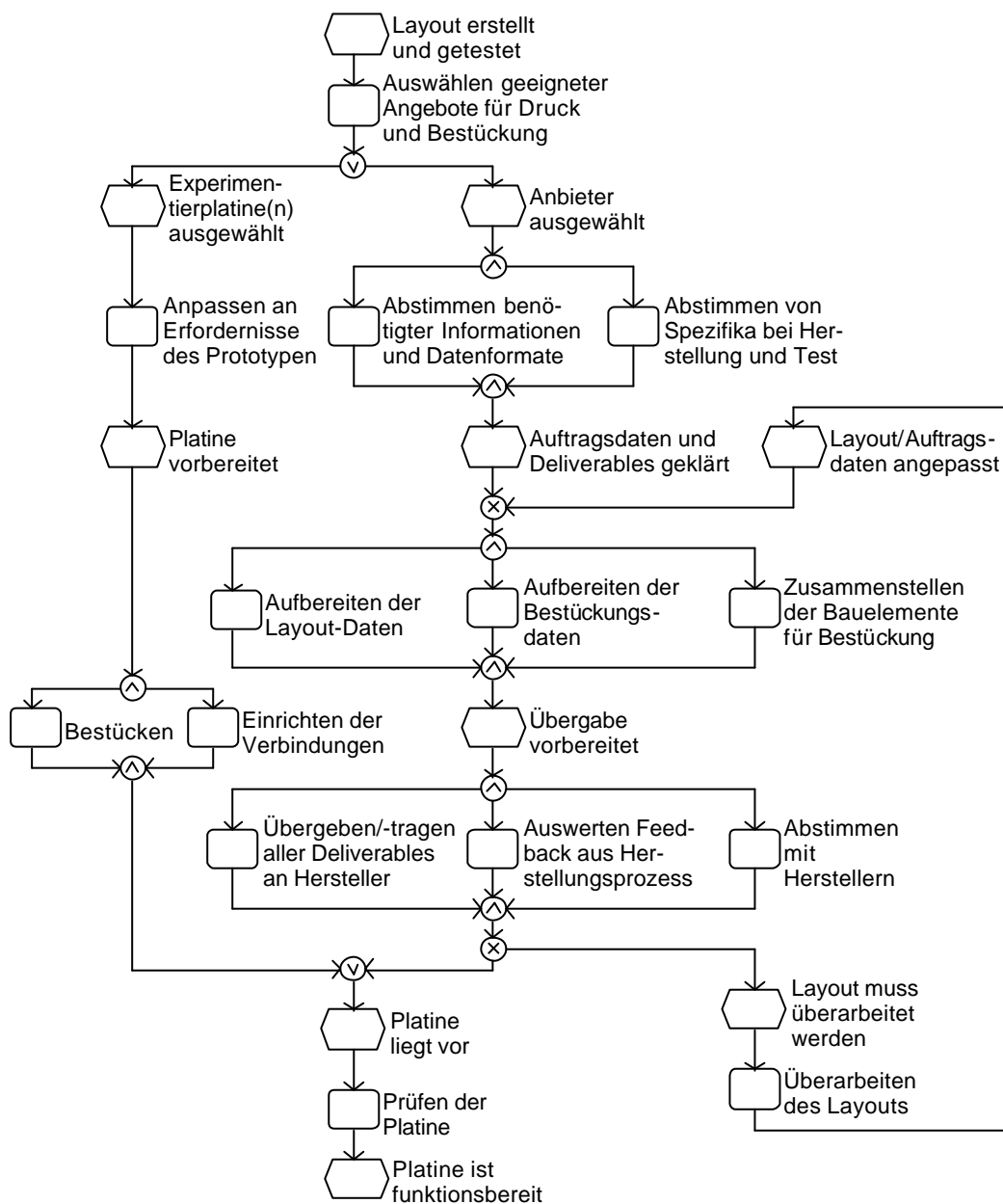


Abbildung 15: Herstellen der Platine für den Prototypen.

Der Aufwand zur Herstellung einer Prototypenplatine hängt stark vom Umfang des Einsatzes diskreter Bauelemente ab. Doch selbst bei einer reinen ASIC-Entwicklung ist der ASIC auf eine Platine zu integrieren. Der hier dargestellte Herstellungsprozess kann die Herstellung komplexer Boards mit mehrfachen Lagen bedeuten. Im einfachsten Fall ist auf einer standardisierten Platine (z. B. Euro-Platine) der Prototyp aufzubauen. (Dabei ist die Abgrenzung zum Teilprozess 3.1.3.10 „Bauen der Testhardware“ fließend.)

3.1.3.7.1 Tätigkeiten: Herstellen der Platine für den Prototypen

- Auswählen geeigneter Angebote für Druck und Bestückung – dabei sind neben preislichen Faktoren Kompetenzschwerpunkte der Anbieter (Zieltechnologien, Integrationsdichten, Qualitätssicherung, Lieferzeiträume etc.) zu berücksichtigen

- Abstimmen mit Hersteller (Auftragsdaten sowie Umfang und Formate benötigter Informationen (Maskendateien, Bohrpläne, Bestückungslisten), Möglichkeiten zur Datenübertragung u. Ä.)
- Abstimmen der Spezifika bei Herstellung und Test – wesentliche Informationen wurden schon beim Layout festgelegt (Anzahl der Lagen, Technologien zur Bestückung etc.); es sind noch Festlegungen zu treffen bzgl. der Auswahl des Materials, der Oberflächenbehandlung sowie bzgl. Arten und Umfang der durch den Hersteller durchzuführenden Tests
- Aufbereiten der Layout-Daten gemäß Anforderungen (üblicherweise Exportieren aus den CAE-/Layout-Tools in Standard-Formate, z. B. Extended Gerber-Format)
- Zusammenstellen der vom Bestücker zu montierenden Bauelemente
- Übergeben und Übertragen aller Informationen und Deliverables
- Auswerten des Feedbacks aus dem Herstellungsprozess (aus Produktion und Test, ggf. Einschätzen der Notwendigkeit einer Layout-Überarbeitung) – hier können wertvolle Informationen für den späteren Übergang in die Serienproduktion gewonnen werden
- Abstimmen mit dem Hersteller bei offenen Fragen/Problemen
- Anpassen von Experimentierplatinen an Erfordernisse des Prototypen (Größen, Bohrungen, Steckleisten, Verbindungsvorbereitung etc.)
- Bestücken der Experimentierplatine und Einrichten der Verbindungen
- Prüfen der fertig gestellten Platine auf (partielle) Funktionsbereitschaft

3.1.3.7.2 Kompetenzfelder: Herstellen der Platine für den Prototypen

Fähigkeiten/Fertigkeiten

- geeignete Anbieter auswählen, Angebote anfordern und preisliche Rahmenbedingungen in Abstimmung mit Vorgesetzten festlegen können
- benötigte Informationen, Formate und Übergabemodalitäten mit Herstellern abstimmen, Deliverables zusammenstellen, aufbereiten und nötigenfalls konvertieren können
- mit Mitteln der elektronischen Datenübertragung arbeiten können (z. B. FTP-, Web-Uploads, Datenbrennen auf CD-ROM)
- Bauelemente zusammenstellen und sachgerecht behandeln/verpacken sowie Zusammenstellung den Standards entsprechend dokumentieren können
- Feedback des Herstellers erfragen und auswerten können und dieses insbesondere zu Fragen evtl. nötiger Layout-Anpassungen bzw. für Rückschlüsse zur Qualitätssicherung der Serienproduktion einschätzen und priorisieren können
- Experimentierplatinen mechanisch bearbeiten (sägen, schneiden, bohren), Lötarbeiten unter Verwendung geeigneter Hilfsmittel durchführen sowie erforderliche Verbindungen fachgerecht herstellen können
- Platine auf auftragsgemäße und qualitätsgerechte Herstellung prüfen und (partielle) Funktionsbereitschaft feststellen können

Wissen

Neben dem bereits bei 0.2 „Die Entwürfe der Einzelkomponenten wurden mit Hilfe der zugehörigen Testbenches simuliert. Da die Schaltungsbeschreibungen und die Testspezifikationen als VHDL-Dateien vorlagen, mussten sie nur mit dem Simulationstool kompiliert werden und konnten dann simuliert werden. Nach einer fehlerfreien Simulation konnte der Entwurf synthetisiert werden.

Das Syntheseergebnis wurde nicht verifiziert, da man davon ausgehen konnte, dass das Syntheseergebnis logisch korrekt war. Eine genaue Timinganalyse sollte nur nach dem Layout stattfinden, weil die Timing-Informationen des Synthesetools sehr ungenau waren.

Im PickToLight-Projekt, dem Rahmenprojekt, fand allerdings eine umfangreiche Verifikation des Syntheseschrittes statt. Innerhalb von PickToLight stellt die gesamte 80C32a18-

Emulation die Vorbereitung dar, um das Syntheseergebnis zu emulieren. Zusätzlich wurde die Netzliste des ASICs mit den vorhandenen Testbenches für die HDL-Dateien simuliert. Dazu wurden zusätzlich zur Netzliste die Verhaltensmodelle der Technologiebibliothek in den Simulator geladen und dann die Simulationen anhand der erstellten Testbenches durchgeführt.

Layouting und Routing“ beschriebenem Wissen hier vor allem solches zur Abwicklung des Herstellungsprozesses:

- Angebotserfragung, Dokumentationspflichten bei Abwicklung einer Bestellung, Gewährleistungsansprüche
- Datenformate und -standards für Layout-Dateien
- Dateien und Verzeichnisstrukturen unter relevanten Betriebssystemen
- Protokolle und Tools zur Datenübertragung
- Kennzeichnung, Systematisierung/Katalogisierung elektronischer Bauelemente
- Bestückungspläne, -listen
- Handhabung elektronischer Bauelemente (z. B. zur Vermeidung von Zerstörungen durch elektrostatische Aufladungen)
- Arten und Standards bei Experimentierplatinen
- Verbindungstechniken (insbesondere Löten)
- technologiebedingte Risiken und Chancen bei Prototypen- und (Klein-)Serienfertigung

Werkzeuge/Methoden

- CAE-/Layout-Tools (für die Extraktion der benötigten Herstellungsdaten)
- Tools zum Datentransfer (FTP-, Web-Clients) und zum Erstellen von CD-ROM
- mechanische Werkzeuge (wie z. B. Bohrmaschine, Lötkolben)

3.1.3.7.3 Beispiel: Herstellen der Platine für den Prototypen

Für die 80C32a18-Emulation mussten in diesen Schritt nur einige Kabel an die Adapterplatine für das LED-Display gelötet werden. Die nötigen Steckerleisten waren auf der Platine schon angebracht. Im PickToLight-Projekt wurde für den Prototypen des ASICs eine Platine entwickelt. Diese Platine ist für Demonstrationszwecke verwendbar, dient aber vorrangig dem Test des Prototypen. Aus diesem Grund und weil das PickToLight-Projekt nur die Entwicklung des ASIC umfasst, fällt die Herstellung dieser Platine unter „Bauen der Testhardware“.

3.1.3.8 Beschaffen der Bauteile

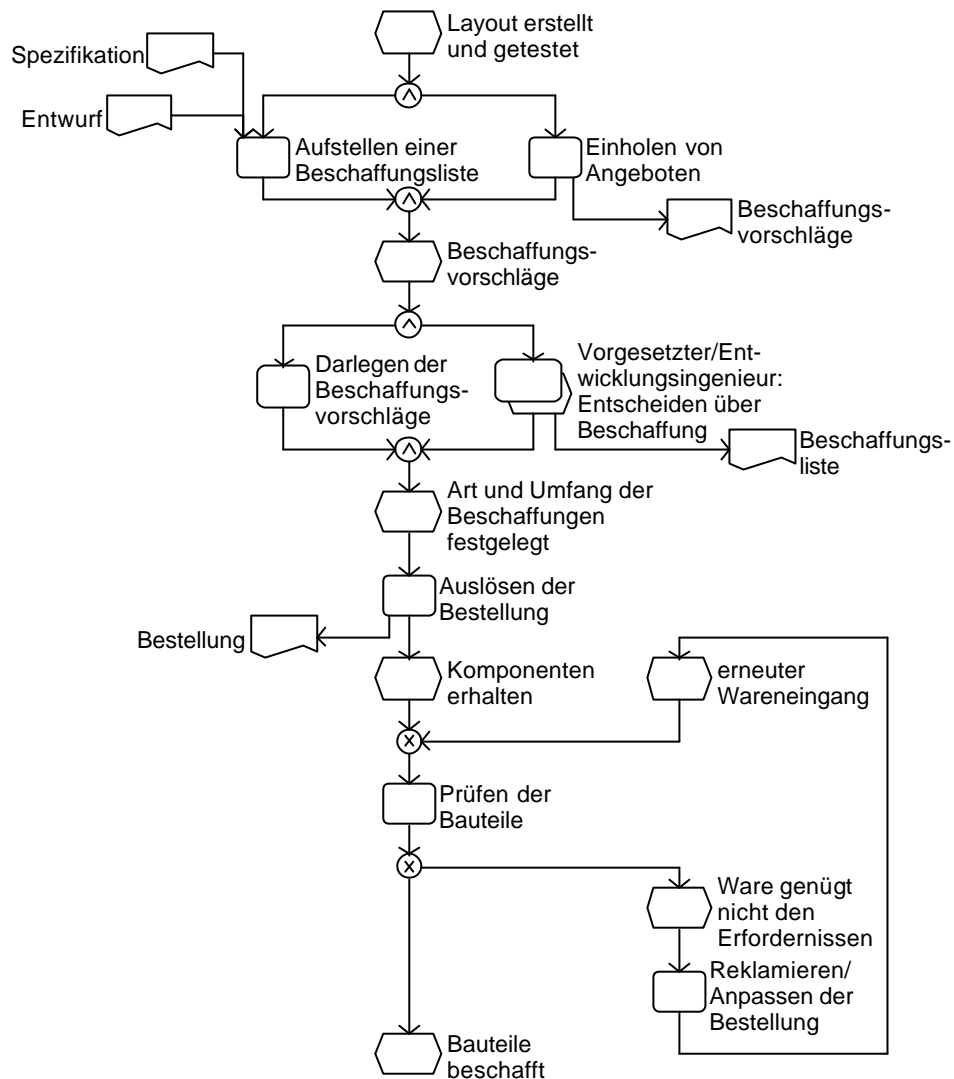


Abbildung 16: Beschaffen der Bauteile.

Je nach Größe und Organisation des Unternehmens, in dem der Component Developer arbeitet, wird die Beschaffung selbstständig oder in Abstimmung mit einem zentralen Einkauf geregelt.

3.1.3.8.1 Tätigkeiten: Beschaffen der Bauteile

- Aufstellen einer Beschaffungsliste – die Komponenten bzw. mögliche Ersetzungen sind durch Spezifikation und Entwurf vollständig vorgegeben
- Einholen von Angeboten (die Verfügbarkeit kritischer Komponenten wurde bereits während der Entwurfsphase geprüft)
- Darlegen der Beschaffungsvorschläge (in Bezug auf entwurfsbedingte und kaufmännische Faktoren) gegenüber Entscheidern
- Auslösen der Bestellung – entweder im eigenverantwortlichen Kontakt mit Herstellern oder durch den organisationseigenen Einkauf

- Prüfen der Bauteile (das beinhaltet sowohl Aufgaben aus der Wareneingangskontrolle als auch u. U. gezieltes Prüfen, ob kritische Bauteile den spezifizierten Anforderungen genügen)
- Reklamieren/Anpassen der Bestellung, dabei Abstimmen mit Herstellern/Lieferanten

3.1.3.8.2 Kompetenzfelder: Beschaffen der Bauteile

Fähigkeiten/Fertigkeiten

- Angebote einholen und mit Herstellern/Lieferanten Eignung der zu bestellenden Komponenten klären können
- technische Informationen, Produktangebote recherchieren und auswerten können (auch auf Englisch)
- Beschaffungsvorschläge/-liste den unternehmenseigenen Dokumentations- und Bestellstandards entsprechend zusammenstellen können
- Auswahl gegenüber Entscheidern darstellen, erklären und motivieren können und dabei deren technischen Kenntnisstand berücksichtigen können
- Bestellungen internen und kaufmännischen Standards entsprechend auslösen können
- eingegangene Ware auf Vollständigkeit und Funktionalität prüfen können

Wissen

- Einkaufsvorgänge, Garantie- und Ersatzansprüche
- Umgang mit Dokumenten/Dokumentationsstandards bei Beschaffungsvorgängen
- unternehmensspezifische Abläufe zur Beschaffung
- Kennzeichnung, Systematisierung/Katalogisierung elektronischer Bauelemente
- technisches Englisch

Werkzeuge/Methoden

- Produktdatenbanken
- Recherchertools

3.1.3.8.3 Beispiel: Beschaffen der Bauteile

Für die 80C32a18-Emulation mussten keine Bauteile beschafft werden.

3.1.3.9.1 Tätigkeiten: Erstellen der systemnahen und der Test-Software

- Abstimmen der Anforderungen bzgl. Kapselung und Ansprechbarkeit der Komponente
- Abstimmen aller für die Softwareerstellung relevanten Details zu Plattformen, Betriebssystemen und sonstigen physischen Spezifika in Bezug auf das Hostsystem und auf Systeme der Produktionsumgebung
- Erstellen von Treibern (Schnittstellen, I/O der Komponente); dabei Einbinden von proprietären Treibern
- Erstellen von Software zum Laden/Konfigurieren/Initialisieren der Chips und der gesamten Komponente; dabei Einbinden von Programmierdateien für die Logikbausteine
- Erstellen von Software zur iterativen Inbetriebnahme (temporäre Treiber, spezielle Loader-Programme u. Ä.) und zum Betrieb der Testhardware
- Erstellen von Software für Fertigungstests und zur Testautomatisierung
- Auswerten der Ergebnisse aus Inbetriebnahme und Test sowie Ableiten zusätzlicher Anforderungen an die bereitzustellende Software
- Erstellen von spezieller Software und Testdesigns zur Fehlersuche

3.1.3.9.2 Kompetenzfelder: Erstellen der systemnahen und der Test-Software

Fähigkeiten/Fertigkeiten

- Anforderungen aus Spezifikation und Gesprächen/Recherchen ableiten können
- Zusammenspiel zwischen verschiedenen Komponenten und dem Hostsystem erkennen und Anforderungen an Softwaregestaltung ableiten können
- Anforderungen und erstellte Programme den Standards entsprechend dokumentieren können
- technische Anforderungen in Programmentwürfe übersetzen und diese gezielt und strukturiert umsetzen können
- Datenaustausch auf hardwarenaher Ebene planen und mit geeigneten Programmstrukturen umsetzen können
- Treiber und Funktionsbibliotheken zur Steuerung der Komponente erstellen können und dabei herstellerspezifische Programme/Bibliotheken einbinden bzw. zielgerichtet modifizieren können
- einfache Programme zur Steuerung des Datenaustauschs bzw. zur Automatisierung von Test- und Fehlersuchläufen unter Verwendung einer Hochsprache erstellen können (der Component Developer ist kein Software Developer – es werden keine speziellen Fähigkeiten in Bezug auf Design und Engineering erwartet)
- Dateien zur programmierbaren Logik einbinden/implementieren können
- Software zur Steuerung der Testhardware erstellen können
- Anforderungen und Möglichkeiten zur Test-/Prüfautomatisierung erkennen und dafür geeignete Skripte und Software erstellen können (z. B. Tests von I/O-Patterns)
- Software in Komponente und Hostsystem einbinden und Integrationstests durchführen können
- Funktionalität einschätzen, Fehler erkennen, systematisch suchen und beheben können – nötigenfalls situationsbezogen Testdesigns und Fehlersuchprogramme erstellen können

Wissen

- Hochsprache (z. B. C oder C++)
- Skriptsprachen
- hardwarenahe Programmierung

- Betriebssysteme
- Rechnerarchitekturen, Daten- und Kommunikations-/Bussysteme
- Datenaustausch und Protokoll
- Schnittstellen

Werkzeuge/Methoden

- Editoren
- Programmier-/Entwicklungsumgebungen (Compiler, Debugger etc.)

3.1.3.9.3 Beispiel: Erstellen der systemnahen und der Test-Software

Die Entwicklung der Software konnte beginnen, nachdem die Schnittstellen der Teilkomponenten für die Hardware definiert waren. Die zu implementierenden Funktionen wurden zunächst genau spezifiziert und dann von einem Mitarbeiter für Softwareentwicklung implementiert.

Für den Betrieb der Komponente mussten Treiberfunktionen für die Hostseite geschrieben werden. Mit diesen Funktionen kann man auf die einzelnen Komponenten der Schaltung zugreifen. Sie basieren auf Treiberfunktionen des Herstellers des FPGA-Boards, mit denen man beliebige Daten an das FPGA senden und vom FPGA empfangen kann. Mit der Software des Herstellers wird auch das FPGA-Board initialisiert und das FPGA konfiguriert.

Für den Funktionstest des Gesamtsystems wurde ein Menüprogramm für den PC erstellt, mit dem die einzelnen zu testenden Funktionen aufgerufen werden können. Dieses Programm wurde ebenfalls von einem Mitarbeiter für Softwareentwicklung geschrieben. Dabei musste man die Anforderungen an das Testprogramm mehrere Male untereinander abstimmen.

3.1.3.10 Bauen der Testhardware

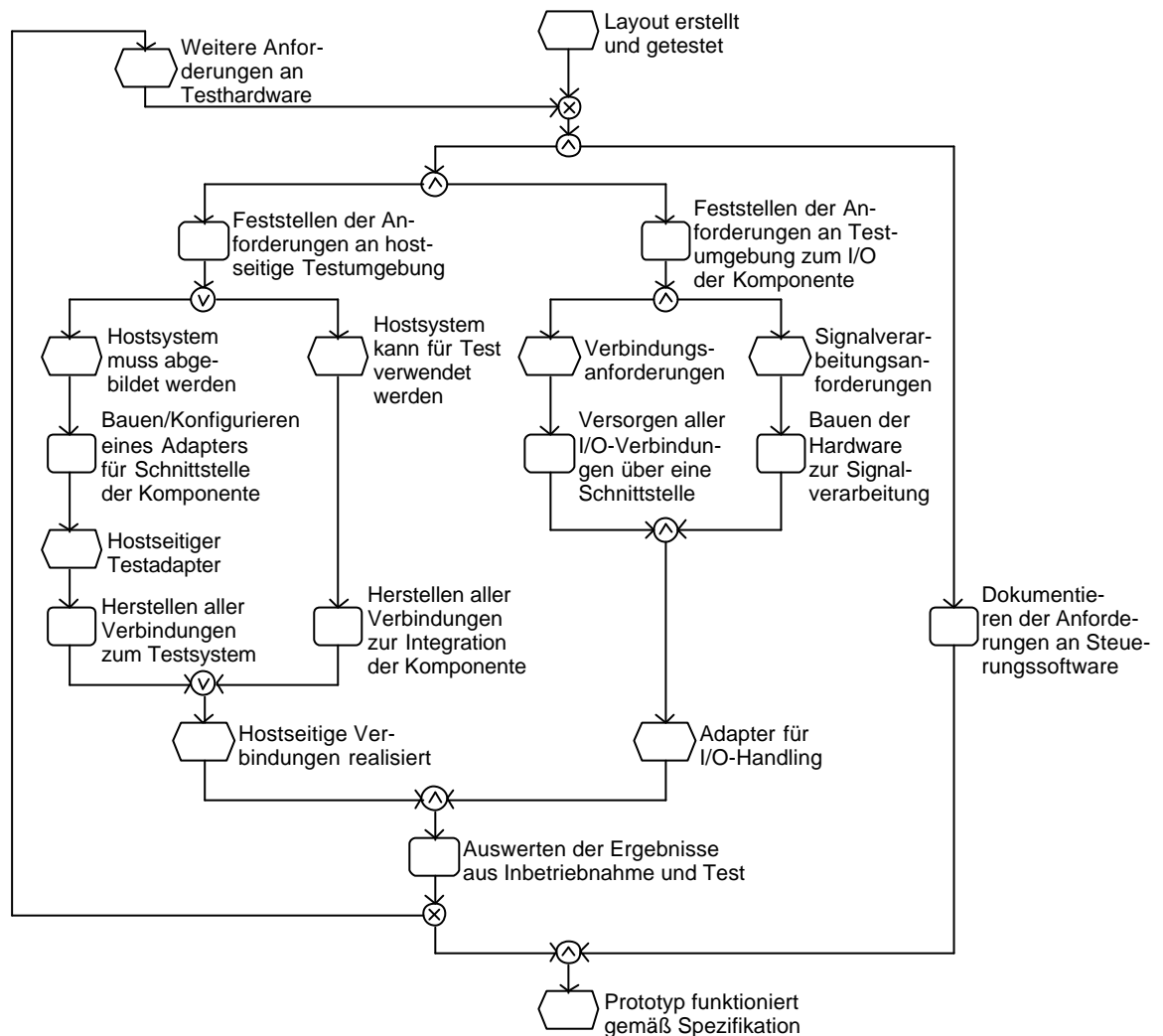


Abbildung 18: Bauen der Testhardware.

Je nach Situation und Anforderungen an Wiedereinsetzbarkeit/Konfigurierbarkeit kann das Bauen von Testhardware („Testadapter“) selbst den Umfang eines eigenen Projekts „Komponentenentwicklung“ annehmen. Möglich ist auch die Verwendung von marktüblichen Standardadaptern (wie z. B. Integrationsboards für FPGA oder solche für verbreitete Prozessoren wie die der 8051-Familie).

Geeignete Testhardware ist die Voraussetzung sowohl für das Prüfen der Komponente gegen die Spezifikation als auch für die Automatisierung von Funktionstests.

3.1.3.10.1 Tätigkeiten: Bauen der Testhardware

- Feststellen der Anforderungen an hostseitige Testumgebung – ist z. B. das Hostsystem ein PC, so ist auf diesem die Test-/Diagnosesoftware installierbar; steht das Hostsystem nicht zu Testzwecken zur Verfügung, muss über einen geeigneten Adapter der Zugang zu einem Steuerungssystem hergestellt werden
- Bauen eines individuellen bzw. Konfigurieren eines vorhandenen Adapters zur hostseitigen Anbindung der Komponente über deren jeweilige Schnittstelle (beinhaltet die Anpassung an die jeweiligen Signalanforderungen)
- Herstellungen der Verbindungen zur hostseitigen Integration/Testverbindung
- Feststellen der Hardware-Anforderungen zum Testen des I/O der Komponente, insbesondere in Bezug auf Signal- und Verbindungsanforderungen

- Versorgen/Entsorgen aller Input-/Output-Kanäle der Komponente über entsprechende Verbindungen und geeignete Zusammenfassung dieser Verbindungen
- Bauen der Hardware zur Signalverarbeitung/-wandlung
- Dokumentieren von Anforderungen an für die Testhardware zu erstellende Software
- Auswerten der Ergebnisse aus Inbetriebnahme und Test sowie Ableiten zusätzlicher Anforderungen an die bereitzustellende Testhardware

3.1.3.10.2 Kompetenzfelder: Bauen der Testhardware

Fähigkeiten/Fertigkeiten

- aus Anforderungen an Testbarkeit (bei Inbetriebnahme und für automatisierte Tests) Anforderungen für Testhardware feststellen, erarbeiten, zusammenstellen und priorisieren können
- Softwareanforderungen für die Testhardware erkennen und Dokumentationsstandards entsprechend spezifizieren können
- Aufwand/Nutzen zwischen dem Einsatz von Standardlösungen und der Selbstentwicklung einschätzen können
- Nutzbarkeit des Hostsystems für Testzwecke einschätzen können
- Möglichkeiten zur Signalerzeugung und Notwendigkeiten der Signalwandlung einschätzen und Hardwarebedarfe ableiten können
- Testhardware bauen können (da der Bau von Testhardware den Umfang einer Komponentenentwicklung annehmen kann, sind u. U. Ausschnitte aus allen Kompetenzen des gesamten Referenzprozesses gefordert)
- mechanische Verbindungen herstellen können (Löt-, Steck-, Klemmverbindungen usw.)
- mit Testsystemen und Messgerätetechnik umgehen können

Wissen

- Adapter, Verbindungstechnik
- Signalerzeugung, -verarbeitung, -wandlung
- analoge und digitale Messgerätetechnik
- Protokolle, Datenaustausch aus hardwarenaher Ebene
- Standardtestsysteme/-lösungen/-adapter für elektronische Bauelemente

Werkzeuge/Methoden

- mechanische Werkzeuge
- Adapter, Konverter, Verbindungswerkzeuge
- Signalgeneratoren
- analoge und digitale Messgeräte
- Mikroprozessorsysteme

3.1.3.10.3 Beispiel: Bauen der Testhardware

Weil sich die Schaltung am einfachsten im Zielsystem (PC) testen ließ, musste für die 80C32a18-Emulation keine Testhardware gebaut werden.

Der ASIC 80C32a18 allerdings wurde als Prototyp getestet. Im Zielsystem kommuniziert der ASIC über einen seriellen Bus mit dem Zentralrechner des Lagers. Diese Kommunikation sollte beim Prototypentest durch die serielle Schnittstelle eines PCs modelliert werden. Derselbe PC sollte auch andere Pins des ASICs für automatische logische Tests kontrollieren sowie den Speicher des ASICs programmieren können. Weiterhin sollte die

Schnittstelle des ASICs zum LED-Display mit einer Anzeige verbunden werden, und einige Pins des Prototypen sollten manuell kontrollierbar sein.

Diese verschiedenen Verbindungsanforderungen wurden auf einer Testplatine, die unter anderem mit der LED-Anzeige bestückt wurde, verwirklicht. Zur Durchführung von automatischen logischen Tests wurde die Platine mit dem FPGA-Board verbunden, auf dem wieder verwendbare Teile der Schaltung für die Emulation implementiert waren.

3.1.3.11 Programmieren der Bauteile

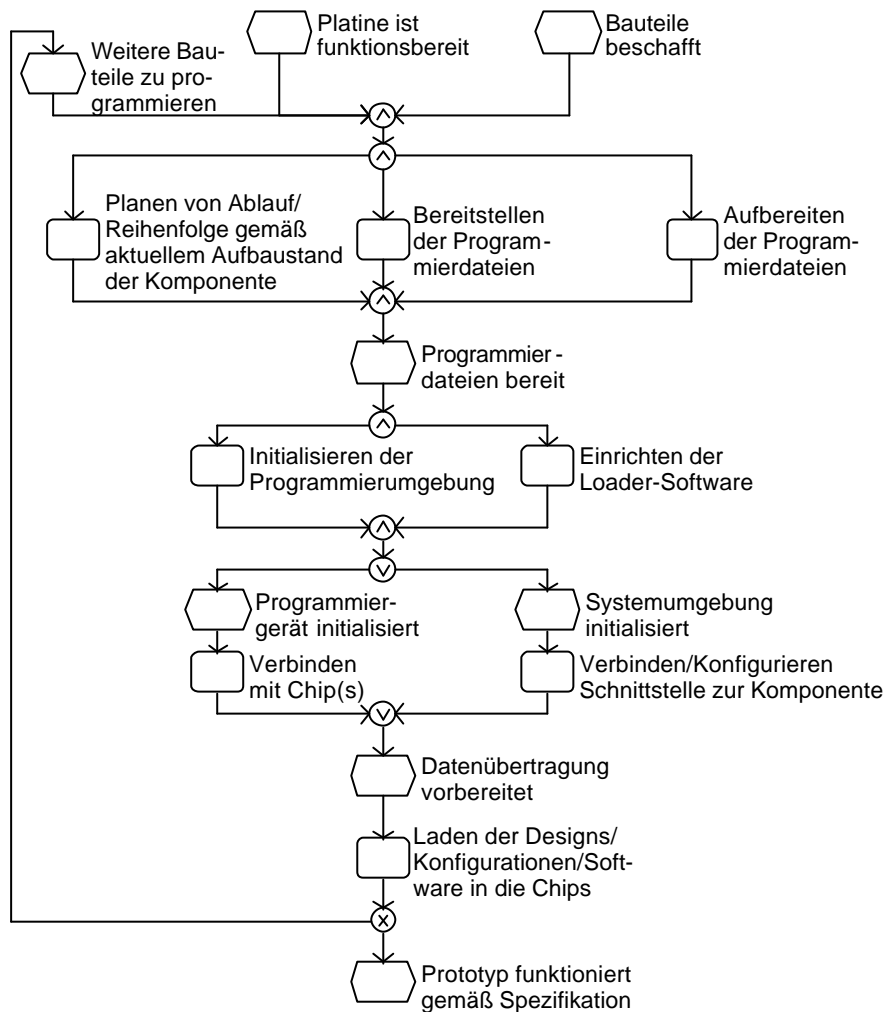


Abbildung 19: Programmieren der Bauteile.

Komplexität und Umfang dieses Teilprozesses variieren erheblich in Abhängigkeit von den verwendeten konfigurierbaren Komponenten und deren Verknüpfung (z. B. FPGAs in einer Daisy Chain). Möglich sind z. B. das Einbrennen eines BIOS in ein PROM, die Implementierung von Programmbefehlen in einen Flash-Speicher für einen Mikrocontroller ebenso wie die Programmierung eines CPLDs oder FPGAs. Die Bauteile können über entsprechende Programmiergeräte oder „in system“ programmiert werden. In letzterem Fall muss die Komponente bereits hinreichend aufgebaut und funktionsfähig sein (siehe „Iteratives Inbetriebnehmen und Testen der Baugruppen“!) Unter Umständen ist eine die initiale Programmierung steuernde Software vom Component Developer zur Verfügung zu stellen.

Die für die Programmierung nötigen Dateien wurden bereits beim Entwurf durch die Synthesetools bereitgestellt.

3.1.3.11.1 Tätigkeiten: Programmieren der Bauteile

- Planen des Ablaufs/der Reihenfolge zu programmierender Bauteile – dabei sind der aktuelle Stand der iterativ aufzubauenden Komponente sowie Einschränkungen infolge des Vorgehens beim Bestücken zu berücksichtigen
- Bereitstellen der Programmierdateien – üblicherweise werden beim Kompilieren der Schaltungsbeschreibungen diese durch die entsprechenden Tools unter Angabe von

Herstellerfamilien und Typ des Bauelements mit erstellt; Datenformate sind dabei parametrisierbar; ggf. müssen hier die Dateien noch einmal erstellt werden

- Aufbereiten der Programmierdateien – je nach Architektur der Komponente und nach Verknüpfung der logischen Bauelemente sind die Programmierdateien noch aufzubereiten; möglich ist ein Zusammenführen (Merge) oder die Einbindung in entsprechende Loader-Programme/Skripte
- Initialisieren der Programmierumgebung (spezielle Programmiergeräte, Host- oder den Host simulierende Systeme) und Einrichten der das Programmieren steuernden Software (Loader-Software)
- Herstellen aller nötigen (physischen) Verbindungen
- Einrichten der zur Programmierung nötigen Schnittstellen zur Komponente
- Übertragen der Daten in die Bauteile

3.1.3.11.2 Kompetenzfelder: Programmieren der Bauteile

Fähigkeiten/Fertigkeiten

- Arbeitsabläufe unter Berücksichtigung von Wechselabhängigkeiten planen können
- CAE-Tools/Firmware zur formatgerechten Bereitstellung der Programmierdateien konfigurieren und einsetzen können
- Programmierdateien an Architektur der Komponente anpassen können
- mit Programmierumgebungen umgehen sowie diese initialisieren/einrichten können
- Erfordernisse an zu erstellende Loader-Software erkennen, formulieren und implementieren können
- benötigte systemnahe Software (Loader, Teiber, Skripte etc.) erstellen, konfigurieren und einsetzen können
- physische Verbindungen herstellen können – dazu auch geeignete Verbindungstechniken einsetzen können
- Herstellerdokumentationen auswerten können
- Programmierdateien in die Chips übertragen können – dazu benötigte Ressourcen und Zeitbedarfe abschätzen können

Wissen

- Hardware-Beschreibungssprachen
- Design-Flow für die Abbildung digitaler Schaltungen in (re-)konfigurierbaren Bauelementen
- Bauformen von ICs, Standardprodukte und Logikfamilien relevanter Hersteller
- Aufbau programmierbarer Bauteile (Speichertechnologien sowie programmierbare Logik wie FPGA, GAL, CPLD)
- Programmier-/Entwicklungsumgebungen und typische Datenformate – insbesondere das JEDEC-Format (Joint Electronic Device Engineering Council)
- Programmiergeräte (Komponenten, Adapter, Schnittstellen, Arbeitsweise)
- Schnittstellen – insbesondere JTAG (Joint Test Action Group) und PCI (Personal Computer Interface) sowie LPT, COM, USB (Universal Serial Bus)
- Protokolle

Werkzeuge/Methoden

- Entwurfswerkzeuge für Bausteine mit programmierbarer Logik (z. B. Signal-, Schaltungssimulatoren, Editoren, Graphics-Editoren, Compiler, Synthesetools)
- Programmiergeräte und -umgebungen für Bausteine mit programmierbarer Logik

- PCs und andere Hostsysteme mit integrierten (Mikro-)Controllern

3.1.3.11.3 Beispiel: Programmieren der Bauteile

In diesem Schritt wurde das FPGA konfiguriert. Dies geschah während der Inbetriebnahme des Gesamtsystems.

Weil die Software des Herstellers des FPGA-Boards selbständig die Kommunikation mit dem Board herstellt und das FPGA mit beliebig auswählbaren Konfigurationsdateien programmieren kann, war dieser Schritt trivial.

3.1.3.12 Iteratives Inbetriebnehmen und Testen der Baugruppen

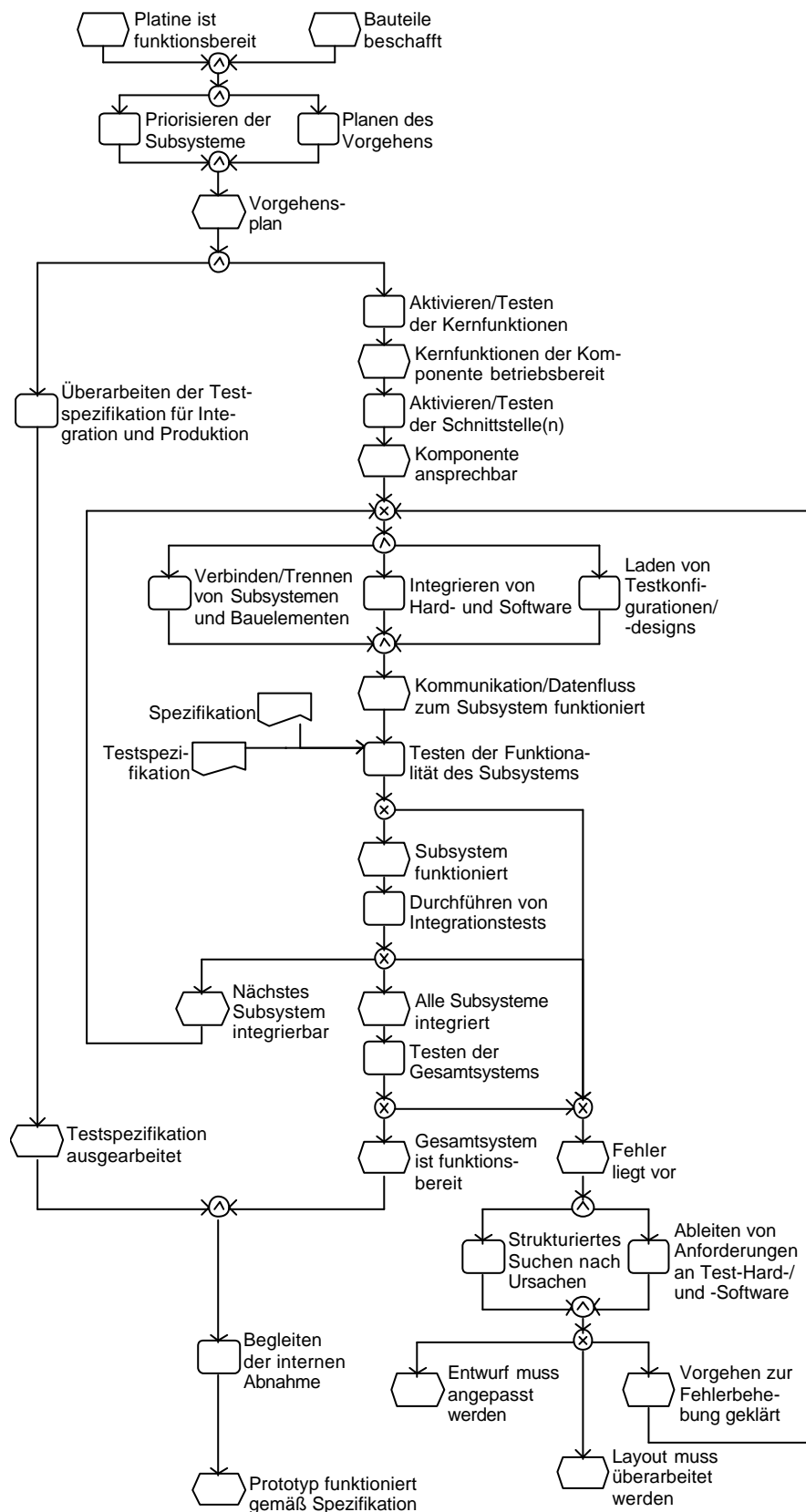


Abbildung 20: Iteratives Inbetriebnehmen und Testen der Baugruppen.

Der sukzessive Aufbau einer Komponente ist der klassische Weg, der jedoch in Abhängigkeit von den verwendeten Bauteilen und den jeweiligen Bestückungstechnologien nicht immer möglich ist.

3.1.3.12.1 Tätigkeiten: Iteratives Inbetriebnehmen und Testen der Baugruppen

- Planen des Vorgehens und Priorisieren der Subsysteme (funktionskritische bzw. teure Komponenten etc.) – die Reihenfolge der Inbetriebnahme orientiert sich am Daten-/Signalfluss
- Aktivieren und Testen der Kernfunktionen (z. B. Strom, Spannung, Takt/Synchronisation) sowie der Schnittstelle(n) (Zugriffssicherung auf die Komponente); erst nach Betriebsbereitschaft kann mit der Komponente weitergearbeitet werden
- Verbinden/Trennen von Subsystemen (sowohl physisch als auch softwaregesteuert, z. B. Zugriff auf die Pins eines Prozessors im JTAG-Modus, Benutzung von Testdesign zum (De-)Aktivieren von Teilsystemen etc.)
- Verbinden und Trennen von Bauelementen (in Abhängigkeit von verwendeten Bestückungstechnologien Abwägen von Zerstörungsrisiken bei Nachbestückung oder aber bei Inbetriebnahme der Gesamtkomponente)
- Integrieren von Hard- und Software (auch der Testsysteme): In-System-Programming, konfigurierbare Bauelemente, Initialisierungen etc.
- Laden von speziellen Testkonfigurationen/-designs für Fehlersuche oder partielle Inbetriebnahme
- Testen der Funktionalität von Subsystemen gemäß Spezifikation und Testspezifikation (beinhaltet das Dokumentieren der Testergebnisse)
- Durchführen von Integrationstests und Testen des gesamten Systems (einschließlich Test auf Einhaltung relevanter Standards, z. B. CE-Kennzeichnung, Störfestigkeit, Störaussendungen etc.)
- strukturiertes Suchen nach Fehlerquellen und nötigenfalls Ableiten von speziellen Anforderungen an Testhard-/software (z. B. Signalverfolgung, spezielle Designs für programmierbare Logik zur Fehlersuche, Auswählen geeigneter Betriebsmodi verwendeter Bauteile oder Subsysteme etc.) – eine Anpassung des Entwurfs zu diesem Zeitpunkt sollte nicht mehr erforderlich sein, evtl. lassen sich noch Layout-Fehler ausmerzen
- Überarbeiten der Spezifikation von Integrations- und Produktionstests (Nachspezifizieren auf der Grundlage der beim jetzt abgeschlossenen Entwurfsprozess, bei Produktion und Test gewonnenen Informationen und Erfahrungen)
- Begleiten – und evtl. Gestalten – der (internen) Abnahme

3.1.3.12.2 Kompetenzfelder: Iteratives Inbetriebnehmen und Testen der Baugruppen

Die Kompetenzfelder in diesem Abschnitt sind ein Querschnitt aus den bei den Teilprozessen zum Entwurf und in den Abschnitten 3.1.3.9.2 „Erstellen der systemnahen und der Test-Software“, 3.1.3.10.2 „Bauen der Testhardware“, 3.1.3.11.2 „Programmieren der Bauteile“ dargestellten Kompetenzen (inkl. Werkzeuge/Methoden). Ergänzend sind zu nennen:

Fähigkeiten/Fertigkeiten

- aus Architektur sowie Daten- und Signalflüssen Wechselbeziehungen zwischen den Teilkomponenten erfassen und auf dieser Basis Reihenfolge für Inbetriebnahme ableiten können
- Hard- und Software zusammenführen können (Installieren, Initialisieren und Konfigurieren: nach Power Up oder im laufenden Betrieb, extern und per In-System Programming), ggf. unter Zuhilfenahme zusätzlicher Tools
- mit analoger und digitaler Messgerätetechnik, Labortechnik wie Spannungsversorgungen, Signalgeneratoren/-wandlern etc. arbeiten können

- Zerstörungsrisiken bei Inbetriebnahme erkennen und diesen gezielt begegnen können
- Schnittstellen konfigurieren und in Betrieb nehmen können
- Teilsysteme und Gesamtkomponente gegen Spezifikation testen können (Funktions- und Integrationstests)
- Fehler erkennen, Fehlerursachen systematisch suchen können, dazu komplexe Schaltungszusammenhänge berücksichtigen können
- Fehlersuchstrategien erarbeiten und dafür ggf. spezielle Anforderungen an Testhard- und -software ableiten können (z. B. Testdesigns, Neutralisieren von Subsystemen, Testmuster, Betriebsmodi)
- Bauelemente montieren und demontieren können
- Abnahmeprozesse organisieren und durchführen können sowie ggf. mit Qualitätsmanagementsystemen umgehen können

Wissen

- Montagetechnologien
- Elektrostatik, elektromagnetische Verträglichkeit
- Debugging (Vorgehen, Methoden, Werkzeuge)
- Abläufe bei Funktions- und Integrationstests einschl. Testdokumentation
- Messverfahren
- Signalerzeugung und -wandlung
- induktive und deduktive Methoden der Fehlersuche
- Industrie- und Technologienormen (z. B. EN 61000-x-x zur Störfestigkeit und Störausendung u. a.)
- Abnahmeprozesse, Qualitätsmanagement, Dokumentationspflichten
- Qualitätssicherung, insbesondere interne Abnahmen/Reviews

Werkzeuge/Methoden

- mechanische Werkzeuge zur De-/Montage von Bauelementen
- Diagnosetools, Debugger
- Fehlersuchmethoden (z. B. Checklisten, Störfallanalysen, Fehlerbäume)

3.1.3.12.3 Beispiel: Iteratives Inbetriebnehmen und Testen der Baugruppen

Dieser Punkt überschneidet sich stark mit dem nächsten Teilprozess, da die Komponente sofort in ihrem Zielsystem getestet wurde.

Weil das FPGA-Board und das LED-Display früher schon oft verwendet worden waren, konnte man davon ausgehen, dass diese Komponenten fehlerfrei funktionieren. Kritisch waren das neu entworfene FPGA-Design, das Zusammenspiel mit der Host-Software und die Schnittstelle zur Platine mit dem LED-Display.

Da das Display nur über das ASIC-Modell auf dem FPGA angesteuert werden kann, wurde als erstes das bloße FPGA-Design mit Hilfe der Hersteller-Software für das FPGA-Board getestet. Danach wurde das LED-Display angeschlossen, und es wurden nacheinander alle Funktionen der Hardware und Software überprüft. Alle Tests wurden dokumentiert.

3.1.3.13 Unterstützen bei Integration und Test im Zielsystem

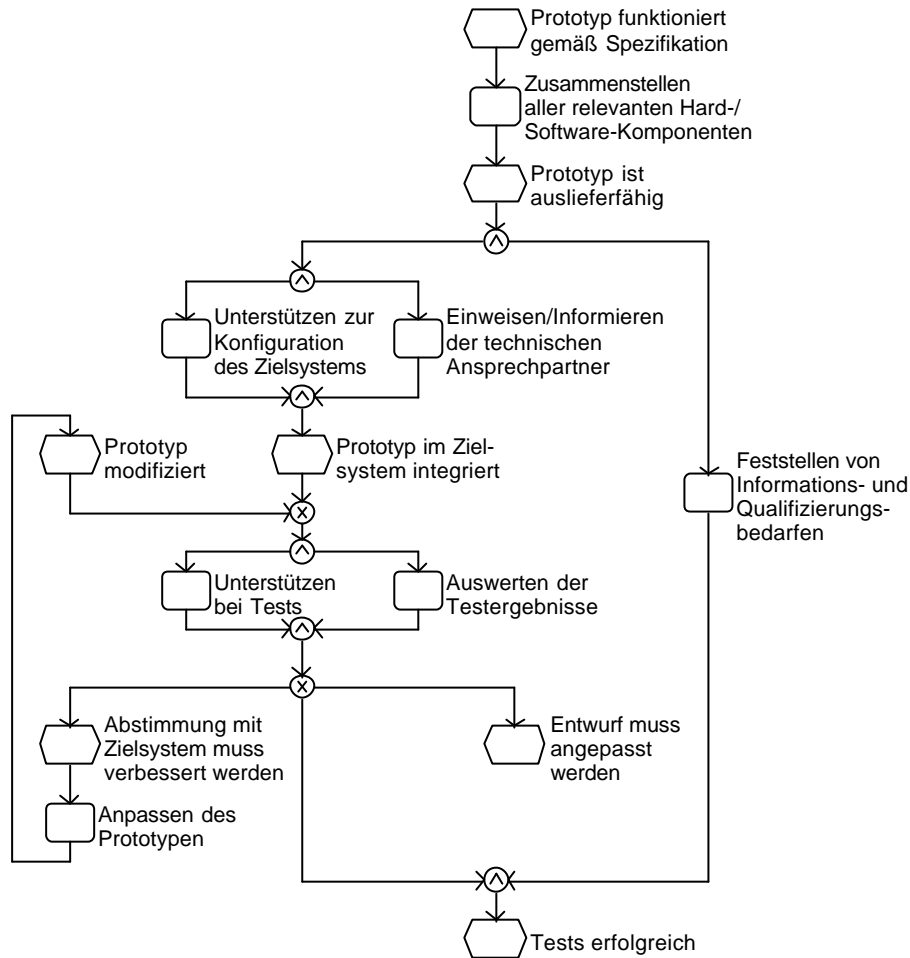


Abbildung 21: Unterstützen bei Integration und Test im Zielsystem.

Vom Component Developer werden sichere Kenntnisse im Bereich „IT-System“ erwartet. Für die Integration kann es zu einer Zusammenarbeit mit Spezialisten der jeweiligen Zielumgebung kommen, insbesondere wenn das Zielsystem Bestandteil einer sehr speziellen Umgebung ist.

3.1.3.13.1 Tätigkeiten: Unterstützen bei Integration und Test im Zielsystem

- Zusammenstellen aller relevanten Hard-/Softwarekomponenten (einschließlich für Tests benötigter Komponenten) – die formale Auslieferung als Bestandteil der Übergabe erfolgt im Teilprozess „Übergeben der Komponente“; existiert das Zielsystem (z. B. ein PC) vor Ort beim Component Developer, kommt es u. U. zu diesem Zeitpunkt noch nicht zu einer Auslieferung
- Unterstützen bei Konfiguration des Zielsystems (das kann auch die Mitarbeit bei der Einrichtung einer Testumgebung innerhalb des evtl. im Produktivbetrieb arbeitenden Zielsystems bedeuten)
- Einweisen/Informieren der technischen Ansprechpartner (insbesondere bzgl. Schnittstellen, Inbetriebnahme und Umgang mit der Komponente)
- Erkennen und Dokumentieren der Informations- und Qualifizierungsbedarfe
- Unterstützen bei (Integration-)Tests und Auswerten der Ergebnisse (ob tatsächlich ein Anpassen des Entwurfs notwendig ist, die Abstimmung mit dem Zielsystem verbessert werden muss – z. B. Toleranzen –, ob sich Hinweise auf notwendige Anpassungen für

die Produktion von Kleinserien ergeben oder aber der Prototyp erfolgreich integriert werden kann)

- Anpassen des Prototypen, um den Anforderungen aus Integration bzw. evtl. geplanter Kleinserienproduktion gerecht zu werden (der prinzipielle Entwurf selbst steht dabei nicht infrage)
- Erkennen bzw. Ableiten von Hinweisen auf Informations- und Qualifizierungsbedarfe aus allen kundenseitigen Kontakten (ebenfalls für die Zusammenstellung der Dokumentation)

3.1.3.13.2 Kompetenzfelder: Unterstützen bei Integration und Test im Zielsystem

Fähigkeiten/Fertigkeiten

- Beteiligte informieren bzw. von diesen Informationen (zu Details des Zielsystems etc.) einholen können
- Arbeitsabläufe mit beteiligten Spezialisten koordinieren können
- Regelungen zum Arbeitsschutz umsetzen können
- Auslieferungen vorbereiten können (Zusammenstellung, Setup-Wizards etc.)
- technische Ansprechpartner klar und gezielt unter Berücksichtigung von deren Kenntnissen und Informationsbedarfen einweisen/informieren können
- Informations- und Qualifikationsbedarfe erkennen und formulieren können
- Integrationstests planen, durchführen bzw. begleiten sowie auswerten können
- Testergebnisse hinsichtlich Tauglichkeit und Modifikationsbedarf (Umfang, Ebene: Entwurf oder Abstimmung mit Zielsystem) einschätzen können

Wissen

- Vorgehen und Rechtsnormen bei Auslieferungen
- Integrationstests (Vorgehen, Methoden)
- Fehlerdokumentation
- Change Request Management
- Arbeitsschutz, Gefahren beim Umgang mit elektrischen Geräten

Werkzeuge/Methoden

- Setup Wizards/Installer Shields
- Fehlerdatenbanken

3.1.3.13.3 Beispiel: Unterstützen bei Integration und Test im Zielsystem

Da die Schaltung sofort im Zielsystem getestet wurde, wurde dieser Teilprozess schon während des „iterativen Inbetriebnehmens und Testens der Baugruppen“ durchgeführt.

3.1.3.14 Erstellen der Nutzer- und Produktionsunterlagen

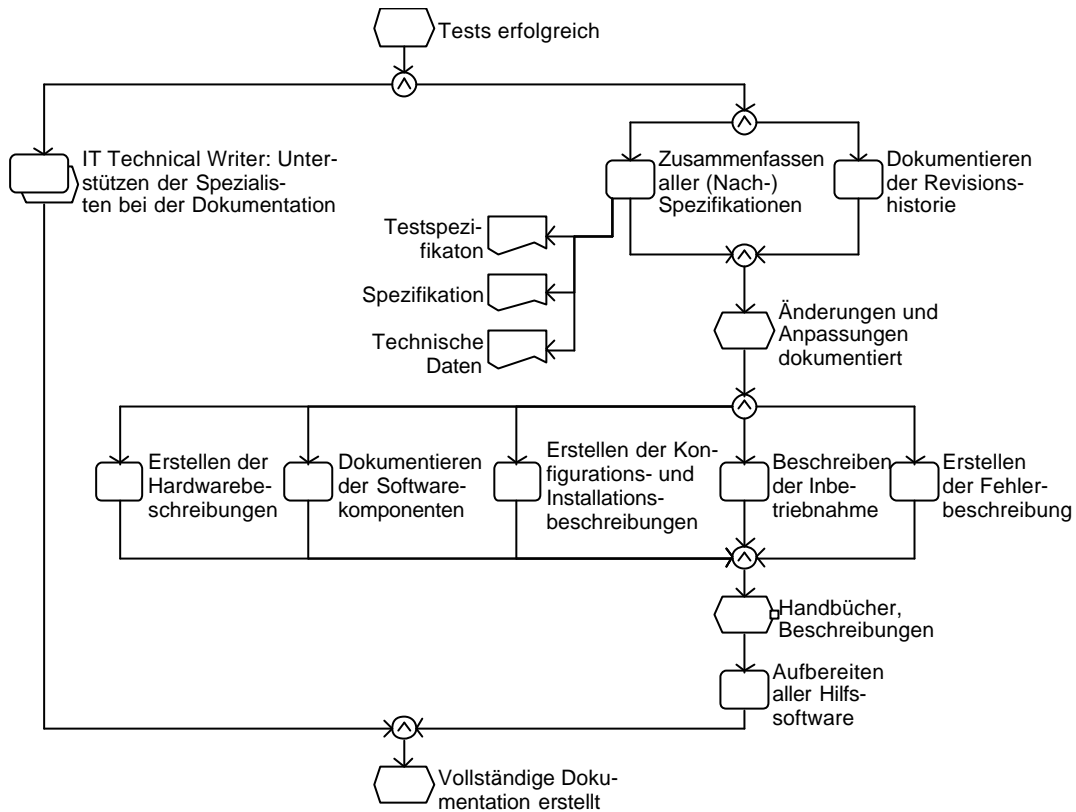


Abbildung 22: Erstellen der Nutzer- und Produktionsunterlagen.

Die Erstellung der Dokumentationsmaterialien erfolgt u. U. – in Abhängigkeit vom erforderlichen Umfang, potenziellen Anwendern der Komponente usw. – unter Beteiligung des Spezialisten IT Technical Writer.

3.1.3.14.1 Tätigkeiten: Erstellen der Nutzer- und Produktionsunterlagen

- Zusammenfassen aller im Verlauf des Entwicklungs- und Prototyp-Prozesses nachträglich spezifizierten Merkmale und Herstellen eines einheitlichen, konsistenten Dokumentstandes
- Dokumentieren der Revisionshistorie – u. U. unter Berücksichtigung spezieller Anforderungen seitens der Qualitätssicherung des Kunden
- Erstellen von Hardwarebeschreibungen (Adressen, Registersets, I/Os etc.)
- Dokumentieren der Softwarekomponenten (Code, eingesetzte Werkzeuge, Bibliotheken etc.)
- Erstellen von Konfigurations- und Installationsbeschreibungen (Setup, Treiber, Umgang etc.)
- Beschreiben der Inbetriebnahme
- Erstellen von Fehlerbeschreibungen (Hard- und Software) sowie möglicher Fehlerquellen und -behandlungen
- Aufbereiten mitzuübergabender Hilfssoftware (je nach Vereinbarung: Software aus Test, Fehlersuche, iterativer Inbetriebnahme; das technische Personal des Kunden kann dadurch wertvolle zusätzliche Informationen zum Umgang mit der Komponente bekommen)

3.1.3.14.2 Kompetenzfelder: Erstellen der Nutzer- und Produktionsunterlagen

Fähigkeiten/Fertigkeiten

- Dokumentationsstandards sowie spezielle Dokumentationsanforderungen des Kunden umsetzen können
- Dokumentationsstand zur Wahrung der Rechtssicherheit herstellen können
- Dokumentationspflichten aus Auftrag/Pflichtenheft erkennen und erfüllen können
- ggf. sich mit einem IT Technical Writer abstimmen, die Zusammenarbeit gestalten und erforderliches Material zusammenstellen und vorbereiten können

Wissen

- Dokumentationsstandards, -pflichten
- technologische und Branchenstandards
- Einsatz von Dokumenten zur Qualitätssicherung
- technische Spezifikationen
- Dokumentation von Softwarekomponenten
- Aufbau von Gebrauchsanweisungen
- Kriterien für nutzerfreundliche Dokumentationen

Werkzeuge/Methoden

- Tools zum Dokumentenmanagement (einschl. Versionskontrolle)
- Tools zum Zeichnen/Erstellen von graphischen Darstellungen

3.1.3.14.3 Beispiel: Erstellen der Nutzer- und Produktionsunterlagen

Eine umfassende Dokumentation über die einzelnen Hardwarekomponenten mit ihren Schnittstellen und über alle Softwarefunktionen wurde erstellt. Als Grundlage dazu diente die Spezifikation, die im Laufe des Entwurfsprozesses verfeinert worden war. Weiterhin wurden genaue Anweisungen zur Inbetriebnahme und Hinweise zur sinnvollen Benutzung der Software-Funktionen in die Dokumentation aufgenommen. Damit die Schaltung später bei Bedarf modifiziert werden kann, wurde auch der Designflow für das FPGA-Design dokumentiert.

3.1.3.15 Übergeben der Komponente

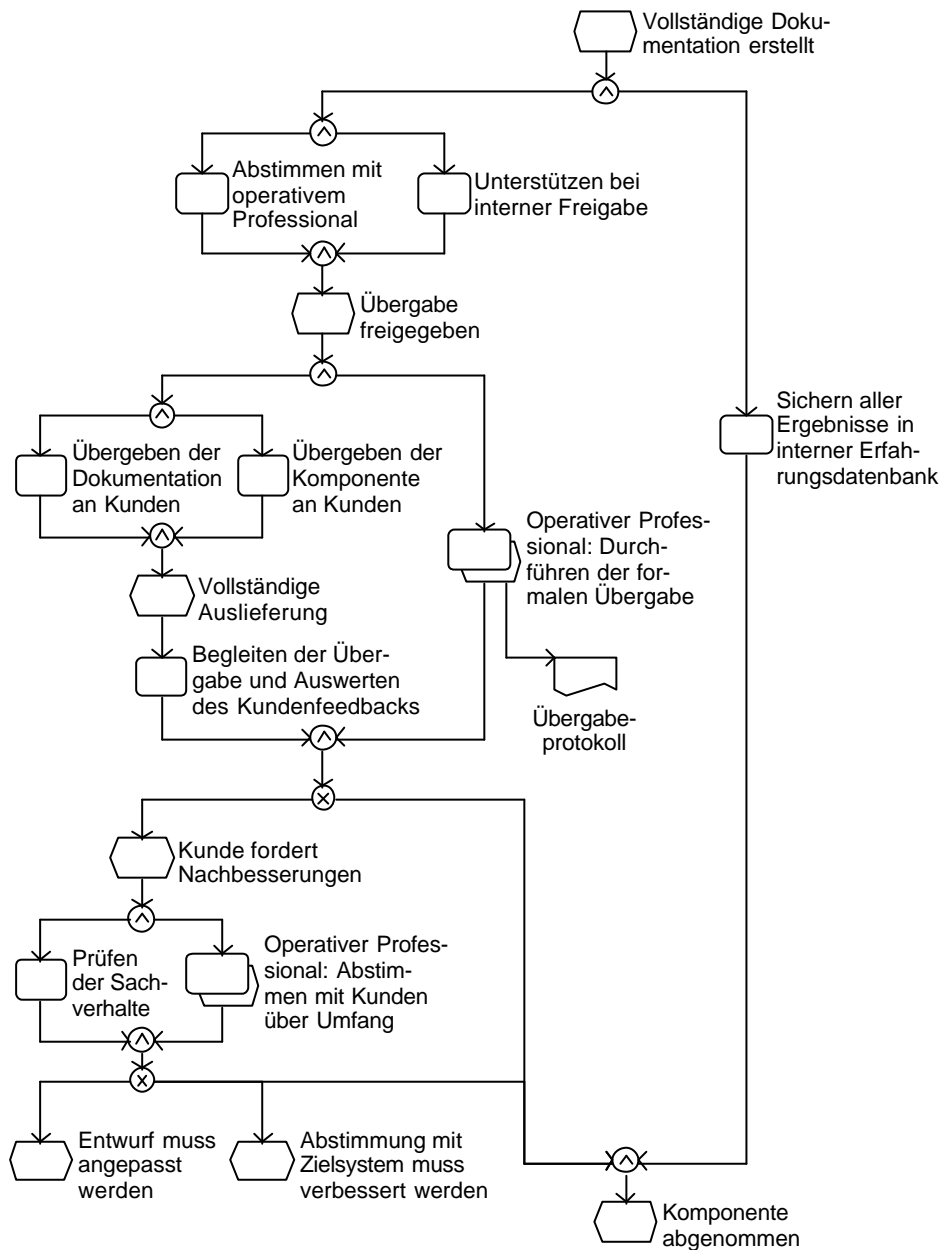


Abbildung 23: Übergeben der Komponente.

3.1.3.15.1 Tätigkeiten: Übergeben der Komponente

- Abstimmen mit operativem Professional über das weitere Vorgehen (beinhaltet das Bereitstellen aller benötigten Informationen)
- Unterstützen der internen Freigabeprozesse
- Übergeben aller Deliverables an den Kunden (Zusammenstellung, Versand etc.) – eine Vorauslieferung ist evtl. bereits erfolgt im Zusammenhang mit der Integration des Prototypen im Zielsystem)
- Begleiten des operativen Professionals beim Akt der formalen Übergabe sowie Auswerten des Feedbacks des Kunden aus dessen Prüfungen
- Prüfen von Beanstandungen bzw. Nachbesserungsforderungen in Bezug auf Anspruchsberechtigung, Tiefe und Umfang der Konsequenzen; Informieren des operativen Professionals für dessen Abstimmung/Verhandlung mit dem Kunden – eine Entwurfsan-

passung zu diesem Zeitpunkt des Prozesses (Ereignis „Entwurf muss angepasst werden“) ist sehr unwahrscheinlich und stellt gewissermaßen einen „Entwicklungs-GAU“ dar, da alle Schritte des Entwicklungsprozesses neu durchlaufen werden müssen

- Sichern aller im Projekt gemachten Erfahrungen und Ergebnisse für die interne Weiterverwendung und Qualitätssicherung

3.1.3.15.2 Kompetenzfelder: Übergeben der Komponente

Fähigkeiten/Fertigkeiten

- Vorgaben aus der internen Qualitätssicherung erfassen und umsetzen können
- interne Freigabe organisieren und durchführen können
- Informationsbedarfe von Entscheidern erkennen, einschätzen und Inhalte kontextgemäß und dem Kenntnisstand der Beteiligten angemessen vermitteln können
- Wesentliches klar und präzise erfassen und darstellen können
- Auslieferungsprozesse organisieren und abwickeln können
- präsentieren können
- Kundenfeedback auswerten und prüfen können, inwieweit dieser gerechtfertigt ist bzw. welche (technischen und auch wirtschaftlichen) Konsequenzen sich ergeben
- operativen Professional beim Verhandeln von Nachforderungen mit Informationen zum fachlichen und historischen Kontext unterstützen können
- Erfahrungen und Arbeitsergebnisse aus der Projektabwicklung zusammenfassen und für interne Wiederverwendung sowie zur Qualitätssicherung verfügbar machen

Wissen

- Qualitätssicherung, insbesondere Ablauf und Dokumentationspflichten bei der Übergabe
- Umgang mit Lasten- und Pflichtenheften
- Mängelansprüche und Fristen, Produkthaftung, Gewährleistung
- Präsentationsmethoden

Werkzeuge/Methoden

- Datenbanken zur Erfahrungssicherung (dies kann auch ein entsprechend strukturiertes Intranet sein)
- Tools zum Dokumentenmanagement

3.1.3.15.3 Beispiel: Übergeben der Komponente

Da es sich hier um ein internes Projekt handelt, fand die Übergabe an den Projektleiter und die anderen Mitarbeiter im Projekt statt. Aus Zeitgründen wurde die Komponente praktisch sofort nach erfolgreichem Funktionstest übergeben, so dass die sie im Rahmen der Emulation sofort eingesetzt werden konnte. Erst danach wurde die Dokumentation fertig gestellt und an einer für die Projektmitarbeiter zugänglichen Stelle gespeichert.