Visualisierung von Parametern komplexer Schnittstellen für eingebettete Systeme auf Basis von XML

Studienarbeit

Studiengang Informatik
Vertiefungsgebiet Softwaretechnik
Nebenfach Mathematik

von

Oliver Fick
An den Steinkisten 26
33178 Borchen

vorgelegt bei

Prof. Dr. rer. nat. Franz Josef Rammig

Dank und Erklärung

Dieses Dokument entstand im Rahmen einer Studienarbeit in Kooperation des IPL (Informatik- und Prozesslabor) und der Arbeitsgruppe Rammig (HNI) der Universität Paderborn. Während dem Anfertigen der Studienarbeit lernte ich Problemstellungen der Informatik zielstrebig zu studieren.

Für das interessante Studienarbeitsthema möchte ich mich im Besonderen bei Wolfram Hardt und Franz J. Rammig bedanken. Besonderer Dank gilt Stefan Ihmor, der mich über den Zeitraum der Erstellung der Studienarbeit fachlich engagiert betreut hat. Auch Markus Visarius, Michel Camel Kouamo Sime, Gilles Betrand Gnokan Defo, Andreas Scholand und Johannes Lessmann bin ich für ihre fachliche Unterstützung zu dank verpflichtet. Nicht zuletzt möchte ich Kathrin Tofall und Olaf Müller für das intensive Korrekturlesen dieser Arbeit danken.

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Paderborn, Juni 2003

1. Verzeichnisse

1.1. Inhaltsverzeichnis

1.2. Abbildungsverzeichnis

2. Einleitung

Diese Arbeit betrachtet Aspekte der Kommunikation, die beim Entwurf eingebetteter Systeme eine entscheidende Rolle spielen. Zentraler Blickpunkt dieser Studienarbeit ist die Definition und Darstellung von Parametern, die zur Beschreibung komplexer Kommunikations-Schnittstellen innerhalb von eingebetteten Systemen wichtig sind. Diese Parameter wurden festgelegt und mithilfe eines Editors visuell dargestellt.

2.1. Motivation

Der Entwurf eingebetteter Systeme hat sich in den vergangenen Jahren stark verändert. Die Systeme wurden immer komplexer und die dadurch erhöhten Entwicklungszeiten ließen die Kosten für die Entwickler in die Höhe schnellen. Um dem entgegen zu wirken, kam bald der Begriff des „Intellectual Property“ (IP) auf, der geistiges Eigentum betrifft, welches von einzelnen Entwicklern oder Firmen geschaffen und vermarktet wird. Hierbei handelt es sich hauptsächlich nicht mehr um Produkte, die Komplettlösungen für gestellte Aufgaben darstellen, sondern um Teilprodukte. Diese werden einzeln entworfen und können in Kombination mit anderen IPs zu komplexen Systemen, wie z.B. einem Handy oder einem PDA zusammengebaut werden. Der zentrale Aspekt hierbei ist, dass einmal entworfene IPs wieder verwendet werden können. Das spart Zeit bei der Entwicklung und dadurch auch Geld in den Firmen.
Das an dieser Stelle auftretende Problem ist, dass die gekauften Teilprodukte, wie z.B. ein Baustein für die Infrarot-Kommunikation, wie er in aktuellen Handys oft existiert, in das eigene Produkt integriert werden müssen, ohne die genau Funktionsweise zu kennen. Zur Anbindung müssen die Schnittstellen, die ein logischer Baustein besitzt, genau beschrieben werden. Die Anzahl der vorhandenen Anschlüsse, die genaue Beschreibung der eingesetzten Protokolle, Informationen über Stromverbrauch und weitere Aspekte müssen vom Entwickler vorher genau studiert werden, bevor er sich aus dem breiten Angebot von IPs diejenige aussucht, die am besten zu dem Rest seines Produktes passt. Im Fall eines Handys, beispielsweise, kann der Entwickler unter verschiedenen Infrarot-Schnittstellen unterschiedlicher Anbieter wählen.
Eine IP-Beschreibung kann dabei mehr oder weniger umfangreich sein, was wiederum Einfluss auf die Entwicklungszeit und die Kosten des Endproduktes hat. Es sollte aus ihr schnell herauszulesen sein, ob die Einbettung der gekauften Elemente in das eigene System überhaupt möglich ist.
Nachdem fremde IPs eingekauft wurden, muss der Entwickler diese noch in sein Produkt integrieren. Die Kommunikation zu anderen Bausteinen innerhalb des eingebetteten Systems muss aufgebaut werden. In einigen Fällen kann dies eine einfache Verdrahtung der Schnittstellen zwischen gekaufter IP und dem eigenen System sein. Meist wird dafür aber ein zusätzlicher logischer Baustein benötigt, der die Kommunikation zwischen den beiden Blöcken übernehmen wird, ein Schnittstellenmodul. Dieses wird Interface Block (*IFB*) genannt. Je nach Komplexität der Schnittstellen bedeutet dies wiederum großen Aufwand in der Entwicklungsphase.

Die Doktorarbeit von Stefan Ihmor, Diplom-Informatiker an der Universität Paderborn, beschäftigt sich unter anderem mit dem Entwurf der Kommunikation zwischen IPs. Ziel der Forschung in diesem Bereich ist es, die Interface Blöcke, die die Kommunikation zwischen einzelnen Blöcken des Systems übernehmen, möglichst automatisiert zu generieren [8, 9, 10, 11]. So wird schon in frühen Entwicklungsphasen der Datenaustausch zwischen gekauften und selbst entworfenen Bausteinen ermöglicht, ohne viel Arbeit von Seiten der Entwickler zu

fordern. Diese Studienarbeit unterstützt diese Forschung und zeigt die Ergebnisse auf, die eine Vorarbeit auf dem Weg zur Generierung der *IFBs* darstellen.

2.2. Problemstellung

Die Vorarbeit auf dem Weg zur automatischen Erzeugung eines Interface Blocks teilt sich in folgende Teilprobleme auf:

1. Die Definition der Parameter, mit deren Hilfe alle benötigten Daten zur Generierung der Kommunikation zwischen den Schnittstellen zweier IPs bereitgestellt werden. Hierbei werden die folgenden Aspekte differenziert betrachtet:
    - der physikalische Aufbau – die Struktur – des zu entwickelnden Systems;
    - die Eigenschaften der hier auftretenden Komponenten;
    - die verwendeten Protokolle der einzelnen IPs.
2. Die Entwicklung eines Editors, der die definierten Parameter
    - schnell und übersichtlich visualisiert,
    - auf Einhaltung der Datentypen prüft,
    - die Konsistenz zwischen den Schnittstellenbeschreibungen wahrt.
3. Um die Anzeige und Editierbarkeit der Parameter gewährleisten zu könne, muss zuvor ein Datenformat gewählt werden, in dem die Parameter beschrieben werden.

2.3. Aufgabenstellung

Die Aufgabe dieser Studienarbeit ist es, die für die Generierung benötigten Parameter zu definieren. Hierbei stellte sich die Strukturierung in folgende Teilbeschreibungen als sinnvoll heraus:

- die Architekturbeschreibung des Gesamtsystems,
- die Beschreibung der Schnittstellen (Interface Description, IFD) mitsamt eingesetzter Protokolle,
- die Beschreibung der Zielplattform (Target Platform Description, TPD), auf der ein generiertes Schnittstellenmodul (Interface Block, IFB) in der Entwicklungsphase realisiert werden soll.

Der entwickelte Editor erfüllt die Aufgabe, die Parameter in der Struktur des XML Schemas zu visualisieren und bei der Einhaltung von Einschränkungen, wie z.B. Datentypen der einzelnen Parameter, den Entwickler zu unterstützen.
Das Datenformat soll in Form einer auf XML basierten Gesamtbeschreibung festgelegt und mit Kommentaren versehen werden.

2.4. Gliederung der Arbeit

Zu Anfang wurde bereits eine kleine Einleitung darüber gegeben, welche Intention hinter dieser Arbeit stand. Der Hintergrund der Aufgabe und die Aufgabenstellung selbst sollten dem Leser damit vermittelt werden. Im folgenden Kapitel werden die Grundlagen auf denen die Arbeit aufbaut und die benutzten Hilfsmittel näher erläutert. Kapitel 4 beinhaltet eine detaillierte Beschreibung des entwickelten Interface Synthesis Formats (IFS Format). Dabei wird auf die Struktur und den Inhalt des zugehörigen XML Schemas eingegangen. Kapitel 5 zeigt den parallel entwickelten Editor, der auf Basis des IFS Formates die Eingabe und Anzeige der Parameter ermöglicht, die für die Interface Synthese relevant sind. In Kapitel 6 wird eine abschließende Zusammenfassung über die geleistete Arbeit und die noch offen stehenden Aufgaben gegeben. Im Anhang befinden sich zusätzlich Ausschnitte des IFS Formates und Editors in XML und Java.

3. State of the Art / Stand der Dinge

3.1. Anknüpfung an das IPQ Transfer Format

Bei der Identifizierung der benötigten Parameter war ein anderes Projekt innerhalb des Informatik- und Prozesslabors der Universität Paderborn sehr hilfreich. Im Rahmen dieses noch laufenden Projektes wurde eine formale Beschreibung komplexer Komponenten (Intellectual Property) entworfen, das IPQ Transfer Format.
Eine IP Beschreibung mithilfe des IPQ Transfer Formates hat den Zweck, anderen Designern die Funktionalität einer entwickelten IP zu vermitteln und somit eine Basis zu schaffen, an Hand derer ein Entwickler Entscheidungen über die Verwendung der angebotenen IP innerhalb des eigenen Projektes treffen kann. Bei der Entwicklung des IPQ Transfer Formates standen die Vereinheitlichung und Standardisierung von IP-Beschreibungen und die Übertragung dieser, beispielsweise über das Internet, im Vordergrund.

Das IPQ Transfer Format ist unterteilt in folgende Blöcke:

- *IP Charakterization*: Die Charakterisierung ist die Sammlung von Eigenschaften einer IP. Dabei handelt es sich um unterschiedlichste Parameter, wie z.B. Energieverbrauch, Datendurchsatz, Zugehörigkeit zu einem bestimmten Marktsegment oder angewandte Testmethoden. Während der Suche nach passenden IPs, sollen Designer anhand der Charakterisierung über eine Suchmaschine die für ihre Zwecke am meisten geeignete aussuchen.
- *IP Content*: Der Content (engl. für: Inhalt) stellt die Informationen über den Aufbau und die Funktionsweise einer gekauften IP dar. Er beinhaltet die exakte Hardwarebeschreibung, so dass der Entwickler die fremde IP aus diesen Informationen auf seinem eigenen Mikrochip realisieren kann.

Hinzu kommt der folgende Block, der nicht direkt Bestandteil des IPQ Transfer Formats ist. Er ist als Zusatz gedacht.

- *IP Taxonomies*: Die Taxonomies beinhalten zusätzliches Wissen über die IP. Dieses Wissen kann in unterschiedlichster Form dem Designer vermittelt werden. Hierbei wird die IP nicht einzeln betrachtet, sondern meist in einem Umfeld und größeren Rahmen. Zusätzlich wird Wissen über bereits angewandte Integrationen in komplexe Systeme übermittelt.

Das IPQ Transfer Format wurde mithilfe eines XML Schemas definiert. Damit wurde eine Baumstruktur geschaffen, die eine strukturierte Beschreibung der wichtigen Parameter ermöglichte. Die Informationen sind in den Blättern des Baumes, während die Knoten von der Wurzel bis zu den Blättern nur für die logische Einteilung wichtig sind. Beispielsweise hat der Knoten Frequency (engl. für Frequenz) die Kinder (Unterknoten) Minimum, Typical, Maximum, wodurch die Frequenz, mit der ein Chip Daten sendet oder empfängt, in einem Intervall zwischen Minimum und Maximum und einem typischen Wert angegeben werden kann.
Das IPQ Transfer Format spielt im Rahmen dieser Studienarbeit eine zentrale Rolle. Es diente einerseits als Hilfe für die grundlegenden Gedanken, andererseits war schon früh die enge Verknüpfung zwischen IPQ und IFS vorgesehen.
Der strukturelle Aufbau, also die Baumstruktur, und die damit verbundene auf XML basierende Beschreibung waren der Auslöser, dieses Konzept auch bei der Entwicklung des IFS Formats anzuwenden.
Wichtiger an dieser Stelle ist, dass im IPQ Transfer Format bereits viele Informationen über Schnittstellen integriert sind. Diese sind allerdings verstreut und damit nicht hierarchisch geordnet, somit für die Schnittstellensynthese ungünstig angeordnet. So sind beispielsweise

Informationen über die Anzahl von Pins nicht mit Informationen über Spannung, Frequenz und Latenz in Relation gebracht. Dieser Zusammenhalt zwischen allen wichtigen Parametern ist der zentrale Punkt dieser Arbeit. Das IPQ Transfer Format bot dabei einen guten Einstieg, da im Rahmen des IPQ Projektes schon viel Wissen über Beschreibungen von IPs und den dabei auftretenden Schnittstellen gesammelt wurde.
Die bereits im IPQ Transfer Format beinhalteten und für die Interface Synthese relevanten Parameter wurden neu strukturiert und um weitere ergänzt. Dadurch wurden das IPQ Projekt und das IFS Projekt, dem diese Studienarbeit angehört, eng miteinander verknüpft. Ein wichtiger Bestandteil des IFS Formates – die zentrale Schnittstellenbeschreibung – wurde komplett in das IPQ Transfer Format integriert. Sie wurde – wie auch schon das IPQ Transfer Format selbst – VSIA konform gehalten. VSIA ist die Virtual Socket Interface Alliance, eine Organisation, bestehend aus Einzelpersonen, Gesellschaften und anderen Organisationen, die Standards im Bereich der System-on-Chip Entwicklung festlegt. Der in diesem Zusammenhang wichtige Standard ist der „Virtual Component Attributes With Formats for Profiling, Selection and Transfer Standard Version 2 (VCT 2 2.x)“. Er definiert Parameter zur Beschreibung von IPs, Datentypen dieser Parameter und ihre Struktur.

3.2. Andere Schnittstellenbeschreibungen

Neben dem hier entwickelten IFS-Format existieren weitere Formate zur Beschreibungen von Schnittstellen. Einige davon sind standardisiert andere proprietär, manche formal und eignen sich zur automatisierten Synthese andere erfüllen diesen Anspruch nicht.
Eine Schnittstellenbeschreibung ist an dieser Stelle erwähnenswert, da sie ebenfalls eine Beschreibung in XML-Codierung nutzt. Diese heisst ComiX [12] und ist am OFFIS in Oldenburg entwickelt worden. Die Beschreibung unterscheidet sich im Vergleich zum hier vorgestellten IFS Format darin, dass nur Schnittstellen in Registerform beschrieben werden können. Die für den Zugriff dieser Register-Schnittstellen notwendigen Protokolle weichen in ihrer Ausdrucksmächtigkeit stark von dem FSM-basierten Ansatz dieser Arbeit ab, da eine Beschreibung allgemeiner Protokolle bei ComiX nicht Ziel war.

3.3. XML als Werkzeug

3.3.1. Einleitung

XML ist eine Metasprache, die entwickelt wurde, um Daten- und Dokumenttypen zu definieren. Sie ermöglicht die Formatierung von Daten auf bestimmte Weise. Im Vordergrund stand bei der Entwicklung von XML der Austausch von Daten über das Internet. In diesem Zusammenhang ist es stets notwendig, dass sich Entwickler von Programmen, die Daten austauschen sollen, über ein Format einig werden, mithilfe dessen die Daten übermittelt werden sollen.
Herkömmliche Verfahren waren dabei sehr einseitig und nicht immer erweiterbar. Das Hauptproblem war, dass in den übermittelten Dateien die Zugehörigkeit von einem Datum zu einem Verwendungszweck nicht sichtbar war. Viele Datenformate beruhten darauf, dass beispielsweise nur die Reihenfolge der Daten eine Zuordnung eines Datums in einer Datei zu seiner semantischen Bedeutung zuließ. Der Gebrauch von XML als Beschreibungssprache von Daten liefert diesen Zusammenhang.
An dieser Stelle ist eine Veranschaulichung anhand eines Beispiels sehr hilfreich. Die Daten der Studenten, die an einem Seminar teilnehmen, sollen über ein Webformular von den Studenten eingegeben und an einen Datenbankserver übermittelt werden. Das Format der übertragenen Daten könnte wie folgt aussehen:

```
Meier, Hans; hans@upb.de; Informatik; 1234567; Bauer, Ralf;
ralf@upb.de; Informatik; 2345678;
```

Im gewählten Format wurde das Semikolon (;) zum Trennen der einzelnen Daten voneinander benutzt. Dies ist kein sinnvolles Format, um die Daten zu übertragen. Es geht aus den Daten nicht hervor, was eine Name, eine Emailadresse oder ein Studiengang ist.

3.3.2. Document Type Definitions

Das folgende Format beinhaltet alle Informationen über die Zusammensetzung der Daten.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Student SYSTEM "studenten.dtd">
<Student matrikelnummer="1234567">
  <name>Meier, Hans</name>
  <mailadresse>hansi@upb.de</mailadresse>
  <studienfach>Informatik</studienfach>
</Student>
<Student matrikelnummer="2345678">
  <name>Bauer, Ralf</name>
  <mailadresse>ralf@upb.de</mailadresse>
  <studienfach>Informatik</studienfach>
</Student>
```

Die eigentlichen Daten sind hier durch *Tags* (so werden die Schlüsselworte zwischen „<“ und „>“ genannt) umschlossen. Es gibt – bis auf spezielle Ausnahmen – immer ein startendes und ein abschließendes Tag. Dabei sind startendes und abschließendes identisch bis auf den Schrägstrich „/“. Anhand dieser Tags wird den Daten eine Bedeutung gegeben. „Bauer, Ralf“ ist somit in einem Datensatz eindeutig als Name zu erkennen und durch ein verarbeitendes Programm von dem Studienfach unterscheidbar.

Ein Programm, welches die Daten verarbeitet, kann aufgrund dieser Technik relevante Daten erkennen und eindeutig identifizieren. Wenn z.B. ein Teil der Daten fehlte, könnte das Programm den aktuellen Datensatz verwerfen, aber alle anderen Datensätze der Datei trotzdem verarbeiten. Diese Möglichkeit ist bei anderen Formaten nicht gegeben oder nur mit relativ hohem Aufwand zu realisieren. Um in Programmen ein solches Datenformat verwenden zu können, werden mithilfe von XML Document Type Definitions (DTD) Übertragungsformate definiert. Eine passende DTD zu den obigen Datensätzen sieht wie folgt aus:

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT Student (name, mailadresse, studienfach)>
<!ATTLIST Student matrikelnummer CDATA #REQUIRED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT mailadresse (#PCDATA)>
<!ELEMENT studienfach (#PCDATA)>
```

Auf die genaue Syntax soll an dieser Stelle nicht eingegangen werden. Der Gebrauch von DTDs stellte sich zu Anfang der Studienarbeit als nicht sinnvoll heraus. Besser als DTDs sind XML Schemata.

3.3.3. XML Schemata

XML Schemata sind flexibler und erlauben wesentlich umfangreichere Definitionen von Dokumenttypen. Sie haben gegenüber Document Type Definitions einige Vorteile:

- DTDs sind eine eigene Sprache, XML Schemata sind selbst XML Dokumente. Dadurch sind Lernaufwand und Schulungskosten bei weitem nicht so hoch.
- Eine Modularisierung ist bei DTDs nur schwierig möglich, bei XML Schemata, einfach zu realisieren.
- Es gibt nur wenige Datentypen in DTDs.
- Das Definieren von eigenen Datentypen in DTDs ist gänzlich unmöglich.

Da, die Definition des IFS Formates mithilfe von XML Schemata realisiert wurde, wird zum besseren Verständnis an dieser Stelle dem Leser eine kleine Einführung in die Syntax von XML Schemata gegeben. Folgendes Dokument ist solch ein Schema.

```
<xsd:schema targetNamespace="http://www.upb.de/studienarbeit">
     <xsd:complexType name=“Student“>
      <xsd:all>
         <xsd:element name=“Name“ type=“xsd:string“/>
         <xsd:element name=“Mailadresse“ type=“xsd:string“/>
         <xsd:element name=“Studienfach“ type=“xsd:string“/>
      </xsd:all>
     </xsd:complexType>
</xsd:schema>
```

Ein XML Schema ist ein XML Dokument, welches durch das Schlüsselelement <xsd:schema> (das Präfix „xsd:“ wird nur per Konvention benutzt - es könnte auch jedes andere sein und sogar, bei entsprechend gewähltem Namensraum, entfallen) eingeleitet und abgeschlossen wird. Darauf folgen ein oder mehrere <xsd:element>-Elemente und gegebenenfalls Typdefinitionen (`<xsd:complexType>`). Eine korrekte Instanz des obigen Schemas wäre:

```
<Student>
  <Name>Müller, Hildegard</Name>
  <Mailadresse>hilde@upb.de</Mailadresse>
  <Studienfach>Wirschaftsinformatik</Studienfach>
</Student>
```

An dieser Stelle sei angemerkt, dass Groß- und Kleinschreibung beachtet werden müssen. In dem Schema wurden die Tags mit großem Anfangsbuchstaben definiert. Folgende Instanz wäre somit nicht korrekt:

```
<student>
  <name>Müller, Hildegard</name>
  <mailadresse>hilde@upb.de</mailadresse>
  <studienfach>Wirschaftsinformatik</studienfach>
</student>
```

Datentypen:

Der Standarddatentyp in XML Schemata ist *anyType*. Von diesem sind alle anderen Datentypen abgeleitet. Dies ermöglicht den Aufbau einer Datenstruktur, in der an manchen Stellen beliebige Inhalte vorkommen dürfen. Der Default-Typ *anyType* muss nicht explizit

angegeben werden. Bei fehlender Typangabe wird der Datentyp implizit auf *anyType* gesetzt. Es wird zwischen einfachen und komplexen Datentypen unterschieden.

Zu den einfachen Datentypen zählen unter anderem die ‚atomaren Typen'. Dies sind die aus anderen Programmiersprachen weit verbreiteten Typen wie z.B. *string*, *float*, *decimal*, *boolean*, *date*, *time.*
Besondere Beachtung muss an dieser Stelle der Ableitung von Datentypen aus bereits definierten gegeben werden. So ist *integer* z.B. kein atomarer Datentyp. *Integer* ist durch Ableitung von *decimal* durch Einschränkung auf null Nachkommastellen entstanden.

Komplexe Datentypen werden durch das <complexType>-Element definiert. Sie dürfen weitere Elemente enthalten sowie Attribute tragen. Sowohl Elemente als auch Attribute können dabei Defaultwerte erhalten.

Im Beispiel wurde der komplexe Datentyp *Student* definiert. Er besteht aus den drei aufeinander folgenden Elemenen *Name, Mailadresse, Studienfach,* die alle vom Typ s*tring* sind.

Das Tag `<xsd:all>` gibt das Inhaltsmodell an. Bei komplexen Datentypen können verschiedene Inhaltsmodelle verwendet werden:

- `<all>` wird benutzt, falls alle Kindelemente genau ein Mal oder gar nicht auftreten sollen und die Reihenfolge dieser beliebig ist
- `<sequence>` findet Anwendung, falls sowohl die Anzahl der Kindelemente als auch die Reihenfolge genau festgelegt ist
- `<choice>` wird benutzt, falls genau eines der Kindelemente in der Instanz angegeben werden soll
- `<group>` bietet die Möglichkeit, Elemente zu Gruppen zusammenzufassen und ist somit für die Übersichtlichkeit und Pflege eines Schemas nützlich. Außerdem kann eine mit `<group>` definierte Gruppe an verschiedenen Stellen im Schema referenziert werden.

Modularisierung:

Wie bereits erwähnt, bieten XML Schemata eine einfache Möglichkeit, größere Schemata zu modularisieren. Dies kann sehr einfach geschehen. Der oben bereits definierte Typ *Student* kann einfach innerhalb anderer Schemata eingebunden werden. Nehmen wir an, dass die im Beispiel verwendete Definition von *Student* in der Datei *Student.xsd* erfolgt ist. Sie wird nun in einem neuen Schema, z.B. *Fachschaftsrat.xsd* wie folgt eingebunden:

```
<xsd:schema targetNamespace="http://www.upb.de/studienarbeit">
  <xsd:import namespace="http://www.upb.de/studienarbeit"
                              schemaLocation="./Student.xsd"/>
  <xsd:complexType name="Fachschaftsrat">
    <xsd:sequence>
      <xsd:element name=“Fachschaftsname“ default =“FB17“
                   type=“xsd:string“/>
      <xsd:element name="Vorsitzender" type="xsd:Student"/>
      <xsd:element name="ZweiterVorsitzender"
                   type="xsd:Student"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

Vorsitzender und *ZweiterVorsitzender* sind beide vom Typ *Student*. Eine gültige Instanz hierzu wäre folgende XML Datei:

```
<Fachschaftsrat>
  <Fachschaftsname></Fachschaftsname>
  <Vorsitzender>
    <name>Meier, Hans</name>
    <mailadresse>hansi@upb.de</mailadresse>
    <studienfach>Informatik</studienfach>
  </Vorsitzender>
  <ZweiterVorsitzender>
    <name>Bauer, Ralf</name>
    <mailadresse>ralf@upb.de</mailadresse>
    <studienfach>Informatik</studienfach>
  <ZweiterVorsitzender>
</Fachschaftsrat>
```

Die Technik der Modularisierung wurde bei der Entwicklung des IFS Formates verstärkt eingesetzt. Große XML Schemata lassen sich auf diese Weise in viele kleine aufteilen, was die Übersichtlichkeit fördert. Vorteil dieser Technik ist außerdem, dass – wie schon im Beispiel angewendet – bereits definierte Typen öfter wieder verwendet werden können.

Defaultwerte:

Innerhalb eines Elementes kann bereits im XML Schema ein Standardwert gesetzt werden. Dies ist hilfreich, wenn in einer Instanz bestimmte Werte nicht angegeben werden, aber später ein Wert benötigt wird. Wenn im Fall der obigen Instanz beispielsweise die Daten über einen Webserver in eine Datenbank eingefügt werden, würde ein Schema-Prozessor, der die Daten und das Schema vergleicht, den Standardwert von *Fachschaftsname* übernehmen. Auch diese Technik fand bei der Erstellung des IFS Schemas Verwendung. Sämtliche Elemente innerhalb des IFS Schemas haben Standardwerte, die im Schema verankert sind.

Enumerations:

Durch die Anwendung der Technik der „Enumeration“ kann in einem XML Schema eine Liste an Auswahlmöglichkeiten zu einem Element gespeichert werden.

```
<xsd:element name="Geschlecht">
   <xsd:simpleType>
      <xsd:restriction base="xsd:string">
         <xsd:enumeration value="männlich"/>
         <xsd:enumeration value="weiblich"/>
      </xsd:restriction>
   </xsd:simpleType>
</xsd:element>
```

Das im Beispiel verwendete Element „Geschlecht“ ist vom Datentyp *String*, dessen Wertebereich auf die in der *Enumeration* angegebenen Werte „männlich“ und „weiblich“ beschränkt wurde. Diese Technik wird im IFS Format an vielen Stellen benutzt, an denen eine Auswahl vorgegebener Werte sinnvoll ist.

4. Das IFS Format

Das IFS Format (Interface Synthesis Format) ist als XML Schema realisiert und in hierarchische Module aufgeteilt. Das IFS Format beinhaltet alle für die automatisierte Schnittstellensynthese notwendigen Informationen. Dazu gehören Architektur-, Schnittstellen- und Zielplattformbeschreibungen. Die Aufteilung der einzelnen Beschreibungen in verschiedene Sub-Schemata ermöglicht die Wiederverwendung an verschiedenen Stellen. Im Folgenden wird der strukturelle und inhaltliche Aufbau des IFS Gesamtschemas erläutert. Darauf folgt eine detaillierte Erklärung der einzelnen Schemata.

4.1. Grundgedanken

4.1.1. Aufbau des Schemas

Die Struktur des IFS Formates beruht auf der Architektur eines komplexen Kommunikationssystems. Neben der logischen Protokollsicht werden im IFS Format auch physikalische und elektrotechnische Eigenschaften berücksichtigt. Im IFS Format werden damit Aspekte der Strukturbeschreibung und Verhaltensbeschreibung vereint. Diese Informationen sind grundlegender Bestandteil für eine darauf aufbauende Synthese, die ausgehend vom IFS Format automatisiert Schnittstellenmodule (*IFB*, Interface Block) generiert.

Ein durch das IFS Format beschriebenes Gesamtsystem besteht aus verschiedenen Hauptplatinen (*Boards*), auf denen verschiedene Recheneinheiten, wie z.B. Prozessoren, FPGAs (Field programmable gate array) und ASICs (Application-specific integarted circuits), angebracht sind. Diese werden im Folgenden zusammenfassend mit dem Begriff *Chip* bezeichnet. Innerhalb dieser *Chips* sind die *Tasks* verwirklicht. Die Tasks sind die Funktionsblöcke, welche innerhalb des Gesamtsystems Algorithmen (Funktionalität) realisieren. Auf allen Ebenen der Systemarchitektur sind Medien, z.B. Busse, vorhanden. Alle Systemkomponenten (*Board*, *Chip*, *Task*, *Medium*) außer dem System selbst weisen Schnittstellen auf, die die Kommunikation innerhalb einer Hierarchieebene bzw. zur angrenzenden Ebene ermöglichen.

Folgende Abbildung zeigt eine beispielhafte Systemarchitektur, mit den einzelnen Systemkomponenten, Schnittstellen und Kommunikationswegen.

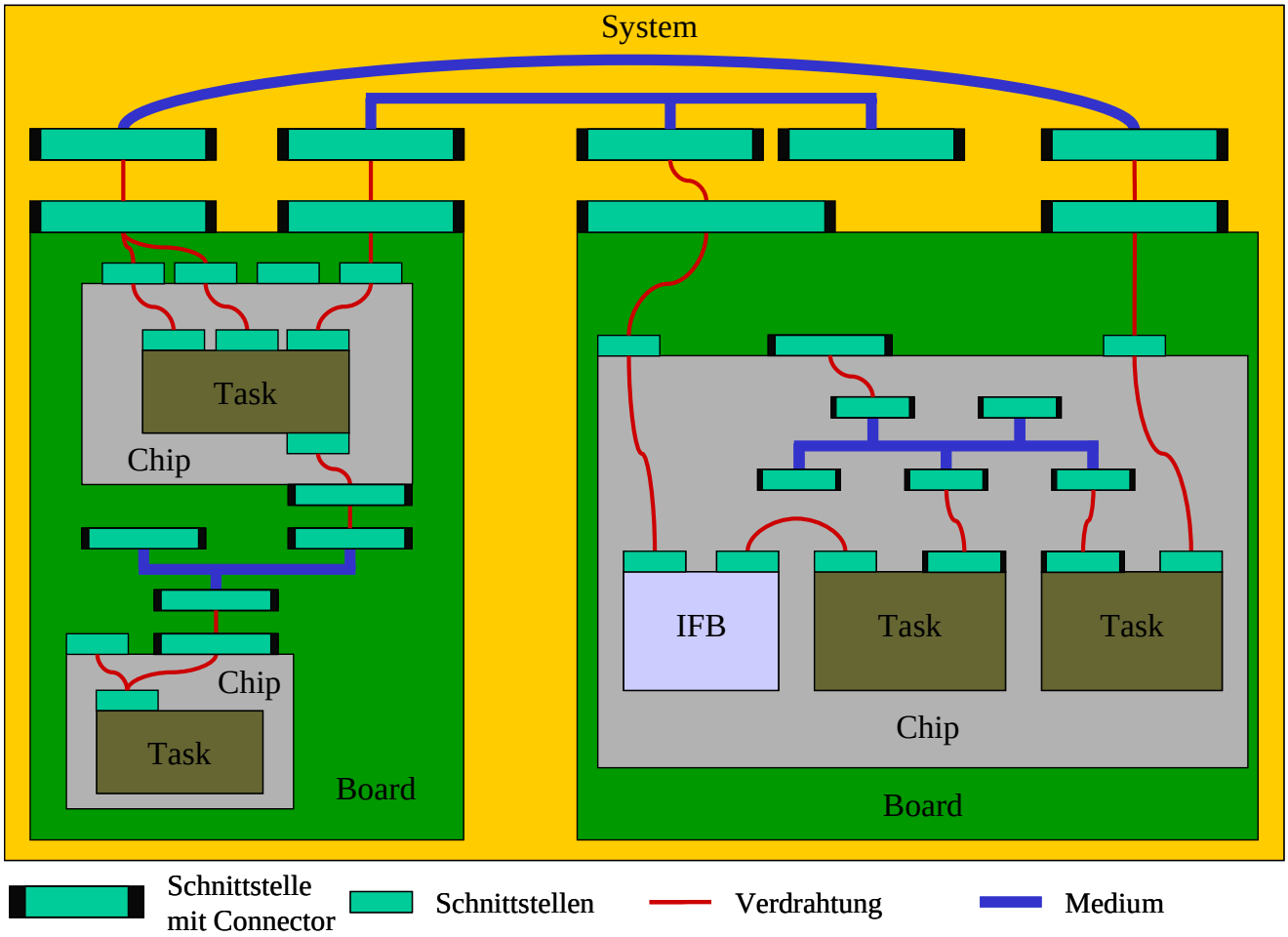

Abbildung 4-1: Komplexe Systemarchitektur

In Abbildung 4-1 sind unterschiedliche Arten von Kommunikation in einem Szenario dargestellt. Es wird ein *System* mit zwei *Boards* dargestellt. Im rechten *Board* ist die Kommunikation zwischen zwei *Tasks* innerhalb eines *Chips* über einen *Medium* (On-Chip-Bus) realisiert. Zusätzlich kommunizieren beide *Tasks* dieses *Chips* über *Chip*- und *Board*grenzen hinaus mit der oberen *Task* des linken *Boards*. Im linken *Board* ist zwischen dem oberen und unteren *Chip* ein On-Board-Bus dargestellt. Über dieses *Medium* kommunizieren die beiden *Chips* miteinander sowie mit anderen, hier nicht dargestellten Komponenten. Im oberen Bereich von Abbildung 4-1 sind zwei *Medien* dargestellt, über die die beiden *Boards* miteinander kommunizieren. Das obere *Medium* könnte dabei eine Kabelverbindung, das Untere ein Bussystem wiedergeben.
Wenn innerhalb dieser geschlossenen Systemarchitektur nun zwei aktive Systemkomponenten (*Task* oder *Medium*) miteinander kommunizieren wollen, diese aber inkompatibel zueinander sind, wird ein *IFB* (Interface Block) benötigt. Ein *IFB* kann nur innerhalb eines *Chips* implementiert werden und ist aufgrund der Zielplattformbeschreibung des *Chips* auch an diesen gebunden.
In diesem Beispiel benötigt die linke *Task* des rechten *Boards* ein solches Schnittstellenmodul (*IFB*), für die Verständigung mit der oberen *Task* des linken *Boards*. Der *IFB* ist hier notwendig, weil die Schnittstellen der beiden *Tasks* nicht kompatibel sind und eine direkte Verdrahtung damit unmöglich ist. Die automatische Erzeugung dieser Schnittstellenmodule ist eine weiterführende Arbeit im Umfeld der Schnittstellensynthese. Die Voraussetzungen für eine automatisierte Generierung soll mithilfe des IFS Formates geschaffen werden.

4.1.2. Aufteilung in Sub-Schemata

Die Fülle an Informationen, die zur Beschreibung von Kommunikationsschnittstellen in komplexen Systemen benötigt werden, machte es sinnvoll, das IFS Format in logisch getrennte Formate aufzuteilen. Eine solche Aufteilung ermöglicht separate Teilbeschreibungen einzelner Systemkomponenten. Beispielsweise kommt in vielen Systemen keine Aufteilung in mehrere *Boards* vor. Hierbei würde eine Modellierung ab *Board*ebene reichen. Die gewählte Aufteilung vereinfacht weiterhin die Wiederverwendung von vollständigen Systemkomponenten. Der später vorgestellte IFS Editor benutzt allerdings immer die Systemkomponente „System“ als Wurzelknoten der Systemarchitektur.

Das IFS Format teilt sich in folgende logische Formate (Schemata) auf:

- **Systemarchitektur**
  - Systemkomponenten: System, Board, Chip, Task, Medium und IFB
  - Verdrahtung
- **Schnittstellenbeschreibung** (**IFD**, Interface Description)
  - Physikalische Schnittstelle (Struktur, Geometrie)
  - Physikalische Eigenschaften
  - Protokolle
- **Zielplattformbeschreibung** (**TPD**, Target Platform Description)

Die Formate sind aus implementierungstechnischen Gründen wiederum in Einzelschemata aufgeteilt. Die Aufteilung in die soeben genannten logischen Formate spiegelt sich bei genauer Betrachtung auch im Aufbau des Schemas wider. Die Systemarchitekturbeschreibung liefert das Grundgerüst, in das die IFD und die TPD eingegliedert sind. Die Protokolle, die Teil der IFD sind und damit logisch zur Schnittstellenbeschreibung gehören, sind im XML Schema aus dieser herausgenommen und auf der Systemebene eingehängt worden. Der Grund hierfür lag darin, dass Protokolle in einem System mehrfach vorkommen können, und die Beschreibung so nur einmal erstellt werden muss. Die Verbindung zwischen IFD Schema und den Protokollen erfolgt daher über einen Referenzmechanismus.

4.1.3. Verwendete Datentypen

Bevor hier der gesamte Aufbau des IFS Formates detailliert erklärt wird, sollten noch ein paar Worte zu den verwendeten Datentypen gesagt werden. Die meisten dieser Datentypen wurden aus den bereits für das IPQ Transfer Format entworfenen Datentypen übernommen. Das Präfix „*ipq:*“ kennzeichnet diese. Sie sind alle aus den XML Grunddatentypen abgeleitet. Die vollständige Definition in XML Notation erfolgt im Anhang unter **8.1.1**. Folgende Datentypen treten im IFS Format auf:

ipq:M..16
- ist abgeleitet von `xs:string`,
- kann beliebige Zeichenketten aufnehmen,
- ist beschränkt auf eine Länge von 16 Zeichen.

ipq:M..32
- entspricht *ipq:M..16*, kann aber bis zu 32 Zeichen fassen.

ipq:M..256
- entspricht *ipq:M..16*, kann aber bis zu 256 Zeichen aufnehmen.

ipq:NR1..8
- ist abgeleitet von `xs:unsignedInt`,
- ist nur in der Lage positive ganze Zahlen oder 0 aufnehmen,
- ist eingegrenzt auf eine Länge 8 Stellen.

ipq:NR1..16
- ist gleichzusetzen mit *ipq:NR1..8*, kann aber bis zu 16 Stellen beinhalten.

ipq:NR2S..3.3
- ist eine Ableitung von `xs:decimal`,
- kann positive und negative Dezimalzahlen aufnehmen,
- ist eingegrenzt auf eine Länge von 6 Stellen, wobei maximal 3 vor und 3 nach dem Komma möglich sind.

ipq:MinTypMaxNR2S..3.3
- ist ein komplexer (zusammengesetzter) Datentyp,
- strukturiert ein Attribut in 3 Sub-Attribute vom Typ *ipq:NR2s..3.3*, wodurch die Angabe von minimalem, typischem und maximalem Wert möglich ist.

4.2. Systemarchitektur

Die Systemarchitektur ist ein Plattform-basiertes Modell, das alle Systemkomponenten, die zur Interface-Synthese benötigt werden, verwaltet. Da in der Systemarchitektur mehrere Systemkomponenten instanziiert werden können wird die Beschreibung komplexer Szenarien ermöglicht. Auf oberster Ebene der Systemarchitektur steht das *System* selbst. Als weitere Systemkomponenten folgen *Board*, *Chip*, *Task* und *Medium*. Hinzu kommen noch die Interface Blöcke (*IFB*), die das Produkt der Schnittstellensynthese sind.

4.2.1. Versionsangaben

Die Vereinheitlichung gleicher und ähnlicher Parameter stand bei der Entwicklung des Schemas im Vordergrund. Um dem Entwickler die Unterscheidung gleicher Komponenten innerhalb seines Projektes zu ermöglichen, wie zum Beispiel mehrerer *Chips* oder *Tasks*, wurde ein XML Schema erstellt, welches Versionsangaben und Informationen über den Entwickler enthält. Es handelt sich dabei um *Version.xsd*. Dieses Teil-Schema wird an verschiedenen Stellen innerhalb des Gesamtschemas eingebunden, wodurch unterschiedliche Komponenten – unter anderem das *System* selbst – identifiziert werden können. Es beinhaltet folgende Parameter mit den angegebenen Datentypen, die schon in **4.1.3** erläutert wurden:

Design (ipq:M..256):
Hiermit soll der Designer der zugehörigen Komponente einen aussagekräftigen Namen geben.

Description (ipq:M..256):
Dies ermöglicht dem Entwickler eine kurze Beschreibung der Funktionalität dieser Komponente.

Designer (ipq:M..256):
Jener Parameter identifiziert den zuständigen Entwickler, wie z.B. durch seinen Namen oder eine Abteilungsbezeichnung bei mehreren Entwicklern.

Company (ipq:M..256):
Dies ist die Angabe der Firma, die die betreffende Komponente entwickelt hat. Falls einzelne Komponenten nicht innerhalb derselben Firma erstellt wurden, sondern von anderen Firmen gekauft wurden, ist diese Angabe sinnvoll.

Release (ipq:NR1..8):
Dies ist die Nummer des Releases.

Version (ipq:NR1..8):
Das ist die Versionsnummer der Komponente.

Die exakte XML Notation des *Version*-Schemas befindet sich im Anhang unter **8.1.2.1**.

4.2.2. System

Die oberste Ebene des IFS Formates bildet die Systemschicht. Sie wird mithilfe des Moduls *System.xsd* beschrieben. Es bildet die Wurzel des IFS Formates und repräsentiert das Gesamtprojekt aus Sicht des Entwicklers. Eine Übersicht über den Aufbau dieses Teilschemas liefert folgende Abbildung.

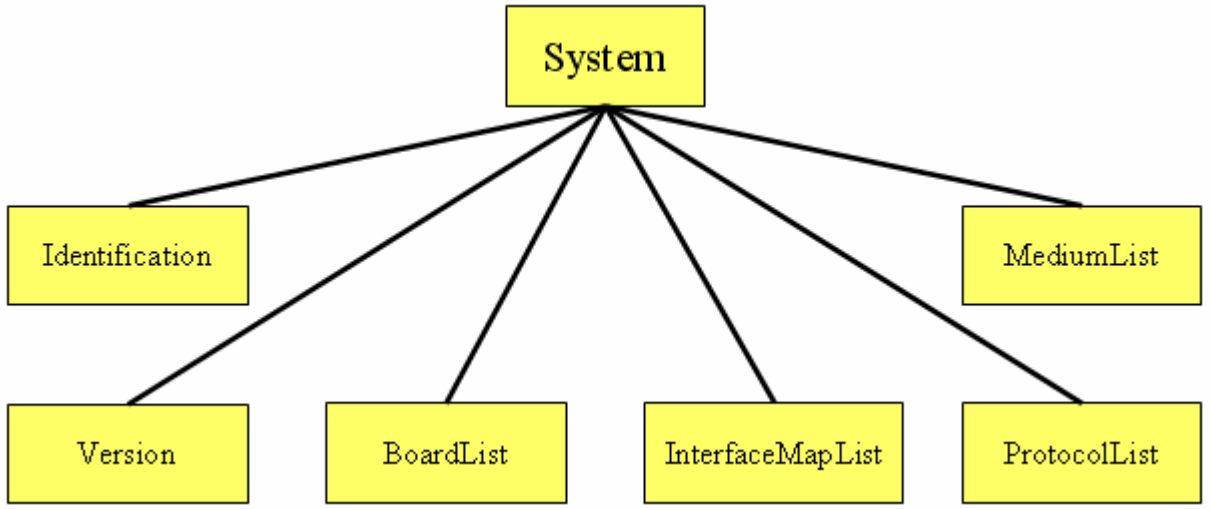

Abbildung 4-2: Aufbau des XML Schemas *System.xsd*

Wie alle Komponenten der Systemarchitektur hat das *System* Parameter, die ein eindeutiges Identifizieren und Referenzieren ermöglichen. Es handelt sich dabei um folgende Parameter, die in *Identification* zusammengefasst sind.

Name (ipq:M..256):
An dieser Stelle kann der Entwickler dem System bzw. dem Projekt einen Namen geben.

ID (ipq:NR1..16):
Zur internen Identifizierung und zum Bilden von Referenzen erhalten nahezu sämtliche Komponenten und Teilkomponenten, die innerhalb des IFS Schemas vorkommen, eine *ID*. Ihre Eindeutigkeit ist beschränkt auf die Ebene, in der sie verwendet wird. Hierbei handelt es sich um die *ID*, mit der das Gesamtprojekt identifiziert werden kann.

Description (ipq:M..256):
Mithilfe der *Description* kann der Designer Anmerkungen und Kommentare zum *System* in den Daten der Beschreibung verankern. Die *Description* kommt genauso wie die *ID* und der *Name* in verschiedenen Komponenten des Systemarchitektur und der Schnittstellen-beschreibung vor.

Des Weiteren beinhaltet das System-Schema das bereits erwähnte Schema zur Angabe von Versionsinformation. Dessen Aufbau wurde in **4.2.1** bereits detailliert dargelegt und findet sich in *Abbildung 4-2: Aufbau des XML Schemas System.xsd* in dem Knoten *Version* wieder.

Die Knoten *BoardList*, *MediumList*, *ProtocolList* und *InterfaceMapList* repräsentieren Listen weiterer Komponenten, die später in separaten Kapiteln behandelt werden. Die Liste der *Boards*, die in der Systemhierarchie auf das *System* folgen, wird in der *BoardList* verwaltet. Auf gleicher Stufe befindet sich die Liste der Medien (*MediumList*), die beispielsweise zum Verbinden mehrerer *Boards* verwendet werden. Wie bereits an anderer Stelle erwähnt, sind die im Gesamtsystem verwendeten Protokolle ebenfalls auf der Systemebene in der *ProtocolList* verankert. Die Verbindungen von Schnittstellen der Systemkomponenten (*Boards* und *Medien*) auf Systemebene werden im Einzelnen durch die „Interface Maps“ beschrieben und in der *InterfaceMapList* verwaltet.

Die exakte XML Notation des *System*-Schemas befindet sich im Anhang unter **8.1.2.2**.

4.2.3. Board

Einzelne Plattformen, z.B. Steckkarten wie PCI-Karten, oder auch separate Platinen wie FPGA-Boards, auf denen sich elektrotechnische Bausteine befinden, werden durch den Begriff *Board* vereinheitlicht. Aus ihnen wird das Gesamtsystem zusammengesetzt. Es ist allerdings auch möglich, dass nur ein einziges *Board* vorhanden ist. Folgende Abbildung liefert einen Überblick über die Struktur eines *Boards*.

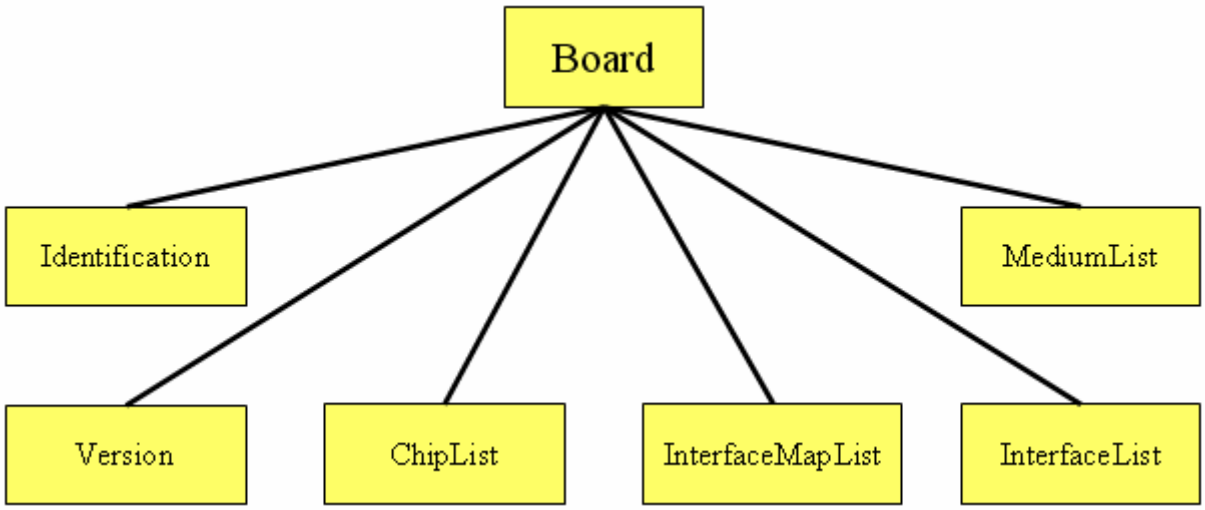

Abbildung 4-3: Aufbau des XML Schemas *Board.xsd*

Der Part der *Identification* weicht nur gering von dem entsprechenden Block des *Systems* ab. Hierbei sind die Parameter *Name (ipq:M..256)*, *ID (ipq:NR1..16)* und *Description (ipq:M..256)* in ihrer Funktion gleich denen in *System*. Ergänzend kommt noch ein Parameter, der eine Klassifizierung von *Boards* ermöglicht:

Category (ipq:M..256):
Die *Board*-Kategorie ordnet dem *Board* einen bestimmten Typ zu. Hierbei bietet das IFS Format mittels der Technik der *Enumerations* bis jetzt zwei Typen zur Auswahl an. Eine spätere Erweiterung des Schemas ist möglich. Die Beschränkung auf die Typen *Altera UP1* und *Spyder Virtex XCV300* begründet sich darin, dass bisher ausschließlich diese beiden FPGA-Board Typen verwendet wurden.

Die Versionsangaben aus *Version* sind selbstverständlich auch in *Board* vorhanden. Ebenso wie das *System* besitzt ein *Board* die Parametergruppen *InterfaceMapList* und *MediumList*. Die Funktion dieser Parameter wird später an passender Stelle näher erläutert. Die nachfolgende Hierarchieebene ist die Menge an *Chips*, die auf einem *Board* vorhanden sind und in der *ChipList* zusammengefasst werden. Das Board ist die erste Komponente, die Schnittstellen aufweist. Diese werden in der *InterfaceList* verwaltet. Eine genaue Beschreibung der Schnittstellen wird in **4.3** später erbracht.

Die exakte XML Notation des *Board*-Schemas befindet sich im Anhang unter **8.1.2.3**.

4.2.4. Chip

Der nächste Betrachtungspunkt betrifft einzelne *Chips*. Hierbei handelt es sich um eine Implementierungsplattform, wie z.B. FPGAs oder ASICs, die unterschiedliche Aufgaben innerhalb eines Systems übernehmen können In ihnen werden die eigentlich ausführenden Komponenten, die *Tasks*, sowie Medien (OnChipBus) und IFBs implementiert. Ein *Chip* wird wiederum durch eine Menge von Parametergruppen und Sub-Schemata zusammengesetzt:

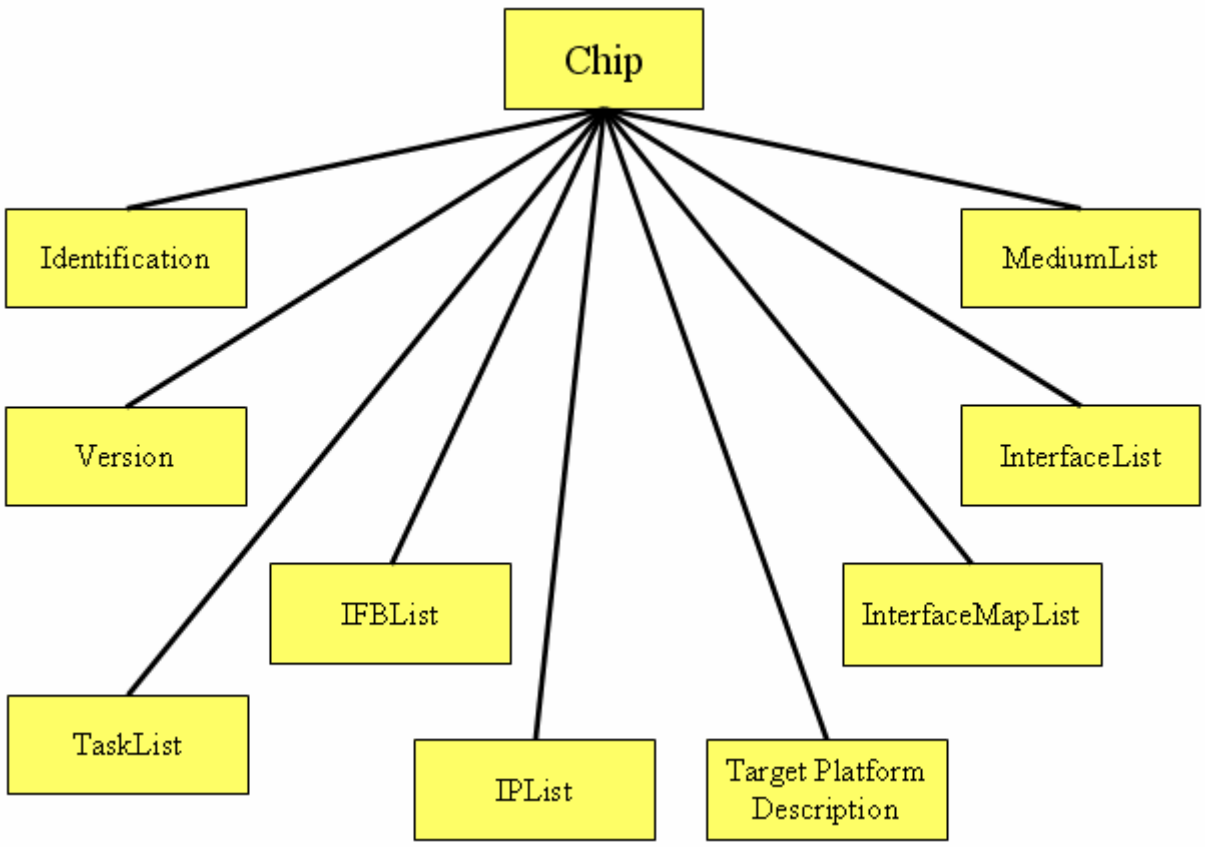

Abbildung 4-4: Aufbau des XML Schemas *Chip.xsd*

Die Module *Version* und *Identification* sind auf *Chip*ebene nicht anders als auf der *Board*ebene. Die *Category* ist wiederum als *Enumeration* realisiert, durch deren Hilfe ein Angebot an Auswahlmöglichkeiten im XML Schema bereit steht: *Xilinx Virtex XCV300BG432 -4*, *Xilinx Virtex XCV 300E6PQ240C*, *Altera Flex EPF10K20RC240-4* und *Altera Max EPM7128SLC84-7*.

Die *IFBList* ist dafür vorgesehen, später bereits generierte oder selbst erzeugte Schnittstellenmodule, Interface Blöcke, aufzunehmen. Mithilfe der IPList werden vollständige IPs in Form des IPQ Transfer Formates in das IFS Format eingebunden. Das IPQ Transfer Format ist allerdings nicht Bestandteil dieser Studienarbeit und wird nicht weiter ausgeführt.
Bereits auf *Board*ebene wurden *InterfaceList* und *InterfaceMapList* erwähnt. Ein *Chip* hat ebenso Schnittstellen und Verdrahtungen (*Interface Maps*). Hinzu kommen wiederum die *Medien*, welche auf *Chip*ebene genutzt werden können. Diese werden in der *MediumList* verwaltet.

Ein wichtiger Bestandteil der *Chip*-Beschreibung ist die *Target Platform Description*. Dabei handelt es sich um eine detaillierte Beschreibung der Ressourcen und Clock-Netzwerke des *Chips*. Diese Informationen der Zielplattform werden zur Synthese, d.h. bei der Generierung von *Interface Blöcken* (IFB), benötigt. Das bedeutet, dass ein generierter IFB an eine bestimmte *Chip*-Kategorie gebunden ist. Eine detaillierte Beschreibung der hierfür vorgesehenen Parameter erfolgt später in **4.3.3**.

Die nächste Hierarchiestufe der Systemarchitektur unterhalb der *Chips* ist die *Task*ebene, auf der auch Medien und IFBs vorkommen. Die auf einem *Chip* realisierten *Tasks* werden in der *TaskList* beschrieben, Medien und IFBs entsprechend in der *Medium*- bzw. *IFBList*.

Die vollständige XML Notation des *Chip*-Schemas ist der des *Board*-Schemas bis auf Namen und zusätzliche Parametergruppen sehr ähnlich und daher nicht explizit im Anhang vorhanden.

4.2.5. Task

Die letzte Hierarchiestufe der Systemarchitektur bezieht sich unter anderem auf die *Tasks*. Dabei handelt es sich um atomare Komponenten innerhalb der Systemarchitektur, die jeweils einzelne Funktionalität des Gesamtsystems implementieren.

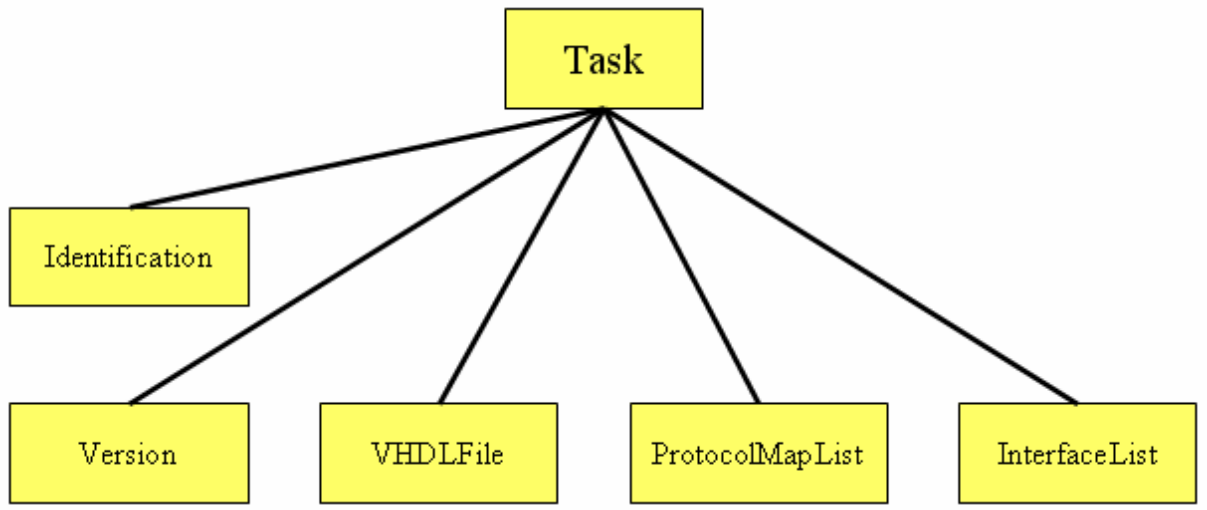

Abbildung 4-5: Aufbau des XML Schemas *Task.xsd*

Der Aufbau des zugehörigen XML Schemas wird in Abbildung 4-5 verdeutlicht. Die Angaben zur Identifikation und Versionskontrolle sind wiederum in *Identification* und *Version* verankert. Die Kategorie (*Category*) innerhalb der Identifikation ist der Parameter, der die Art der Task beschreibt. Mögliche Typen sind *IP*, *VHDL_Entitiy* und *Netlist*. Wenn zu einer *Task* ein VHDL File existiert, kann eine Referenz zu diesem in *VHFLFile* angegeben werden.

Die *InterfaceList* beinhaltet wiederum die Beschreibungen der Schnittstellen der *Task*. Neu ist die *ProtocolMapList*. Hierbei handelt es sich um die Abbildung (Mapping) der verwendeten Protokolle auf die Schnittstellen der *Task*. Dabei werden einzelnen *Pins* – dies sind die physikalischen Pins der Schnittstellen – den logischen Protokollpins innerhalb der Protokollbeschreibung zugeordnet. Einer Schnittstelle kann somit ein vollständiges Protokoll auf Pin-Ebene zugeordnet werden. Dies erledigt die *ProtocolMap*.

Die vollständige XML Notation des *Task*-Schemas ist ebenso wie bei *Chip* dem *Board*-Schema bis auf Namen und zusätzliche Parametergruppen sehr ähnlich und ist aus Platzgründen nicht im Anhang vorhanden.

4.2.6. Medium

Medien können prinzipiell auf allen Hierarchieebenen vorkommen. Dabei kann ein Medium in unterschiedlichen Formen vorliegen. Je nachdem auf welcher Hierarchieebene es im IFS Format vorkommt, wird etwas anderes darunter verstanden. Trotzdem teilen sich alle Medien Gemeinsamkeiten, so dass eine einzige Beschreibung für Medien ausreichend ist. Auf Chipebene, also innerhalb eines *Chips*, sind z.B. Busse (On-Chip-Bus) gemeint. Innerhalb eines *Boards* handelt es sich meistens ebenfalls um Busse, die z.B. verschiedene Chips untereinander oder mit weiteren Komponenten verbinden. Dabei muss man zwischen einem Medium und einer einfachen Verdrahtung unterscheiden. Während eine simple Verdrahtung, auf der beliebige Daten übertragen werden können, hinreichend durch die *InterfaceMap* des Chips beschrieben wird, muss ein komplexer Datentransfer, wie z.B. der eines Busses, als Medium modelliert werden. Auf *System*ebene realisiert ein *Medium* eine komplexe (Kabel-) Verbindung zu einem weiteren *Board*, oder repräsentiert eine vorhandene Busschnittstelle z.B. die Schnittstelle eines PCI-Busses, an den ein *Board* angeschlossen werden soll.

Ein Medium ist somit eine komplexe Systemkomponente wie eine Task mit physikalischen Eigenschaften, Struktur (Geometrie) der Schnittstelle und einem Protokoll. Im Sinne der Schnittstellensynthese ist ein *Medium* somit nichts anderes als eine *Task* und besitzt daher die gleichen Parameter zur Beschreibung. Da ein *Medium*, ebenso wie eine Task, an Protokolle gebunden wird, ist die Verbindung von zwei *Tasks, bzw.* zwei *Medien*, ebenso möglich wie ein heterogener Anschluss von einem *Medium* mit einer *Task*. Die folgende Abbildung zeigt den Aufbau des zugehörigen XML Schemas.

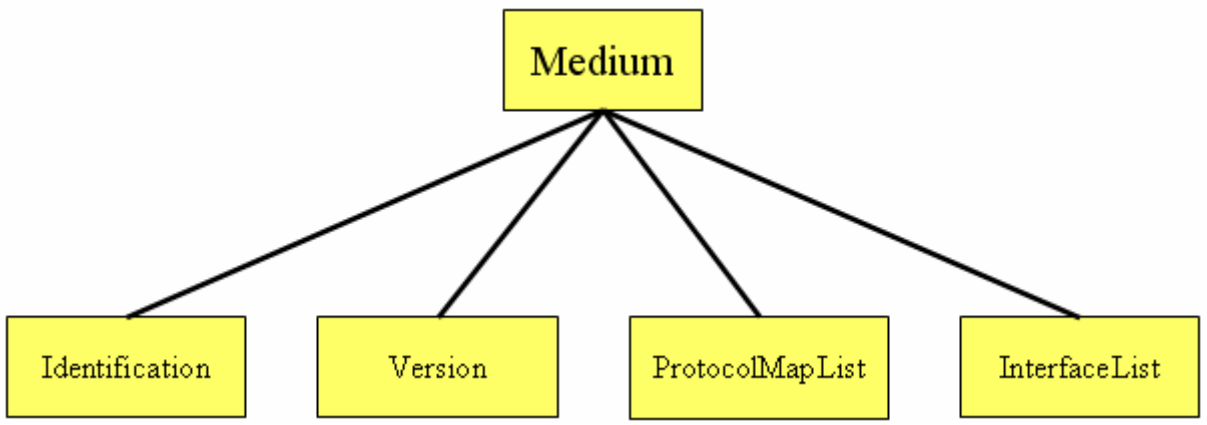

Abbildung 4-6: Aufbau des XML Schemas *Medium.xsd*

Die Parametergruppen *Identification* und *Version* setzen sich aus denselben Parametern wie in den bereits beschriebenen Systemkomponenten zusammen. Aus den verschiedenen Typen von Medien kann im Parameter *Category* ausgewählt werden. Folgende Möglichkeiten sind bis jetzt im IFS Format vorgesehen: *Flachbandkabel*, *USB*, *Twisted Pair*, *Koax* und *Undefined*. Es wurde schon erwähnt, dass ein *Medium* wie eine *Task* behandelt wird. Somit haben die Parametergruppen *ProtocolMapList* und *InterfaceList* dieselben Funktionen wie bei einer *Task*.

Bis auf Namen und zusätzliche Parametergruppen ist die vollständige XML Notation des *Medium*-Schemas der des *Board*-Schemas ähnlich und nicht im Anhang vorhanden.

4.2.7. IFB (Interface Block)

Ein Interface Block ist das Schnittstellenmodul, durch das zwei Schnittstellen verbunden werden können. Die Angabe eines *IFBs* ist theoretisch möglich, aber noch nicht sinnvoll. Wenn die Algorithmen zur Generierung dieser Blöcke implementiert sind, werden die IFB-Angaben innerhalb des IFS Formates automatisch generiert. Dies ist allerdings Zukunftsarbeit und wird im Anschluss an diese Studienarbeit der nächste Schritt sein. Folgende Abbildung zeigt den geplanten Aufbau des XML Schemas *IFB.xsd*.

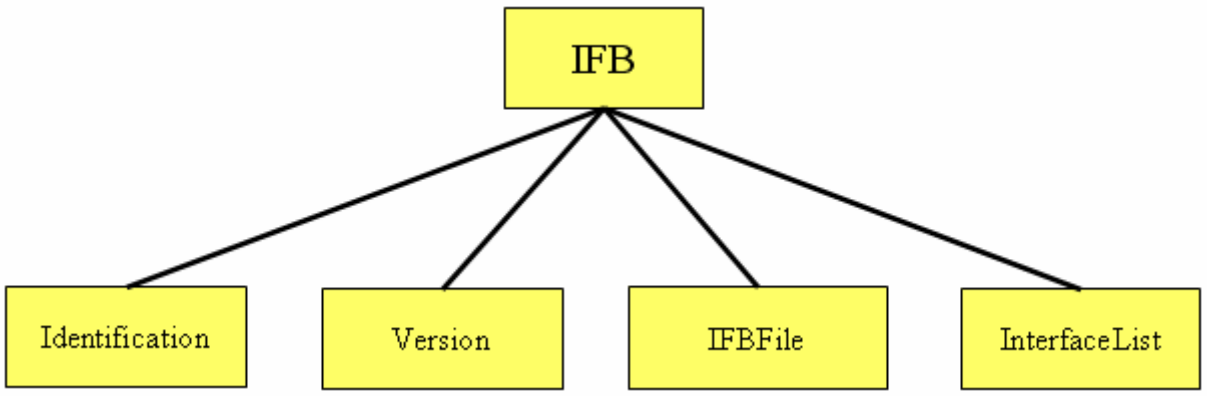

Abbildung 4-7: Struktur des XML Schemas *IFB.xsd*

Die Angaben zu *Identification*, *Version* und *InterfaceList* sind wie in den bereits beschriebenen Komponenten strukturiert. Die zusätzliche Angabe eines zugehörigen *IFBFile*s

ist im Schema vorgesehen. Dabei handelt es sich um das Ergebnis des geplanten Generierungsprozesses.

Die vollständige XML Notation des IFB-Schemas ist der des Board-Schemas bis auf Namen und zusätzliche Parametergruppen sehr ähnlich und ist aus Platzgründen nicht im Anhang vorhanden.

4.3. Interface Description

Ein bedeutender Bestandteil des IFS Formates ist die eigentliche Schnittstellenbeschreibung. Die *Interface Description (IFD)* ist an verschiedenen Stellen im IFS Format aufgehängt. Das Teilschema jeder Komponente der Systemarchitektur unterhalb der *System*ebene beinhaltet eine vollständige Beschreibung der zugehörigen Schnittstellen.. Hinzu kommt eine vollständige Liste der im gesamten System verwendeten Protokolle, die auf *System*ebene verankert ist. Eine Möglichkeit zur Referenzzierung von Protokollen innerhalb der IFD befindet sich allerdings nur in Medium und Task Der dritte Bestandteil der Schnittstellenbeschreibung ist die *Target Platform Description*. Dabei handelt es sich um die Beschreibung des Chips, auf der die generierten *Interface Blöcke* realisiert werden sollen. Im Folgenden werden die genannten Teile der *Interface Description* detailliert erläutert.

4.3.1. Schnittstellen

Die Schnittstellenbeschreibung, die im XML Schema der Komponenten *Board*, *Chip* und *Task* vorkommt hat die folgende Struktur:

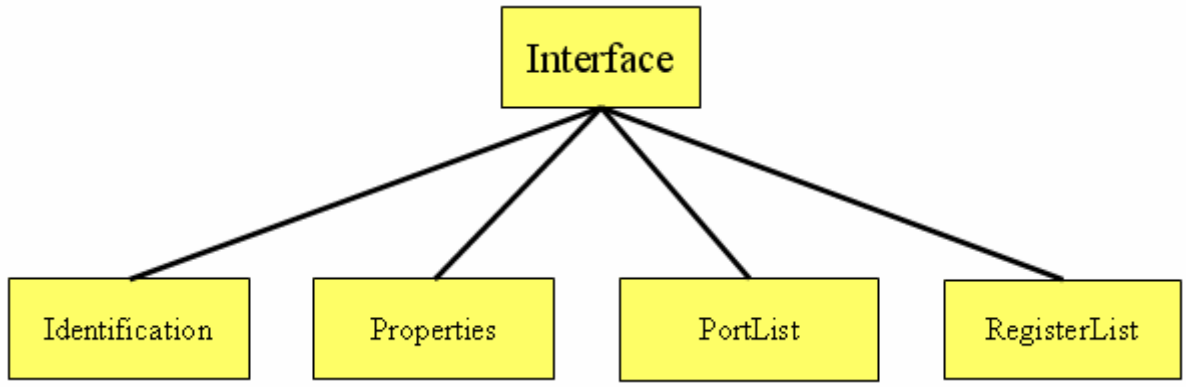

Abbildung 4-8: Struktur des XML Schemas *Interface.xsd*

Die Beschreibung einer Schnittstelle (*Interface*) weicht von den bisher vorgestellten XML Schemata (vgl. VSIA) ab. Sie besteht aus weit mehr Parametern und hat wiederum Sub-Schemata, die eine detaillierte Beschreibung der Schnittstelle ermöglichen. Bevor aber Port und Register, die direkte Bestandteile eines Interfaces sind, näher erläutert werden, folgt zuerst die Beschreibung der Parameter einer Schnittstelle:

Die *Identification* besteht aus den bereits bekannten Parametern *Name (ipq:M..256), ID (ipq:NR1..16)* und *Description (ipq:M..256)*. Zusätzlich ist die Parametergruppe *Connector* optional ein Bestandteil der *Identification*. Diese Untergruppe ist vorgesehen für den Fall, dass ein Stecker Bestandteil der Schnittstelle ist. Folgende Parameter gehören der *Connector*-Gruppe an:

Category (ipq:M..32):
Die *Category* beschreibt den Typen der Steckverbindung. In einer Enumeration stehen folgende Möglichkeiten zur Auswahl: *PIO (Extensionheader)*, *USB_A*, *USB_B*, *IEEE 1394 (Firewire)*, *Wannenstecker*, *Sub-D*, *Undefined*, *Centronics*, *BNC*, *RJ45 (LAN)*, *RJ11 (Phone)*, *N-Connector*, *DIN*, *SCA*, *PS2*, *DIN 41612 (IEC603-2)*, *DIN 41617*, *DIN 41622*, *SCART*, *VG64* und *KEL100*.

Gender (ipq:M..16):
Dieser Parameter beschreibt das Geschlecht des Steckeranschlusses. Die möglichen Optionen sind *MALE* (männlich) und *FEMALE* (weiblich).

Die Parametergruppe *Properties* beinhaltet Parameter, die sich auf die gesamte Schnittstelle beziehen. Es handelt sich dabei um die physikalischen Eigenschaften. Folgende Parameter bilden die Parametergruppe:

MaxFrequency (ipq:NR1..16):
Dieser Parameter bestimmt die maximale Frequenz, mit der Daten über dieser Schnittstelle übertragen werden können.

MaxVoltage (ipq:NR2S..3.3):
Es gibt die maximale Spannung an, die an dieser Schnittstelle angelegt werden darf.

MaxCurrent (ipq:NR2S..3.3):
Es handelt sich hierbei um den maximalen Stromfluss, der an der Schnittstelle möglich ist.

IOTechnologyType (ipq:M..32):
Dieser Parameter bezeichnet die elektronische Technologie, mit der die Datenübertragung realisiert ist. Zur Auswahl stehen hier: *DTL*, *TTL*, *PCI*, *LVDS*, *Differential*, *singleEnded*, *ECL*, *PECL*, *openCollector* und *other*.

MaxWireLength (ipq:NR2S..3.3):
Dieser Parameter beschreibt die maximal mögliche Länge der angeschlossenen Datenbahnen und Kabel. Die Angabe erfolgt in Metern (m).

Die folgenden drei Parameter sind vom Datentyp *ipq:MinTypMaxNR2S..3.3Type*, was bedeutet, dass es sich jeweils um drei Parameter zur Angabe eines minimalen, eines typischen und eines maximalen Wertes handelt.

Latency (ipq:MinTypMaxNR2S..3.3Type):
Diese Parametergruppe beschreibt die Latenz der Signale und wird in Sekunden (s) angegeben.

Jitter (ipq:MinTypMaxNR2S..3.3Type):
Hierbei handelt es sich um den Jitter der Signale in Sekunden (s).

Sensitivity (ipq:MinTypMaxNR2S..3.3Type):
Diese Angabe bezieht sich auf die Sensitivität der Signale und wird in Dezibell (dB) angegeben.

Die exakte XML Notation des Interface-Schemas befindet sich im Anhang unter **8.1.3.1**. Das *Properties*-Schema ist unter **8.1.3.2** nachzulesen.

Ein weiterer Bestandteil der Schnittstellenbeschreibung sind die *PortList* und die *RegisterList*. Die *PortList* beinhaltet eine *Menge* an Ports, aus denen ein *Interface* besteht. Ein *Port* ist gleichzusetzen mit einem Vector in VHDL bzw. einem Bus aus 1 bis n Leitungen mit der gleichen Charakterisierung.

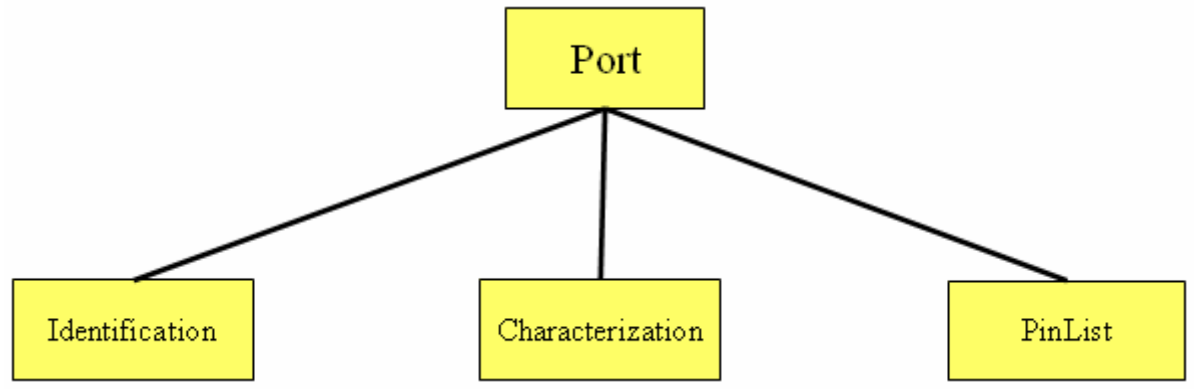

Abbildung 4-9: Struktur des XML Schemas *Port.xsd*

Wiederum besteht die *Identification* aus den Parametern *Name (ipq:M..256), ID (ipq:NR1..16)* und *Description (ipq:M..256)*. Die Parametergruppe Characterization besteht aus den zwei Parametern *VLogicOne* und *VLogicZero*.

VLogicOne (ipq: NR2S..3.3):
Dieser Parameter bestimmt den Spannungspegel, der für eine logische Eins steht. Der vorgegebene Defaultwert ist 3,3V.

VLogicZero (ipq: NR2S..3.3):
Dies ist der Spannungsspiegel, der für eine logische Null steht. Standardwert ist 0,0V.

Die exakte XML Notation des Port-Schemas befindet sich im Anhang unter **8.1.3.3**.

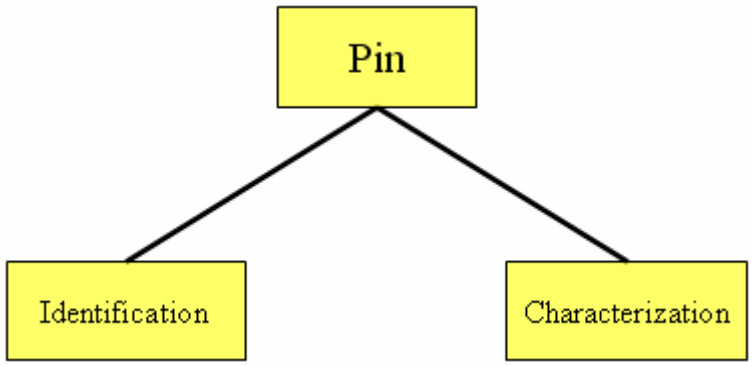

Abbildung 4-10: Struktur des XML Schemas *Pin.xsd*

Da für alle Leitungen eines Ports nicht immer gleiche Bedingungen herrschen, ist dieser noch in Pins unterteilt. In der *PinList* befindet sich eine Menge solcher *Pins*. Dabei handelt es sich um physikalisch vorkommende *Pins* an den Schnittstellen von *Boards* oder *Chips* bzw. um die logischen *Pins* von *Tasks* innerhalb eines *Chips*. Diese Pins werden dann mittels der ProtocolMap auf die logischen Protocolpins eines Protokolls abgebildet.

Zu den Parametern *Name (ipq:M..256)*, *ID (ipq:NR1..16)* und *Description (ipq:M..256)*, die die üblichen Aufgaben übernehmen, kommt noch folgender Parameter innerhalb der *Identification*:

UserID (ipq:M..256):
Dieser ist optional und dient dem Entwickler eigene Bezeichnungen zusätzlich zum Namen und der internen ID zu verwenden.

Die *Characterization* beinhaltet Parameter, die zentrale physikalische Eigenschaften einzelner *Pins* beinhalten.

Direction (ipq:M..32):
Die Richtung, in der der Datenfluss an einem *Pin* möglich ist kann folgende Werte haben: *Input*, *Output*, *Bidirectional*, *Special (GND, VCC)*, und *Undefined*.

Type (ipq:M..32):
Dieser Parameter gibt den Typen eines Pins an. Möglich sind *VCC*, *Ground*, *NC (Not Connected)* und *Undefined*.

Die exakte XML Notation des Pin-Schemas befindet sich im Anhang unter **8.1.3.4**.

Die zweite Hälfte einer Schnittstelle bilden Register und Bit. Diese sind notwendig, wenn eine Komponente einen Datenaustausch über ein oder mehrere Register vollzieht.

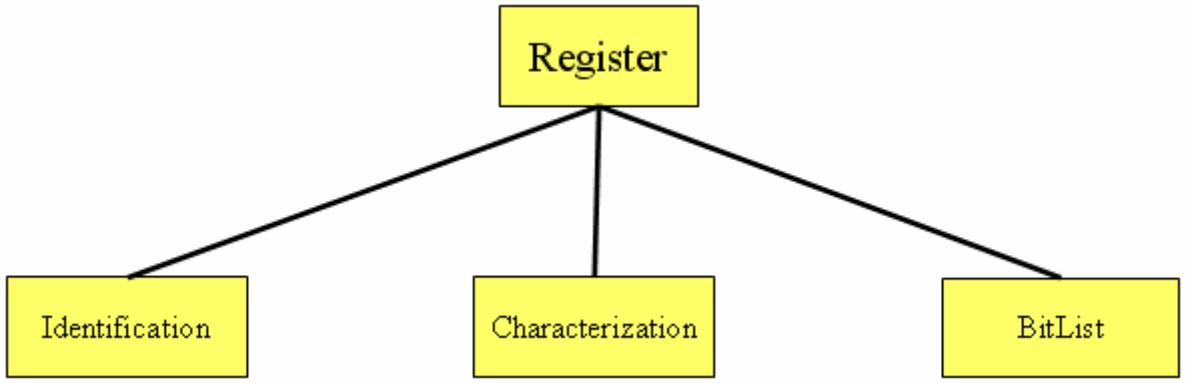

Abbildung 4-11: Struktur des XML Schemas *Register.xsd*

Die *Identification* besteht wie gehabt aus *Name (ipq:M..256)*, *ID (ipq:NR1..16)* und *Description (ipq:M..256)*. Die *Characterization* ist dagegen umfangreicher als die entsprechende Parametergruppe eines Ports:

BaseAddress (ipq:M..32):
Die Basisadresse wird benötigt, um Zugriffe auf das Register zu ermöglichen.

MinimumAllocactionSize (ipq:NR1..16) und **MinimumAllocactionSize (ipq:NR1..16):**
Diese Parameter bestimmen die minimal bzw. maximal mögliche Speicherblockgröße in Bit, die gleichzeitig angesprochen werden kann.

DataTransferType (ipq:M..32):
Durch diesen ihn wird die Zugriffsart bestimmt. Zur Auswahl stehen *Read* (Lesezugriff), *Write* (Schreibzugriff) und *ReadWrite* (Lese- und Schreibzugriff).

AccessableDataSize (ipq:M32):
Dies bestimmt die erreichbare Datengröße. Möglich sind *SingleBit*, was nur den Zugriff auf einzelne Bits ermöglicht und *Register*, wodurch der gleichzeitige Zugriff auf das gesamte Register möglich ist.

Die Parameter *VLogicOne (ipq: NR2S..3.3*) und *VLogicOne (ipq: NR2S..3.3)* entsprechen in ihrer Funktion den gleichnamigen Parametern von *Port*.

Die XML Notation des Register-Schemas gleicht bis auf die Bezeichnungen der Parameter dem Port-Schema und befindet sich aus Platzgründen nicht im Anhang.

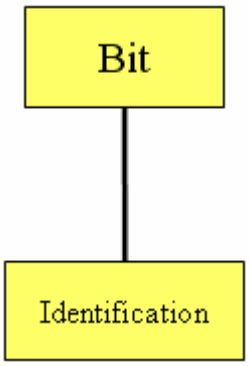

Abbildung 4-12: Struktur des XML Schemas *Bit.xsd*

Ein *Bit* besteht nur aus den Parametern der *Identification*: *Name (ipq:M..256), ID (ipq:NR1..16), UserID(ipq:M..256)* und *Description (ipq:M..256).*

Die XML Notation des Bit-Schemas ist dem Pin-Schema ähnlich und befindet sich aus Platzgründen nicht im Anhang.

4.3.2. Protokolle

Der zweite Bestandteil der Interface Description bezieht sich auf die Protokolle, die im Gesamtsystem vorkommen. Diese werden im IFS Format auf oberster Ebene, also auf der *System*ebene verankert. Folgende Struktur liegt einem *Protocol* zugrunde:

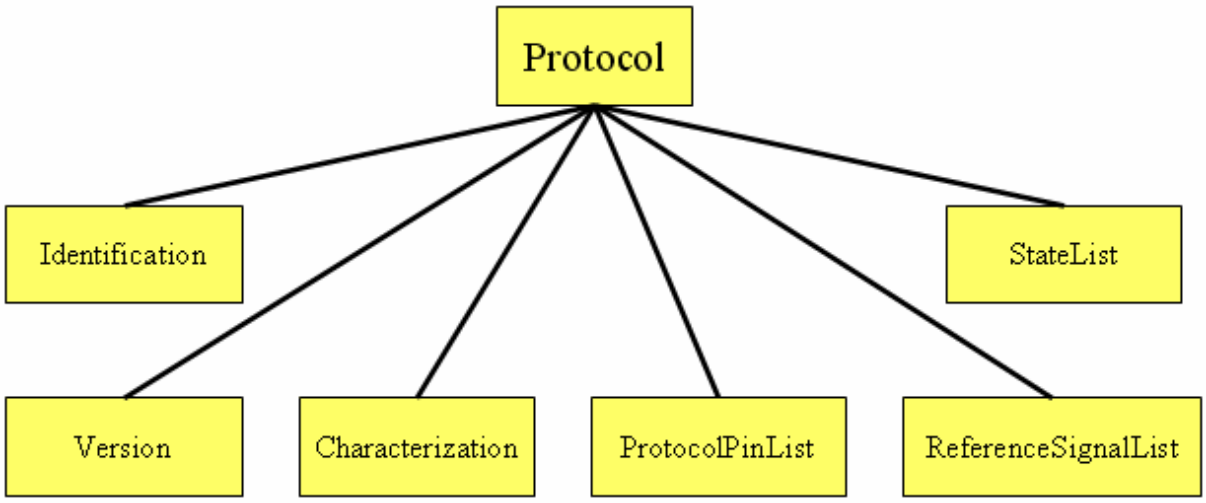

Abbildung 4-13: Struktur des XML Schemas *Protocol.xsd*

Die Parametergruppen *Version* und *Identification* folgen dem bereits bekannten Aufbau. Die *Identification* wird ergänzt durch die *Category*:

Category (ipq:M..256):
Zur Auswahl der Protokollkategorie stehen bisherfolgende Typen: *RS232*, *RS485*, *LVDS*, *Firewire*, *UserDefined* und *Other*.

Die Parametergruppe *Characterization* setzt sich aus folgenden Parametern zusammen.

Control (ipq:M..32):
Dieser Parameter bestimmt, welche Komponente den Transaktionsprozess beginnt. Mögliche Werte sind *Responder*, *Initiator* und *Other*.

Data (ipq:M..32):
Hiermit wird angegeben ob die zugehörige Komponente Daten produziert oder annimmt. Zur Auswahl stehen *Consumer*, *Producer* und *Other*.

UseCase (ipq:M..32):
Die Zuordnung zu speziellen „UseCases“ erfolgt über diesen Parameter mit den Werten *Arbitration*, *Control*, *Data*, *Interrupt* und *Other*.

Flow (ipq:M..32):
Hiermit wird das zugehörige Datenflussmodell angegeben, welches dem Protokoll zugrunde liegt. Zur Auswahl stehen *Persistent*, *Buffered*, *FIFO*, *LIFO*, *Blocking*, *AssignedPortPriority*, *AssignedDataPriority*, *MultiRate*, *Pipelined*, *ExceptionsHandled* und *Other*.

Hinzu kommt die *TransactionTypeList*, in der unterstützte Übertragungsmodi bereitgestellt werden. Es handelt sich um eine Liste des folgenden Parameters:

Method (ipq:M..32):
Die Übertragungsmethode kann folgende Werte annehmen: *transRead*, *transWrite*, *messSense*, *messEmit*, *transOpenChannel*, *transCloseChannel*, *transSynchronize*, *transReset*, *transControl*, *messRead*, *messWrite* und *Other*.

Die exakte XML Notation des Protocol-Schemas befindet sich im Anhang unter **8.1.4.1**.

Die Inhalte der Listen ProtocolPinList, StateList und ReferenceSignalList werden in den folgenden Abschnitten behandelt.

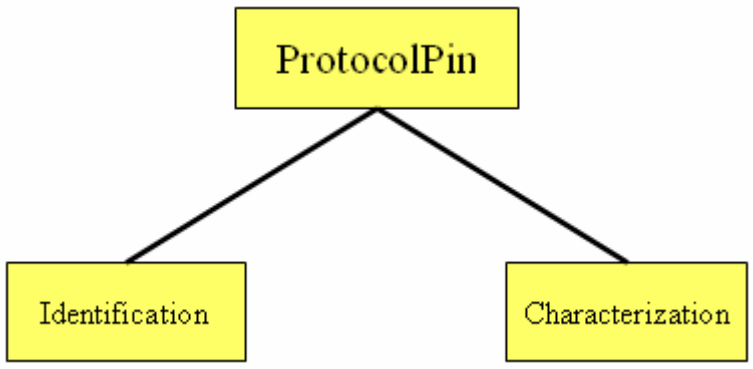

Abbildung 4-14: Struktur des XML Schemas *ProtocolPin.xsd*

Ein *ProtocolPin* ist eine virtuelle Komponente, die vergleichbar mit dem *Pin* einer Schnittstelle ist. Ein *ProtocolPin* ist dabei ein einadriger Kanal, über den Daten übermittelt werden. Die Summe aller *ProtocolPins* eines Protokolls repräsentiert dann die gesamte Übertragung für dieses Protokoll. An die *ProtocolPins* wird mittels eines Referenzmechanismus die Verhaltensbeschreibung des jeweiligen Protokolls gebunden.
Die Parametergruppe *Identification* beinhaltet die üblichen Parameter. *Characterization* besteht aus:

Direction (ipq:M..32):
Hiermit wird die Richtung spezifiziert, in der die Daten übertragen werden. Mögliche Werte sind *Input, Output, Bidirectional, Special (GND, VCC)* und *Undefinded*.

Type (ipq:M..32):
Dieser Parameter bestimmt den Typ eines *ProtocolPins*. Zur Auswahl stehen *Bit*, *Std_Logic*, *VCC, Ground, NC (Not Connected)* und *Undefined*.

Behavior (ipq:M..16):
Das Verhalten eines *ProtocolPins* kann mit folgenden Werten angegeben werde: *Data, Control, Address, Mixed (Data+Control), Status, GND, VCC, NC, Don't Cate* und *Other*.

Die XML Notation des ProtocolPin-Schemas ist dem Pin-Schema ähnlich und befindet sich deswegen nicht im Anhang.

Ein weiterer Bestandteil eines Protokolls ist der Parameter *State*. Zur Beschreibung von Protokollen werden in der IFD endliche Automaten (FSM, finite state machine) genutzt. Mit dem Parameter *State* werden die einzelnen Zustände der FSM, die beim Ablauf eines Protokolls angenommen werden, beschrieben.

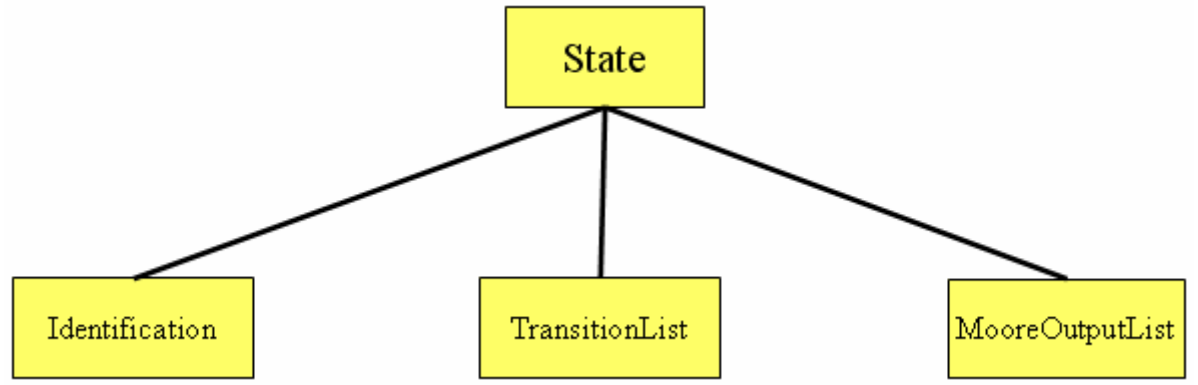

Abbildung 4-15: Struktur des XML Schemas *State.xsd*

Die *Identification* ist wie gehabt. In der *TransitionList* werden die Transitionen verwaltet, mit denen Übergänge zwischen den einzelnen Zuständen modelliert werden. Die *MooreOutputList* beinhaltet Informationen über die Ausgaben, die der modellierte Automat in diesem Zustand (State) liefert:

AutomataOutput.xsd ist das zugehörige XML Schema und besteht aus folgenden zwei Parametern.

ID (ipq:NR1..16):
Diese ID wird verwendet, um einen Pin oder ein Bit zu referenzieren. Diesem wird mit dem nächsten Parameter ein Ausgabewert zugeordnet.

Value (ipq:M..32):
Dies ist der Wert, der einem *Pin* oder *Bit* zugeordnet wird. Zu Auswahl stehen *Logic One*, *Logic Zero*, *High Impedance* und *Not Connected*.

Die exakte XML Notation des State-Schemas befindet sich wegen der Ähnlichkeit zu den bereits vorgestellten Schemata nicht im Anhang.

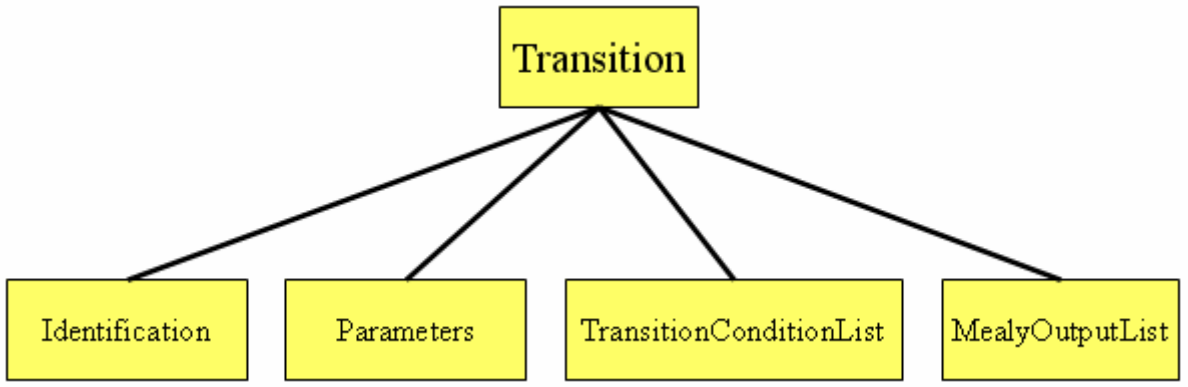

Abbildung 4-16: Struktur des XML Schemas *Transition.xsd*

Mithilfe einer *Transition* wird ein Zustandsübergang innerhalb einer FSM beschrieben. Die Parametergruppe Identification enthält die bereits bekannten Parameter *Name*, *ID* und *Description*. Die Liste *MealyOutputList* beinhaltet wieder Objekte vom Typ *AutomataOutput*. Die restlichen Parameter einer *Transition* sind:

NextStateID (ipq:NR1..16):
Mit dieser *ID* wird der nächste Zustand referenziert zu dem diese *Transition* den Zustandsübergang beschreibt.

TransitionType (ipq:M..16):
Dieser Parameter beschreibt den Typ des Zustandübergangs. Zur Auswahl stehen *Synchronous*, *Asynchronous* und *Deadline*.

Die XML Notation des Transition-Schemas befindet sich wiederum nicht im Anhang.

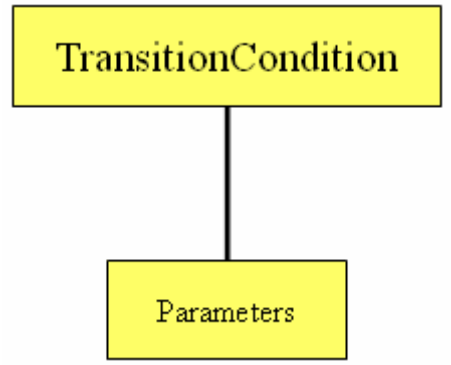

Abbildung 4-17: Struktur des XML Schemas *TransitionCondition.xsd*

Der Inhalt der *TransisionConditionList* wird im Folgenden erläutert. Die gesamte *TransitionConditionList* ist als ein logischer Ausruck zu sehen, der als Bedingung für einen Zustandsübergang wahr werden muss. Hierzu werden die *TransitionConditions* hintereinandergeschrieben als logischer Ausdruck angesehen. Eine *TransitionCondition* besteht aus folgenden Parametern.

BracketOpen (ipq:M..16):
Hiermit kann ein Ausdruck geklammert werden. „(“ ist der einzig mögliche Wert.

Operator (ipq:M..16):
Dieser Parameter ist optional. Die einzelnen *TransitionConditions* der *TransitionConditionList* werden mithilfe con Operatoren miteinander verknüpft. Zur Auswahl stehen *AND, OR, XOR, NAND, NOR* und *XNOR*.

TriggeredSignalID (ipq:NR1..16):
Diese ID wird verwendet um das zugehörige Trigger-Signal referenziert. Dabei kann es sich um einen *ProtocolPin* oder ein *ReferenceSignal* handeln.

Die Parametergruppe *Trigger* gibt eine Bedingung an, die erfüllt sein muss, um den Zustandsübergang anzustoßen. Hierbei ist die Auswahl zwischen einem bestimmten Wert (Value), den ein Signal annehmen muss, und einem Zeitpunkt (Time) zu treffen. Im Fall von *Value* sind folgende Parameter zu setzen.

ValueSignalSource (ipq:M..16):
Es legt fest, ob die Signalquelle eine *Clock* oder ein *Signal* ist.

ValueExpectedValue (ipq:M..256):
Dies ist der Wert, der den Zustandsübergang auslöst. Möglich sind *Low Value*, *High Value*, *Falling Edge, Rising Edge* und *Signal Changed*.

Im Fall von Time bilden folgende Parameter die Gruppe:

TimeSignalSource (ipq:M..16):
Hiermit wird angegeben, ob es sich um *Timer, Global_Time* oder *Deadline* handelt.

TimeExpectedValue (ipq:NR2S..3.3):
Dieser Parameter dient der Angabe des Wertes, der zum Zustandsübergang benötigt wird.

Mit folgendem Parameter wird eine TransitionCondition abgeschlossen.

BracketOpen (ipq:M..16):
Hiermit kann ein Ausdruck geklammert werden. „(“ ist der einzig mögliche Wert.

Die exakte XML Notation des TransitionCondition-Schemas befindet sich im Anhang unter **8.1.4.2**.

Den letzten wichtigen Teil der Protokollbeschreibung bildet die *ReferenceSignalList*. In dieser Liste befinden sich Objekte der folgenden Art.

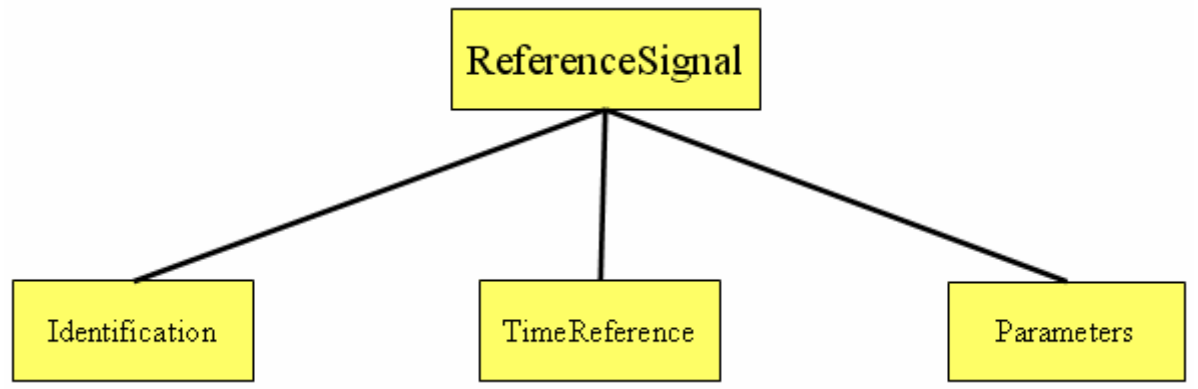

Abbildung 4-18: Struktur des XMl Schemas *ReferenceSignal.xsd*

Die *Identification* besteht aus den üblichen Parametern. Die Parametergruppe *TimeReference* ist eine Auswahl (eine *Choice* in XML) zwischen *Clock*, *TimerOrDeadline* und *GlobalDate*, wodurch die Art des ReferenceSingnals bestimmt wird. Der einzige direkte Parameter eines *ReferenceSignals* ist *TargetPlatformRefClockID*.

TargetPlatformRefClockID (ipq:NR1..16):
Mithilfe dieses Parameters wird die zugehörige *TargetPlatformDescriptionClock* referenziert.

Wenn die Auswahl der *TimeReference* auf *Clock* fällt, sind folgende Parameter vorgesehen.

Frequency (ipq:NR2S..3.3):
Dieser Parameter bestimmt die Frequenz des Referenzsignals.

StartValue (ipq:M..16):
Durch diesen Parameter erfolgt die Zuweisung eines Anfangswertes. Möglich sind **Logic One** und **Logic Zero**.

Die Gruppe *Phaseshift* ist Bestandteil der *Clock* und besteht aus folgenden Parametern:

ReferenceClockID (ipq:NR1..16):
Hiermit wird eine Referenz auf die Clock der Target Platform gesetzt.

ReferenceClockType (ipq:M..32):
Trifft die Auswahl zischen einer Clock der Target Plattform oder einer anderen Referez-Clock. Mögliche Werte sind daher entsprechend *Real Clock (TPD)* und *Reference Clock*.

Phase (ipq:M..16):
Diese Angabe gibt die Phasenverschiebung an *-0˚*, *-45˚ (Pi / 4)*, *-90˚ (Pi / 2)*, *-135˚ (3 Pi / 4)* und *-180˚ (Pi)*.

Die Auswahlmöglichkeit *TimerOrDeadline* besteht nur aus einem Parameter:

MaxTime (ipq:NR2S..3.3)
Diese Angabe liefert eine Zeitschranke.

Die Option *GlobalDate* besteht aus folgenden Parametern.

CycleTime (ipq:NR2S..3.3):
Mithilfe dieses Parameters wird die Periodendauer (Master Cycle Time) einer zyklischen Zeitbasis beschrieben.

Zusätzlich gehört noch eine Liste von *TimeEvents* zur Gruppe *GlobalDate*. Diese bestehen jeweils aus folgenden Parametern.

EventTime (ipq:NR2S..3.3):
Gibt innerhalb der CycleTime einzelne Zeitpunkte (Events) an, zu denen etwas passieren kann.

EventValue (ipq:M..16):
Legt das entsprechende Ereignis (Low- oder High-Pegel) fest, das zum Zeitpunkt *EventTime* am Signal des GlobalDates anliegen soll.

Die exakte XML Notation des ReferenceSignal-Schemas befindet sich im Anhang unter **8.1.4.3**.

4.3.3. Target Platform Description

Der dritte und letzte Teil der *Interface Description* ist die *Target Platform Description*, kurz *TPD* genannt. Die *TPD* ist im IFS Format auf *Chip*ebene angebracht, d.h. sie ist ein Teil der *Chip*-Beschreibung.
Mithilfe der *TPD* wird ein jeder *Chip* soweit beschrieben, wie es für die Implementierung der späteren *Interface Blöcke* (*IFB*) notwendig ist. Dabei handelt es sich bisher um nutzbare Ressourcen eines Chips sowie die vorhandenen Clock-Netzwerke.

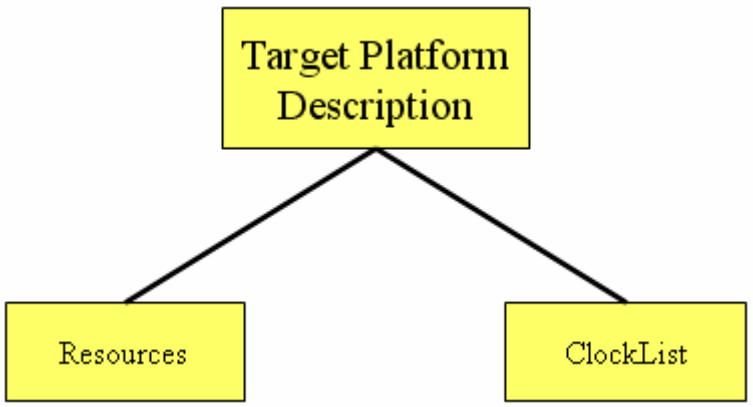

Abbildung 4-19: Struktur des XML Schemas *TargetPlatformDescripion.xsd*

Die Parametergruppe *Resources* beinhaltet folgende Parameter:

Category (ipq:M..256):
Dieser Parameter ermöglicht die Einordnung eines Chips in folgende Kategorien: *FPGA*, *ASIC* und *Processor*.

Quantity (ipq:NR1..16):
Hiermit wird die Anzahl der verfügbaren Resourcen angegeben.

Unit (ipq:M..16):
Dies ist die Einheit, in der die Resourcen angegeben werden. Zur Auswahl stehen *CLBCount*, *GateCount* und *Memory*.

In der *ClockList* befindet sich die Liste der auf dem Chip vorhandenen Clock.

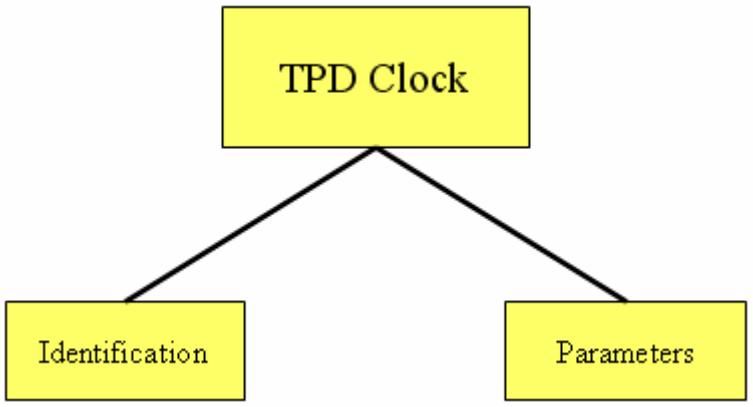

Abbildung 4-20: Struktur des XML Schemas *TargetPlatformDescriptionClock.xsd*

Die Parametergruppe Identification besteht aus *Name*, *ID* und *Description* mit den üblichen Datentypen. Die restlichen Parameter einer *TPDClock* sind folgende:

Frequency (ipq:NR1..16):
Dieser Parameter beschreibt die Frequenz der *Clock*.

Skew (ipq:MinTypMaxNR2S..3.3):
Gibt den *Skew* der entsprechenden *Clock* an, was die maximale Verzögerung einer Takleitung angibt.

Jitter (ipq:MinTypMaxNR2S..3.3):
Gibt den *Jitter* der entsprechenden *Clock* an. Hier wird die maximale Abweichung vom Standardwert des Taktsignals angegeben.

Network (ipq:NR1..16):
Dieser Parameter beschreibt das Clock-Netzwerk zu der entsprechenden *Clock*.

Wegen der einfachen Struktur der Schemata *TargetPlatformDescription* und *TargetPlatformDescriptionClock* befinden sich diese nicht im Anhang.

4.4. Maps

Die so genannten „Maps“ beinhalten einen weiteren Teil des IFS Formates. In ihnen werden die physikalischen Eigenschaften des gesamten Systems zum einen in Relation untereinander, zum anderen in Relation zu den verwendeten Protokollen gebracht.

4.4.1. Verdrahtung

Die Verdrahtung der Schnittstellen untereinander erfolgt über die *InterfaceMaps*. Dabei kann eine Systemkomponente mit einer anderen Systemkomponente verbunden werden. Des

Weiteren kann hier eine Systemkomponente mit der höher gelegenen Systemkomponente verdrahtet werden.

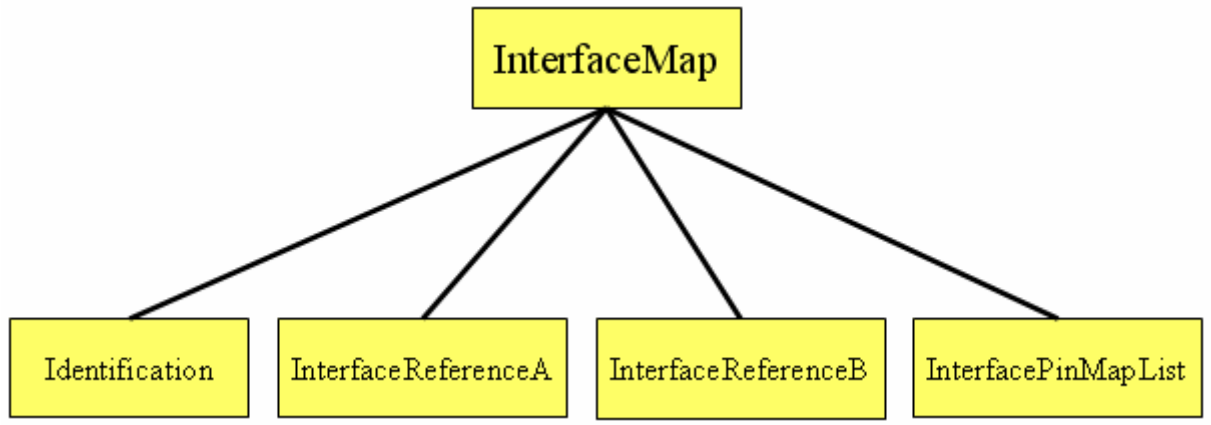

Abbildung 4-21: Struktur des XML Schemas *InterfaceMap.xsd*

Die Parametergruppe *Identification* beinhaltet die Parameter *Name, ID* und *Description* mit den bekannten Datentypen. Die Gruppen *InterfaceReferenceA* und *InterfaceReferenceB* sind die Referenzen auf die beiden zu verbindenden Komponenten. In der *InterfacePinMapList* werden diese Komponenten dann *Pin* für *Pin* miteinander verdrahtet. Die folgenden Parameter bilden die Gruppen *InterfaceReferenceA* und *InterfaceReferenceB*.

DeviceCategoryA / DeviceCategoryB (ipq:M..16):
Mit dieser Kategorie wird die Art der Systemkomponente angegeben, die verdrahtet werden soll. Möglich sind je nach aktueller Ebene *Board, Chip, Task, Medium* und *IFB.*

DeviceIDA / DeviceIDB (ipq:NR1..16)
Durch diese Parameter wird die Komponente, die verdrahtet wird, identifiziert.

InterfaceIDA / InterfaceIDB (ipq:NR1..16)
Mit dieser ID wird die Schnittstelle, die verdrahtet wird identifiziert.

Die exakte XML Notation des InterfaceMap-Schemas befindet sich im Anhang unter **8.1.5.1.**

Die InterfacePinMapList beinhaltet Objekte vom Typ InterfacePinMap. In einer InterfacePinMap werden zwei Pins mit einander verbunden. Dazu sind folgende Parameter notwendig: *PortIDA*, *PinIDA*, *PortIDB* und *PinIDB*. Auf diese Weise wird der *Port* der einen Schnittstelle A eindeutig identifiziert, dann in diesem der *Pin.* Dieser wird dann mit dem *Pin* der Schnittstelle B verbunden.

PortIDA und PortID (ipq:NR1..16):
Mit diesen beiden Parametern werden die beiden *Ports* eindeutig identifiziert, dessen *Pins* miteinander verdrahtet werden sollen.

PinIDA und PinID (ipq:NR1..16):
Hiermit werden die beiden *Pins* eindeutig identifiziert, die miteinander verbunden werden sollen.

4.4.2. Schnittstellen und Protokolle

Die Abbildung der einzelnen Pins der Schnittstellen auf die verwendeten Protokolle ist notwendig, um eine Verbindung mehrerer Komponenten überhaupt zu ermöglichen. Ohne eine Zuordnung zu Protokollen, wäre keine Kommunikation beschreibbar. Durch eine Verbindung von physikalischem Pin und Protokollverhalten wird das Verhalten des Systems mit der Struktur in Verbindung gebracht. Dies geschieht mithilfe der *ProtocolMap*.

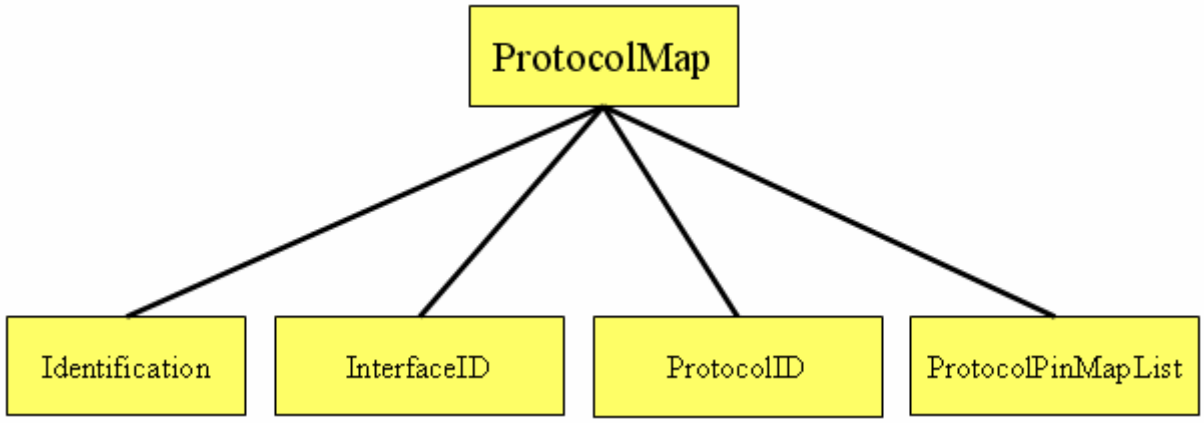

Abbildung 4-22: Struktur des XML Schemas *ProtocolMap.xsd*

Aufgrund der Identification können einer *ProtocolMap* wieder *Name*, *ID* und *Description* gegeben werden.

InterfaceID (ipq:NR1..16):
Mit diesem Parameter wird die Schnittstelle (*Interface*) bestimmt, die mit einem Protokoll verknüpft werden soll.

ProtocolID (ipq:NR1..16):
Dieser legt das Protokoll fest.

In der *ProtocolPinMapList* werden die *Pins* von *Interface* und Protokoll miteinander verbunden. Dabei hat eine *ProtocolPinMap* folgende Parameter:

InterfacePortID(ipq:NR1..16):
Hiermit wird ein *Port* der Schnittstelle ausgewählt.

InterfacePinID(ipq:NR1..16):
Dies bestimmt eindeutig den *Pin* innerhalb des ausgewählten *Ports*.

ProtocolPinID (ipq:NR1..16):
Mit diesem Parameter wird der anzubindende *ProtocolPin* spezifiziert.

5. Der IFS Editor

Die Definition des IFS Formates mittels XML ermöglicht den Einsatz von verschiedenen Anwendungen zur Eingabe und Bearbeitung der Daten. Um diese Operationen benutzerfreundlich zu gestalten, ermöglichen Programme wie z.B. *XML Spy* – vertrieben von der Altova GmbH – die Erzeugung gültiger Instanzen des IFS Schemas. Jedoch können diese Programme den Umfang des IFS Formates nur schwer bewältigen. Die Struktur und logische Zusammenhänge des IFS Formates, wie z.B. in den Teil-Schemata *Protocol* und *InterfaceMap*, werden in einem solchen allgemeinen Editor nicht berücksichtigt.

Um diesen Problemen entgegenzuwirken und ein sinnvolles modellbasiertes Design zu ermöglichen, wurde der IFS Editor entwickelt. Dieser unterstützt eine strukturierte Eingabe mit integrierter Konsistenzprüfung sämtlicher Parameter, die zur Schnittstellensynthese erforderlich sind. Dieser Editor bietet ebenfalls Möglichkeiten zur Wiederverwendung bereits eingegebenener Teilarchitekturen an und erhöht damit zusätzlich die Performance des Designers. Eine kommende Ausbaustufe des Editors wird zusätzlich die Nutzung von IP-Beschreibungen im IPQ-Format unterstützen.
In den folgenden Kapiteln werden der IFS Editor und die zugrunde liegenden Datenstrukturen näher erläutert.

5.1. Der Aufbau der Software

Die Entwicklung des IFS Editors erforderte die Entwicklung einer Datenstruktur, die einen Zugriff auf eine XML Instanz des IFS Schemas ermöglicht. Die folgende Abbildung zeigt eine detaillierte Aufteilung der Software in funktionale Schichten.

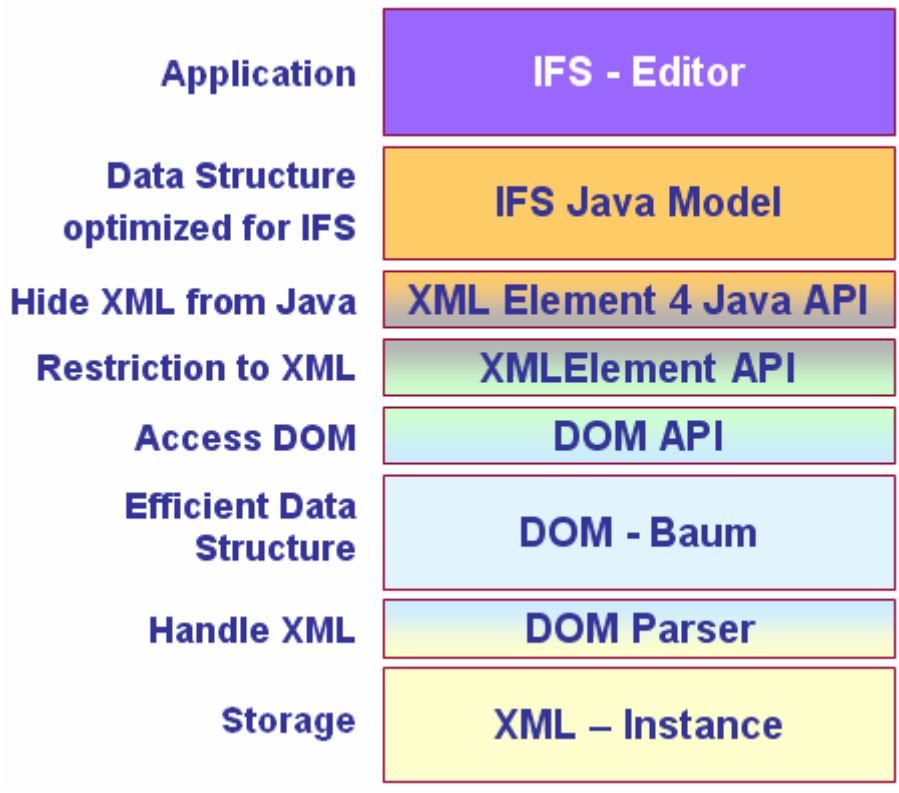

Abbildung 5-1:Struktur der Software

Während am oberen Ende der IFS Editor steht, befindet sich in der unteren Schicht die XML Instanz, in der sich sämtliche Daten der Beschreibung als Datei befinden. Dazwischen befinden sich folgende Schichten:

XML Instanz:
Eine XML Instanz ist die Datei, in der die Daten einer Beschreibung durch das IFS Format gehalten werden.

DOM Parser:
Der DOM Parser ermöglicht das Einlesen einer XML Instanz mit dem XML Schema. Dabei wird überprüft, ob die Daten innerhalb der Instanz dem Aufbau des XML Schemas folgen oder von diesem abweichen.

DOM Baum:
Der DOM Baum ist die vom DOM Parser aufgebaute Datenstruktur, in der die eingelesenen Daten einer XML Instanz gehalten werden. Diese sind in Form von DOM Knoten in dem Baum zugreifbar.

DOM API:
Die DOM API ist die Standard API zum Zugriff auf XML Daten innerhalb eines DOM Baumes. Diese wird hier ausschließlich von der XML Element API aufgerufen.

XML Element API:
Hiermit wird der Zugriff auf einen DOM Baum in Form von XML-Dokumenten ermöglicht. Das Setzen und Abfragen der Daten innerhalb des Baumes, wie auch Operationen zum Laden und Speichern einer Instanz, erfolgen über diese XML Element API.

XML Element 4 Java:
Diese Schicht kapselt die interne Datenstruktur von den Klassen des IFS Editors. Es werden für jeden Parameter Methoden für den Zugriff auf folgende Werte angeboten:

- Standardwert (Default-Value)
- Datentyp (Type)
- Bezeichnung des Parameters (Tag)
- Einheit der Daten (Unit)
- Aufzählung angebotener Werte (Enumeration)
- Kommentar / Tool-Tip (Comment)

Diese API ist eingeführt worden, um einen sicheren Umgang mit den Daten für die Synthese zu gewährleisten. Auf diese Weise können Paradigmen einer objektorientierten Programmiersprache wie Typsicherheit, Methodenkapselung, Objekthierarchie und Vererbung ausgenutzt werden. Damit erfolgt eine Abstraktion von der relativ niedrigen Knoten-Ebene eines Baumes auf eine höhergelegene Objektsicht.

IFS Java Model:
Das IFS Java Model (IFDS, Interface Data Structure) ermöglicht einen einfachen Umgang mit den IFS Parametern. Die eigenlichen Daten, die auch in der XML Instanz gespeichert werden, sind allerdings im DOM Baum gehalten. Während die Daten im DOM Baum als einheitliche Knoten ohne semantische Zusammenhänge gehalten werden, ermöglicht das IFS Java Model eine Zuordnung der Daten zu den einzelnen Komponenten, aus denen das System besteht. Zu jedem Bestandteil des XML Schemas ist eine entsprechende Java-Klasse entwickelt worden.

5.2. Die graphische Oberfläche

Die graphische Oberfläche (*GUI, graphical user interface*) folgt repräsentiert den Aufbau des IFS Formates. Für jedes Teil-Schema wurde eine separate Sicht (*Page*, *View*) entworfen, in dem die Parameter des zugehörigen XML Schemas editiert werden können.

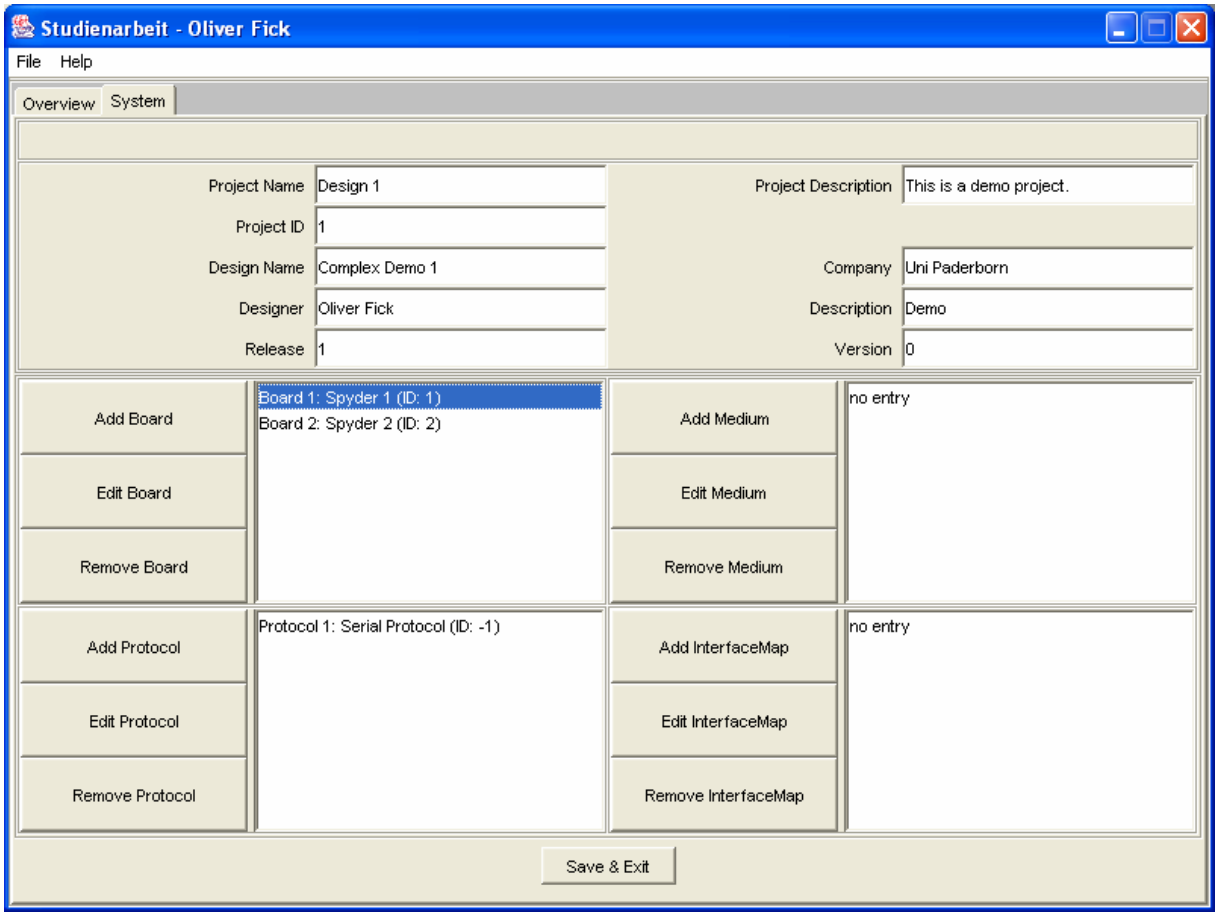

Abbildung 5-2: Der IFS Editor

Abbildung 5-2 zeigt den *System-View*. In ihm werden die aktuellen Werte der Systemkomponente *System* angezeigt, wie sie schon in **4.2.2** vorgestellt wurde. Diese Sicht (*View*) bildet den Einstiegspunkt für einen Designer, der mithilfe des IFS Editors eine Beschreibung der für die Schnittstellensynthese notwendigen Parameter erstellen will.

5.2.1. Verwendete Techniken

In den folgenden Abschnitten wird die Funktionsweise der einzelnen *Views* näher erläutert. Dabei handelt es sich um mehrfach verwendete Techniken, die in allen *Views* wieder zu erkennen sind.

5.2.1.1. Einheitlicher Aufbau

Jedes Sub-Schema des IFS Formates wird durch einen *View* vertreten, der einem einheitlichen Aufbau folgt. Im oberen Bereich befinden sich immer Textfelder und Auswahlboxen zur Eingabe der Parameter, wie z.B. die Werte der häufig auftretenden Parametergruppen *Identification* und *Version*.

Der Zugriff auf Listen, die Bestandteil vieler Teilschemata sind, erfolgt über die Buttons und Listenfelder im mittleren und unteren Bereich. In Abbildung 5-2 befinden sich beispielsweise in der *BoardList* zwei *Boards* mit den Bezeichnungen „Spyder 1“ und „Spyder 2“. Mittels „Add Board“, „Edit Board“ und „Remove Board“ werden entweder bestehende *Boards* bearbeitet (Edit), gelöscht (Remove) oder ein neues *Board* hinzugefügt (Add). In gleicher Weise befinden sich Buttons und Listenfelder in vielen der anderen *Views*, in denen Listen von Systemkomponenten verwaltet werden.
Der untere Abschnitt eines jeden *Views* zeigt einen einzelnen Button, „Save & Exit“. Mit diesem Button wird der aktuelle *View* verlassen und die eingegebenen Daten dieses *Views* akzeptiert und gespeichert.

5.2.1.2. Überprüfung der Eingaben

Alle vorgenommenen Eingaben werden nach Verlassen eines Textfeldes überprüft. Diese Überprüfung beschränkt sich derzeit auf die verwendeten Datentypen. Diese sind innerhalb des IFS Formates definiert und werden mithilfe einer dafür vorgesehenen Java-Klasse überprüft. Im Falle einer fehlerhaften Eingabe werden Fehlerfenster der folgenden Art angezeigt, und die letzte Eingabe rückgängig gemacht.

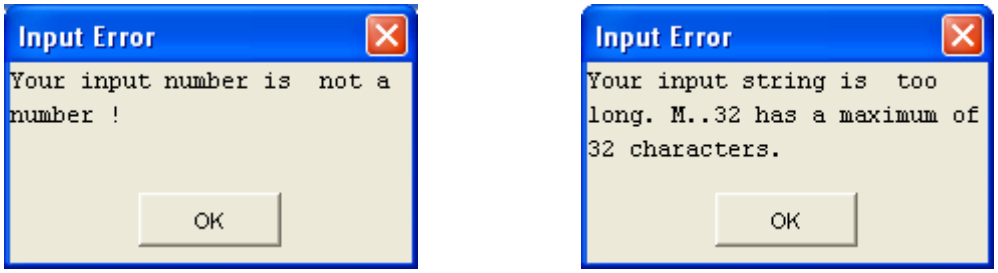

Abbildung 5-3: Fehlerfenster

Die eingegebenen Daten werden auf folgende Eigenschaften getestet. Neben der Art der Eingabe, in diesen Fällen Buchstaben und/oder Ziffern, wird die Länge der Eingabe mit den in den Datenformaten definierten Beschränkungen verglichen.

5.2.2. Systemarchitektur und Zielplattformbeschreibung

Die Struktur sämtlicher *Views* folgt dem bereits erläuterten Aufbau. Eine weiterführende Erklärung bleibt daher hier erspart. Die angezeigten Daten werden in den zugehörigen Unterkapiteln von **4.2** mitsamt ihrer Datentypen erklärt. Zur Systemarchitektur gehören folgende *Views*: *System*, *Board*, *Chip*, *Task*, *Medium*. Der enge Zusammenhang zwischen Systemarchitektur und Zielplattformbeschreibung (*TPD*) erforderte die Integration der *TPD* in den *Chip-View*. Folgende *Views* zeigen die Systemkomponenten und die Zielplattform:

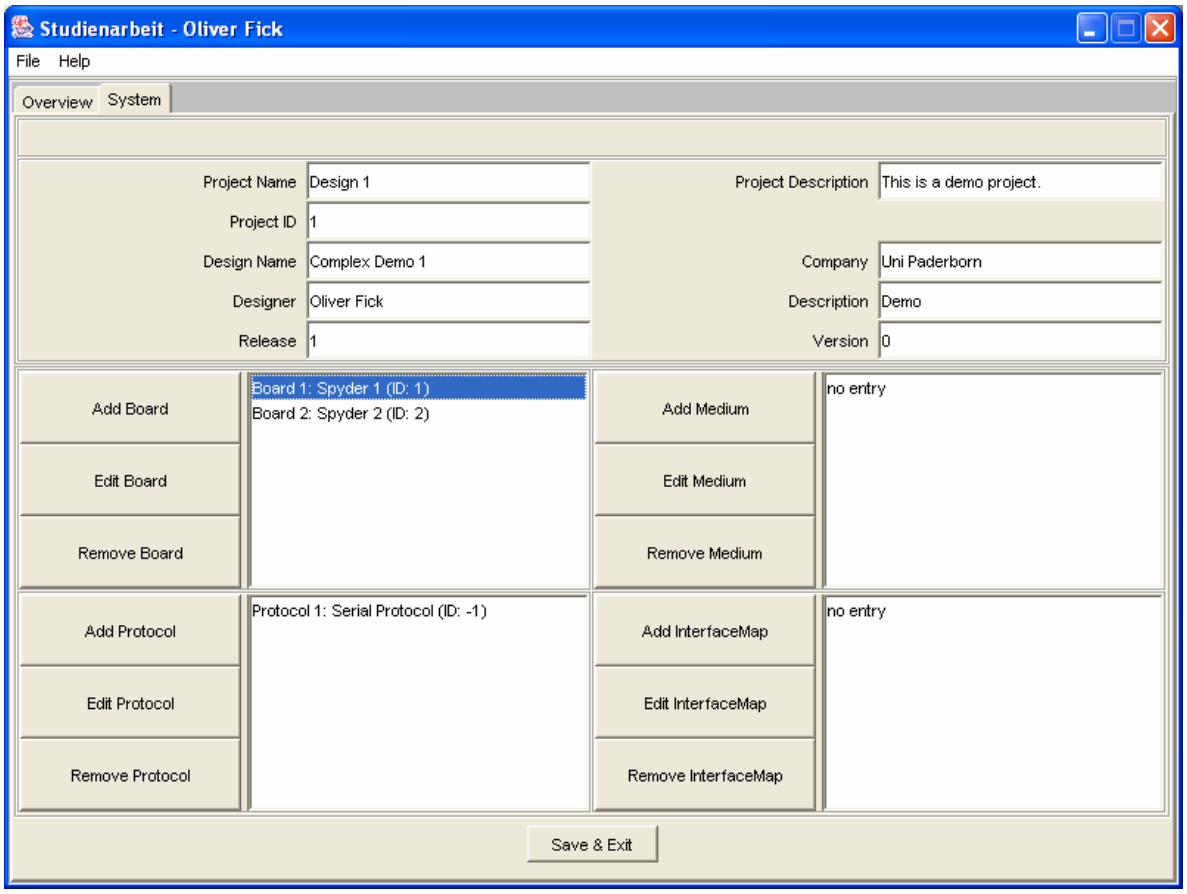

Abbildung 5-4: Der *System-View*

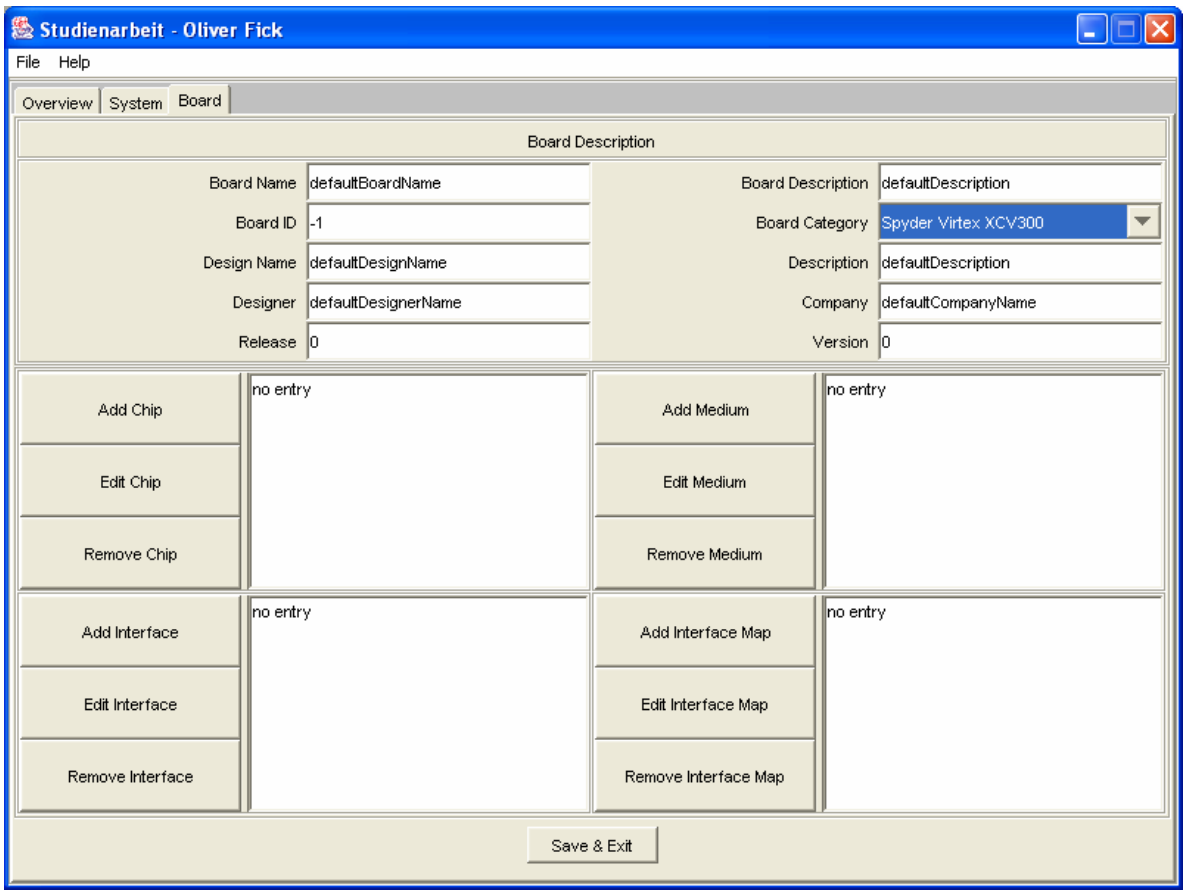

Abbildung 5-5: Der *Board-View*

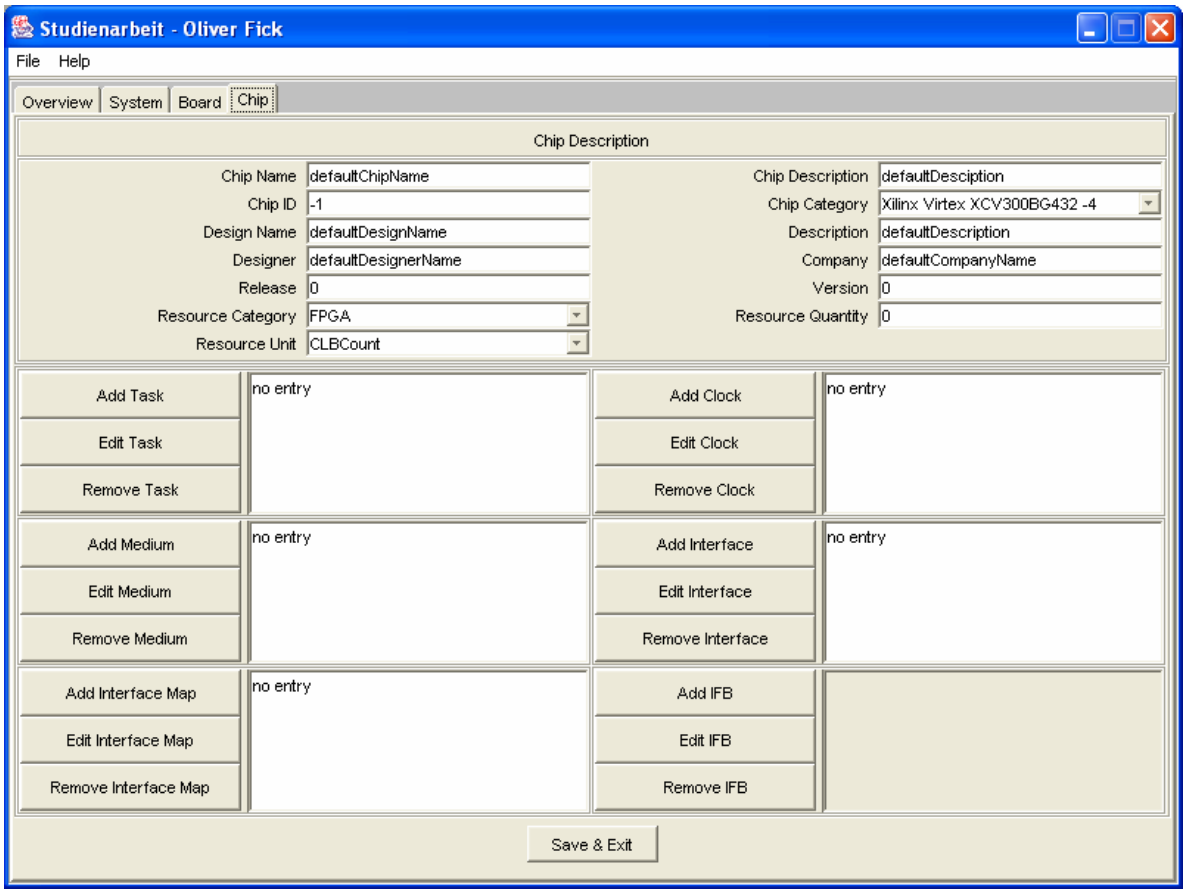

Abbildung 5-6: Der *Chip-View*

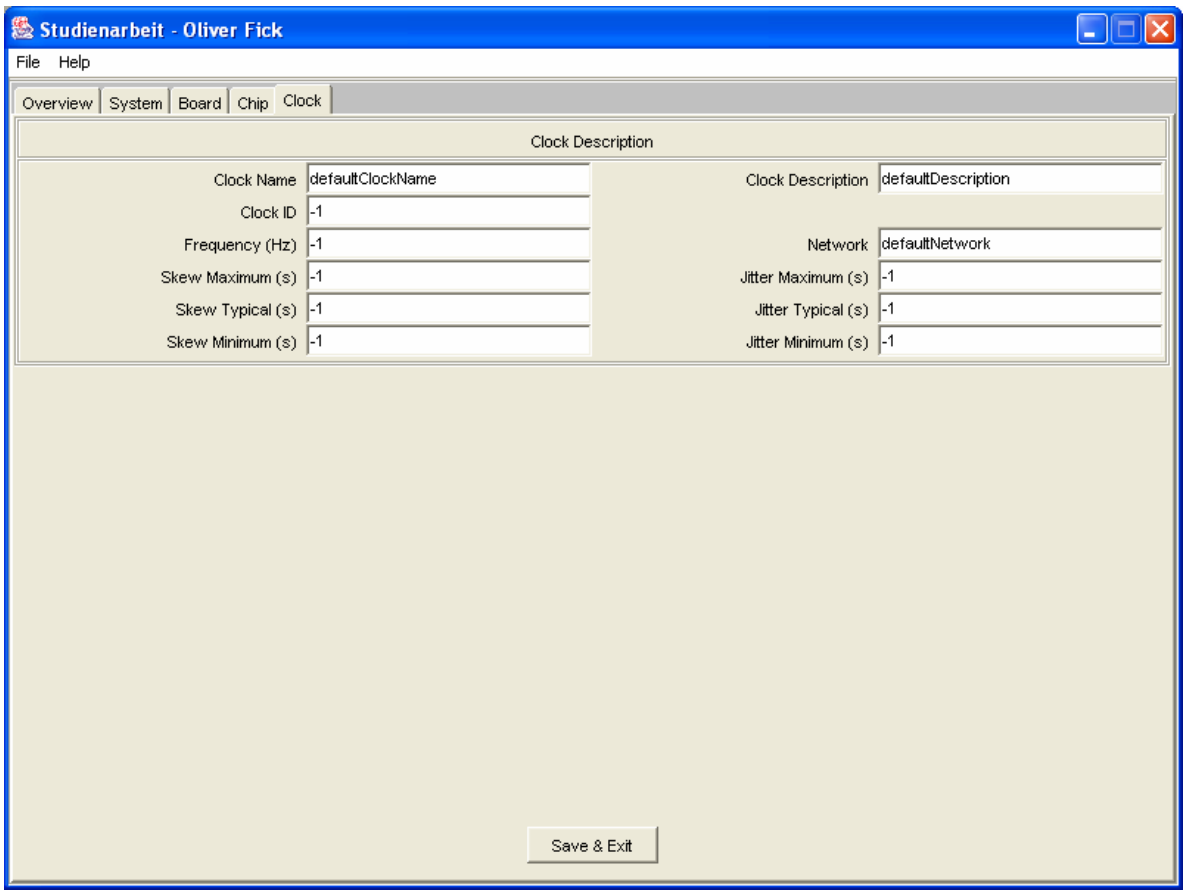

Abbildung 5-7: Der *TPD-Clock-View*

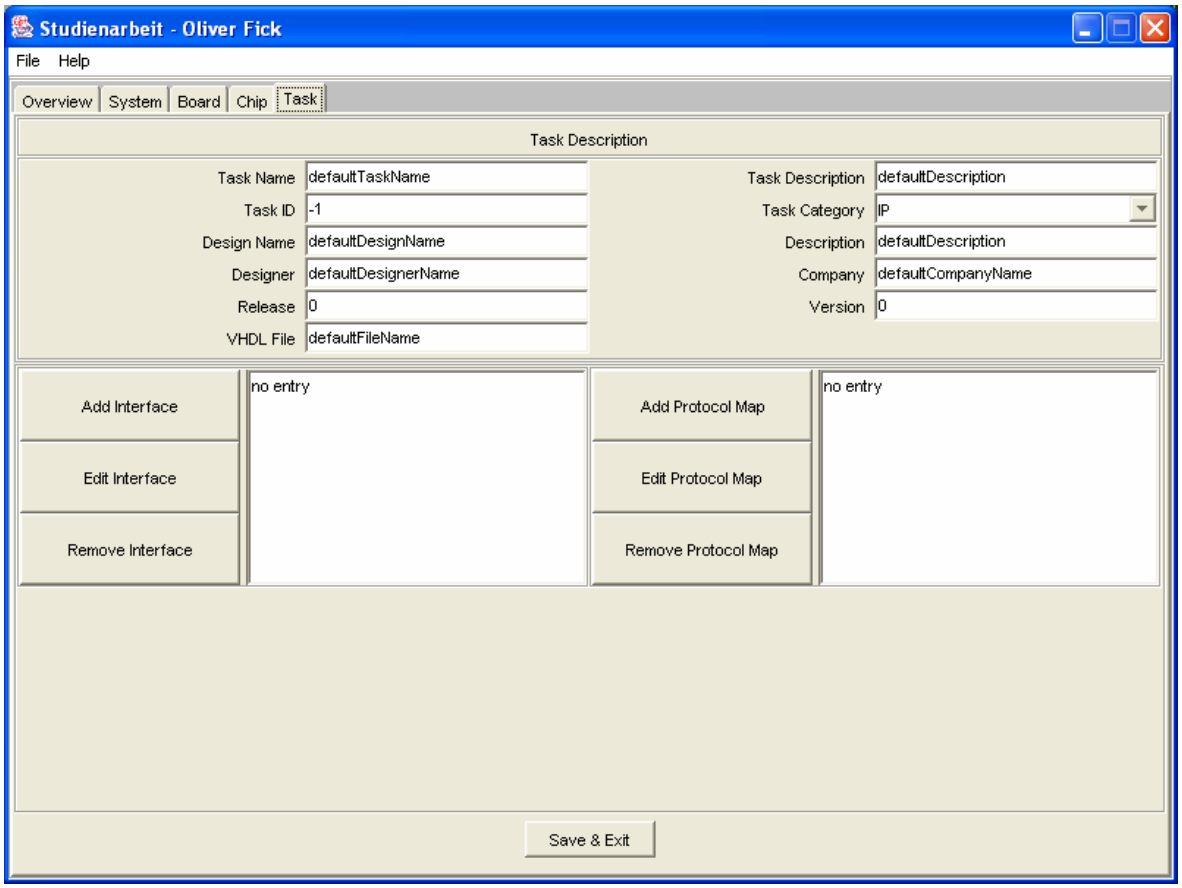

Abbildung 5-8: Der *Task-View*

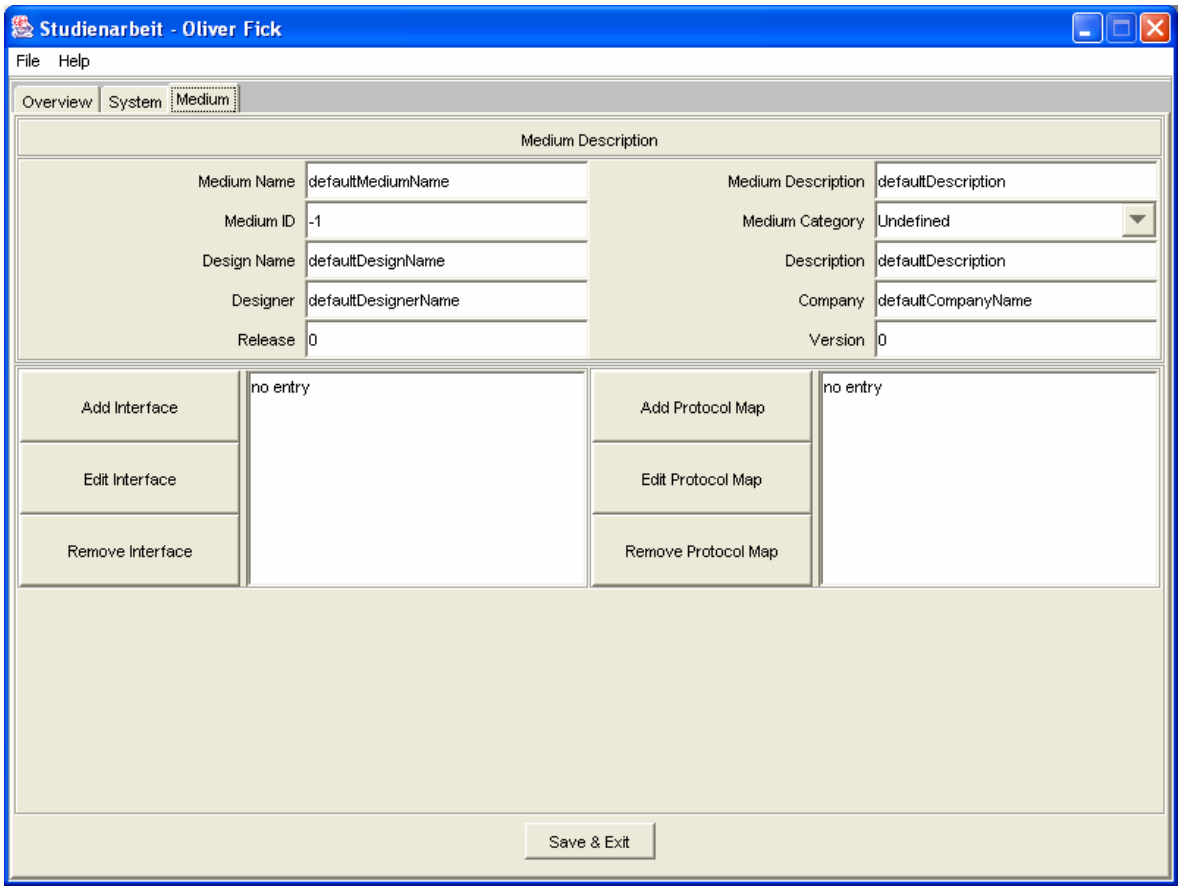

Abbildung 5-9: Der *Medium-View*

Wie aus Abbildung 5-6 hervorgeht, wurde die Zielplattformbeschreibung (*TPD*) in die *Chip*-Sicht integriert. Die *TPD* beinhaltet für die Schnittstellensynthese und die automatisierte Erzeugung von *IFBs* wichtige Informationen über den *Chip*, auf dem eventuell benötigte *IFBs* realisiert werden. Der zugehörige *TPD-Clock-View* gehört ebenso zur Zielplattform-beschreibung.

5.2.3. Schnittstellenbeschreibung

Die Beschreibung der Schnittstellen (*Interface*) erfolgt im *Interface-View*. Damit logisch verbunden sind der *Port*- und der *Register-View*. Diese *Views* folgen dem üblichen Aufbau. Im unteren Teil des *Port*- bzw. *Register-Views* wurden die Darstellungen der *PinList* bzw. *BitList* integriert. Die Parameter einzelner *Pins* bzw. *Bits* werden hier angezeigt. Die Anzeige dieser Parameter innerhalb eines anderen *Views* beruht darauf, dass die Anzahl von *Pins* und *Bits* die Anzahl von *Boards*, *Chips* oder *Tasks* um ein Vielfaches übersteigt. Die Beschreibung einer längeren *PinList* wäre ansonsten mit häufigen Sichtwechseln verbunden.
Ein Zugriff auf die Daten wurde in die Sicht auf einen Port integriert. Er erfolgt mithilfe der Maus und des Kontext-Menüs, welches durch einen Rechtsclick auf einen Pin geöffnet wird. Die üblichen Operationen wie „Add Pin“ und „Remove Pin“ sind auf diesem Weg verfügbar. Die zusätzliche Option zum Kopieren einzelner *Pins* bzw. *Bits* ermöglicht ein schnelles Erstellen umfangreicher *PinLists*.

Die folgenden Abbildungen zeigen die zur Schnittstellenbeschreibung gehörenden *Views*.

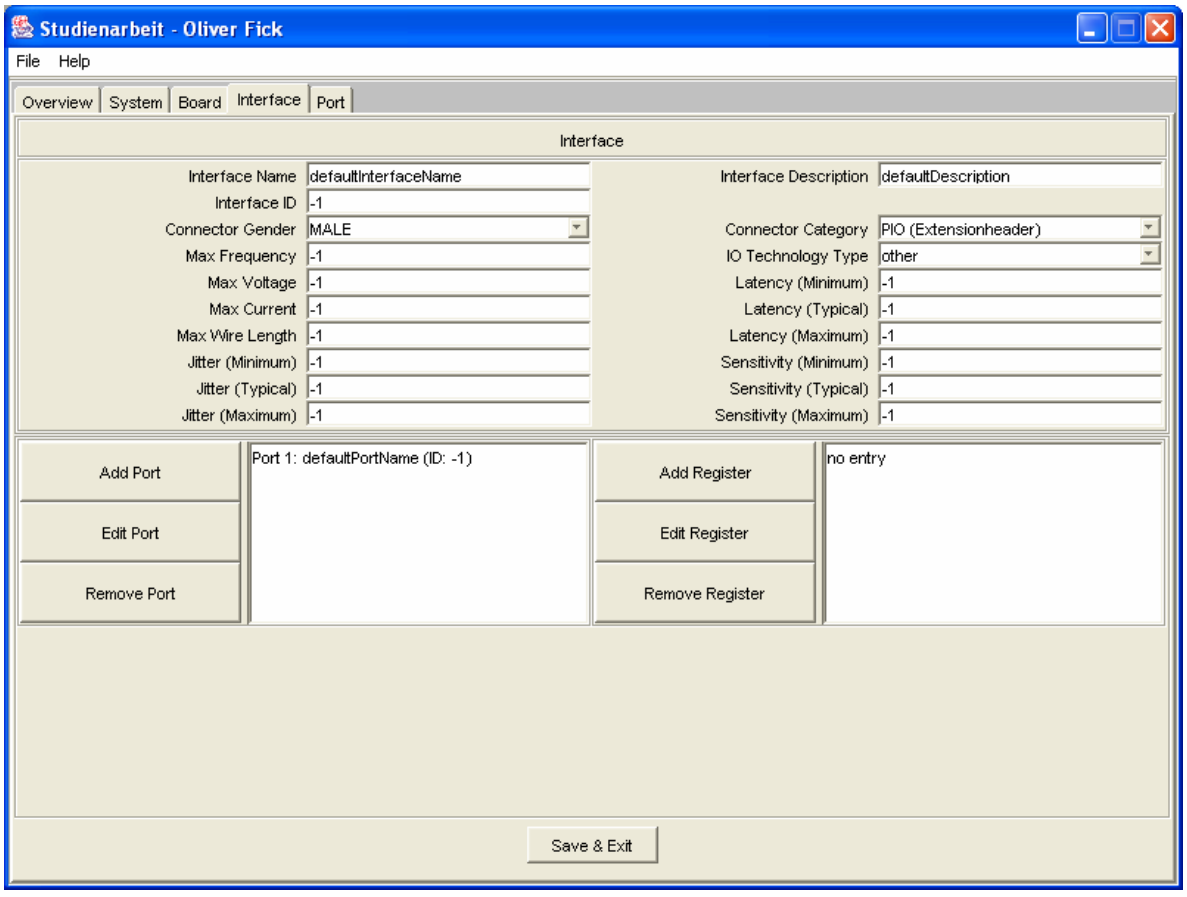

Abbildung 5-10: Der *Interface-View*

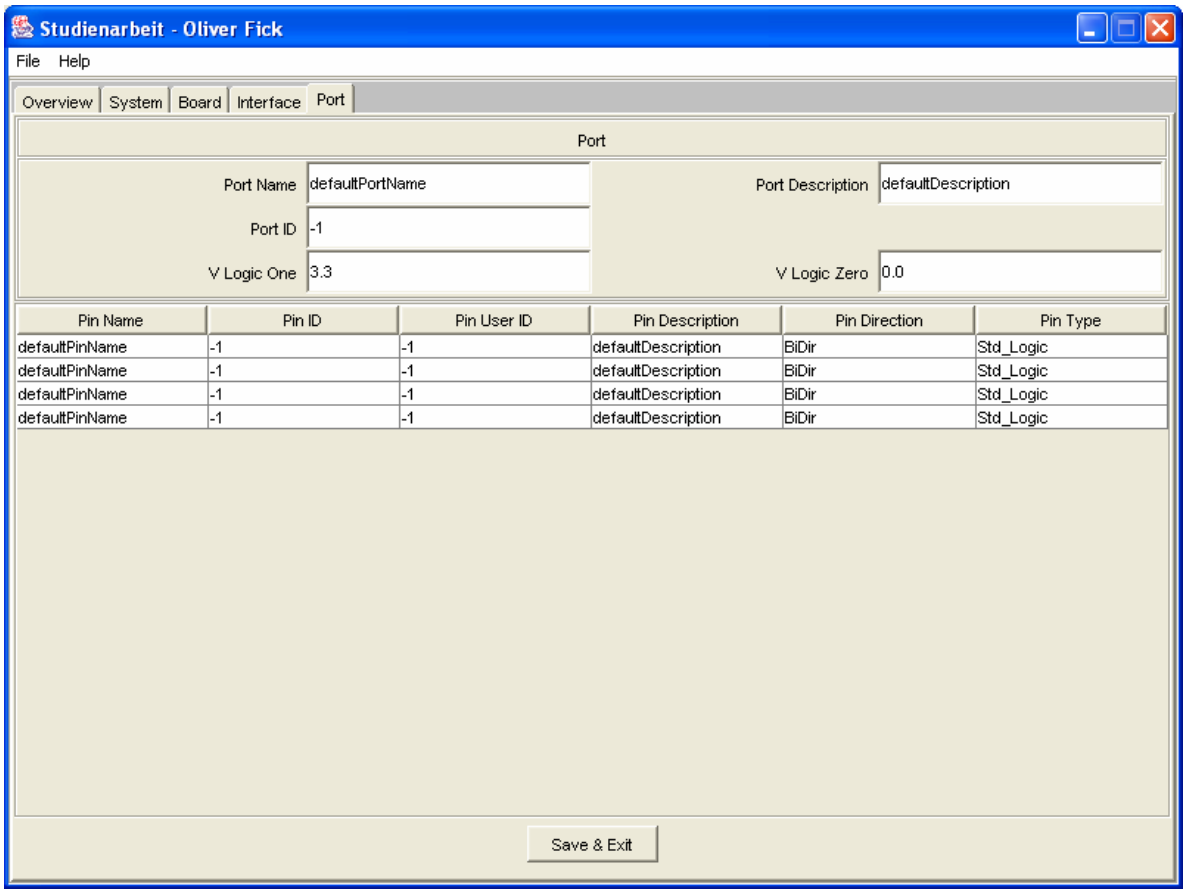

Abbildung 5-11: Der *Port-View*

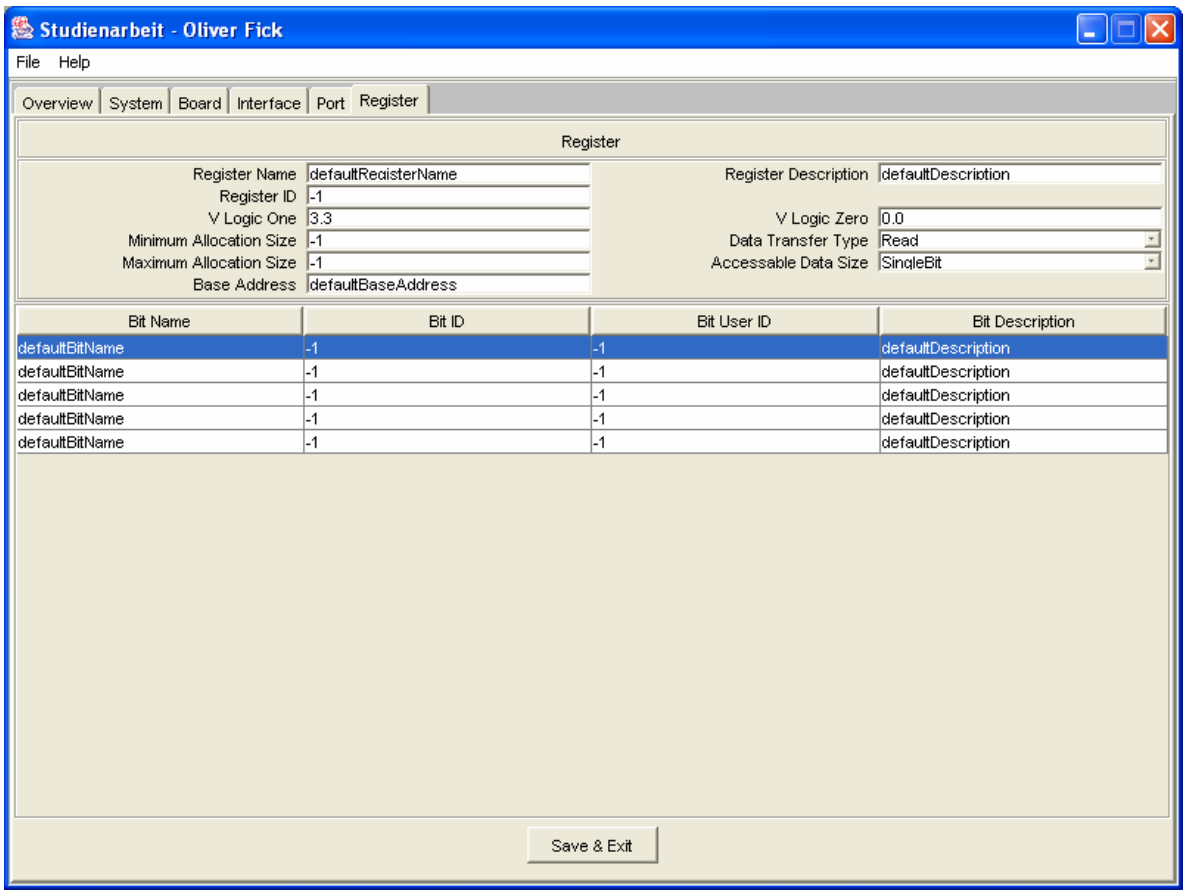

Abbildung 5-12: Der *Register-View*

5.2.4. Protokollbeschreibung

Die Beschreibung der Protokolle erfolgt über mehrere *Views*. Im *Protocol-View* ist wiederum die Liste der *ProtocolPins* integriert. Der Zugriff auf einzelne *ProtocolPins* erfolgt wie schon bei der *Pin-* und *BitList* mithilfe der Maus.

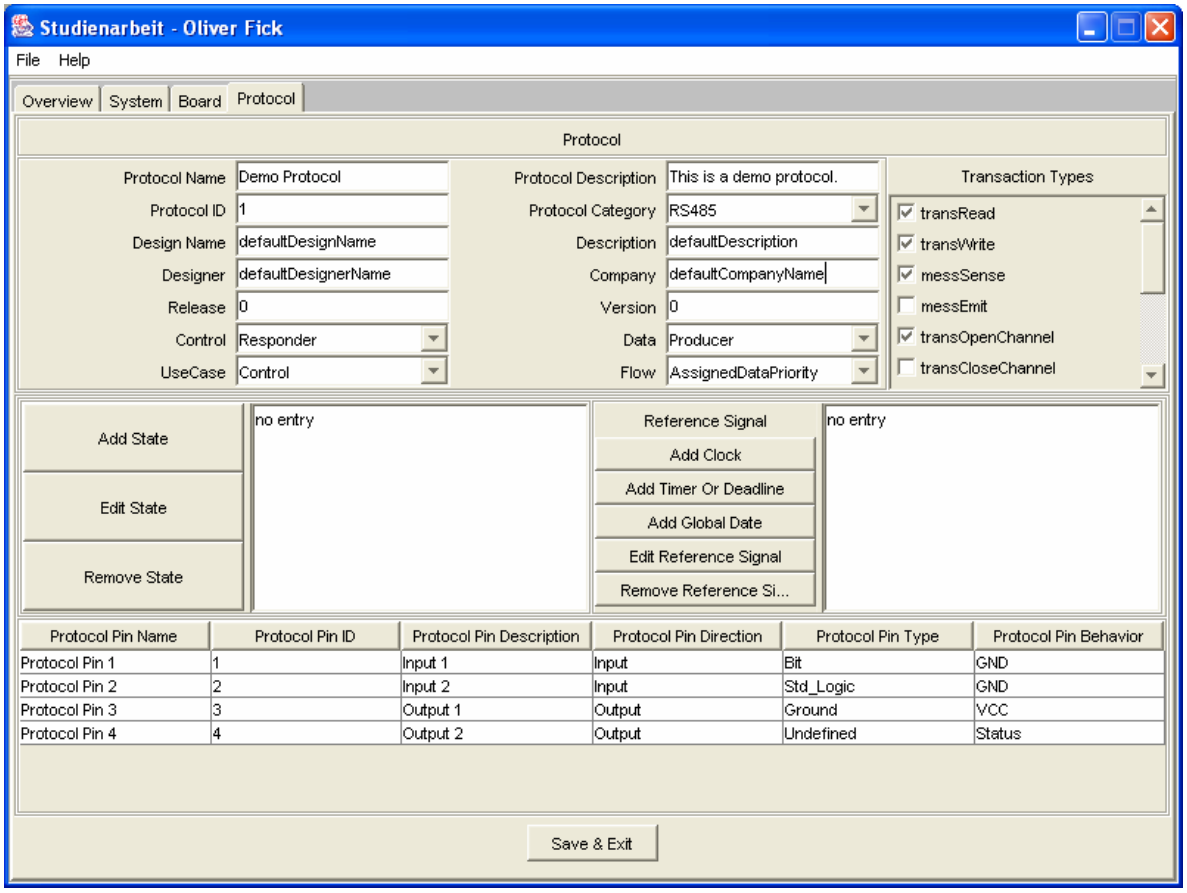

Abbildung 5-13: Der *Protocol-View*

Die *TransactionTypeList* wurde nicht auf dem üblichen Weg realisiert. Da es sich bei einem *TransactionType* um einen einzelnen Parameter handelt und dieser nur eine begrenzte Anzahl von Werten annehmen kann, wurde die *TransactionTypeList* als eine Menge von Checkboxen implementiert.
Der Zugriff auf *StateList* folgt dem üblichen Weg. Die *ReferenceSignalList* ist jedoch auch abweichend implementiert. Wie schon in **4.3.2** beschrieben, beinhaltet ein *ReferenceSignal* drei Parametergruppen, zwischen denen eine Auswahl getroffen werden muss. Diese Auswahl geschieht schon im *ProtocolView*. Mithilfe der Buttons „Add Clock“, „Add Timer Or Deadline“ und „Add Global Date“ wird jeweils ein *ReferenceSignal* erzeugt und der entsprechende *View* geöffnet.
Die folgenden Abbildungen zeigen die zugehörigen *Views* der Sub-Schemata von *Protocol*.

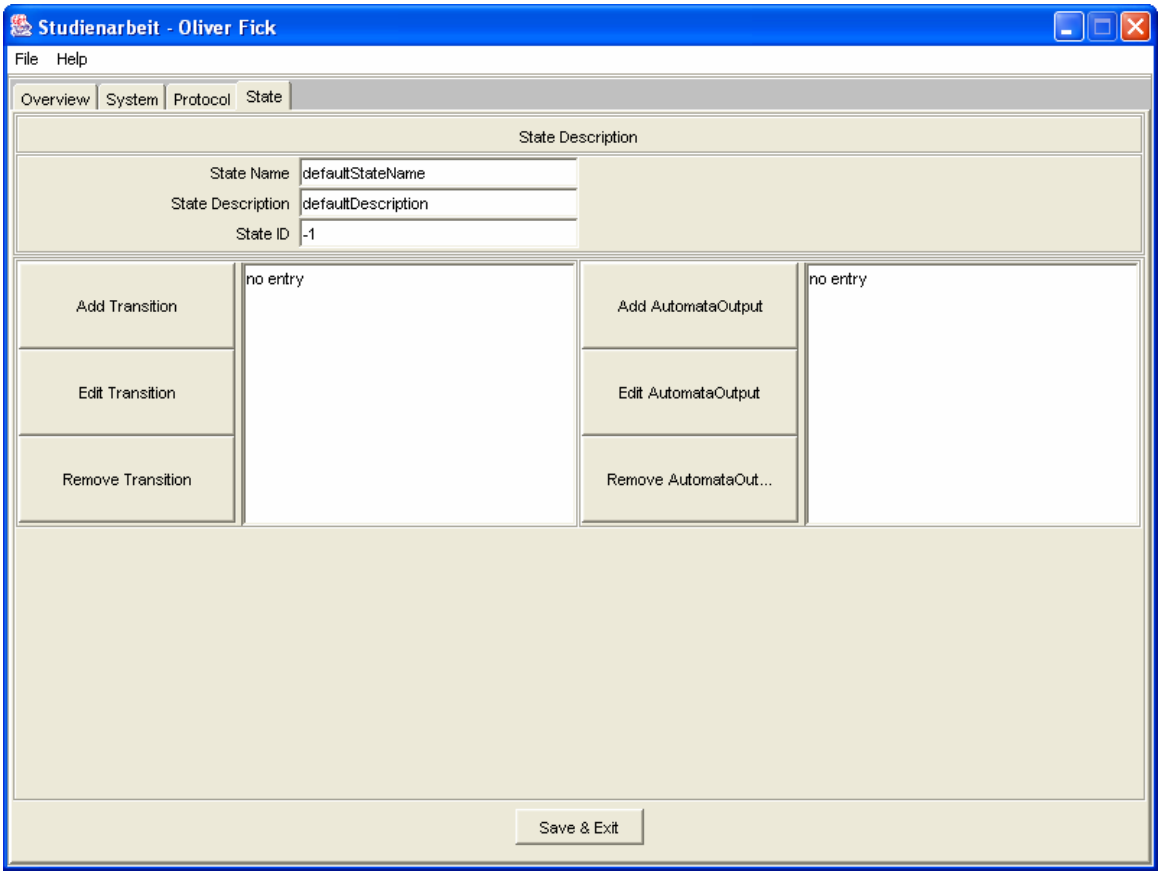

Abbildung 5-14: Der *State-View*

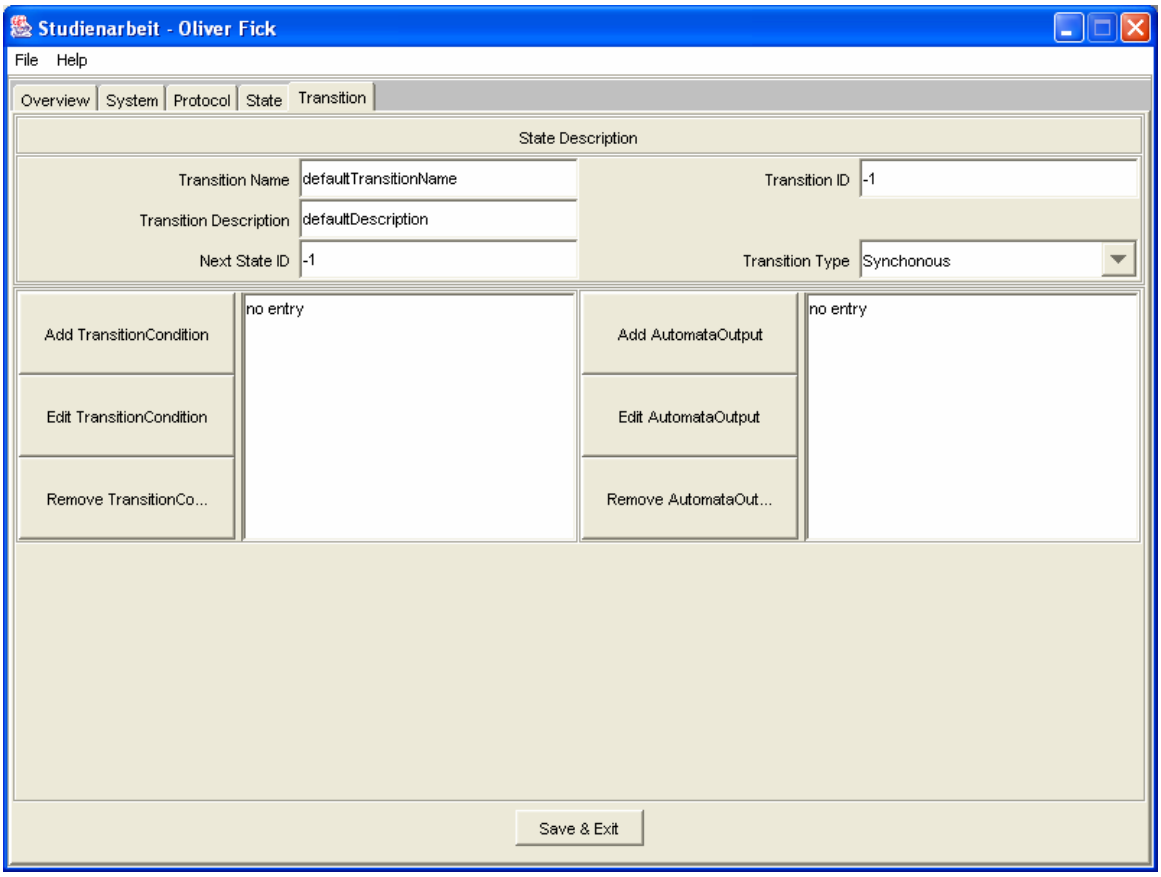

Abbildung 5-15: Der *Transition-View*

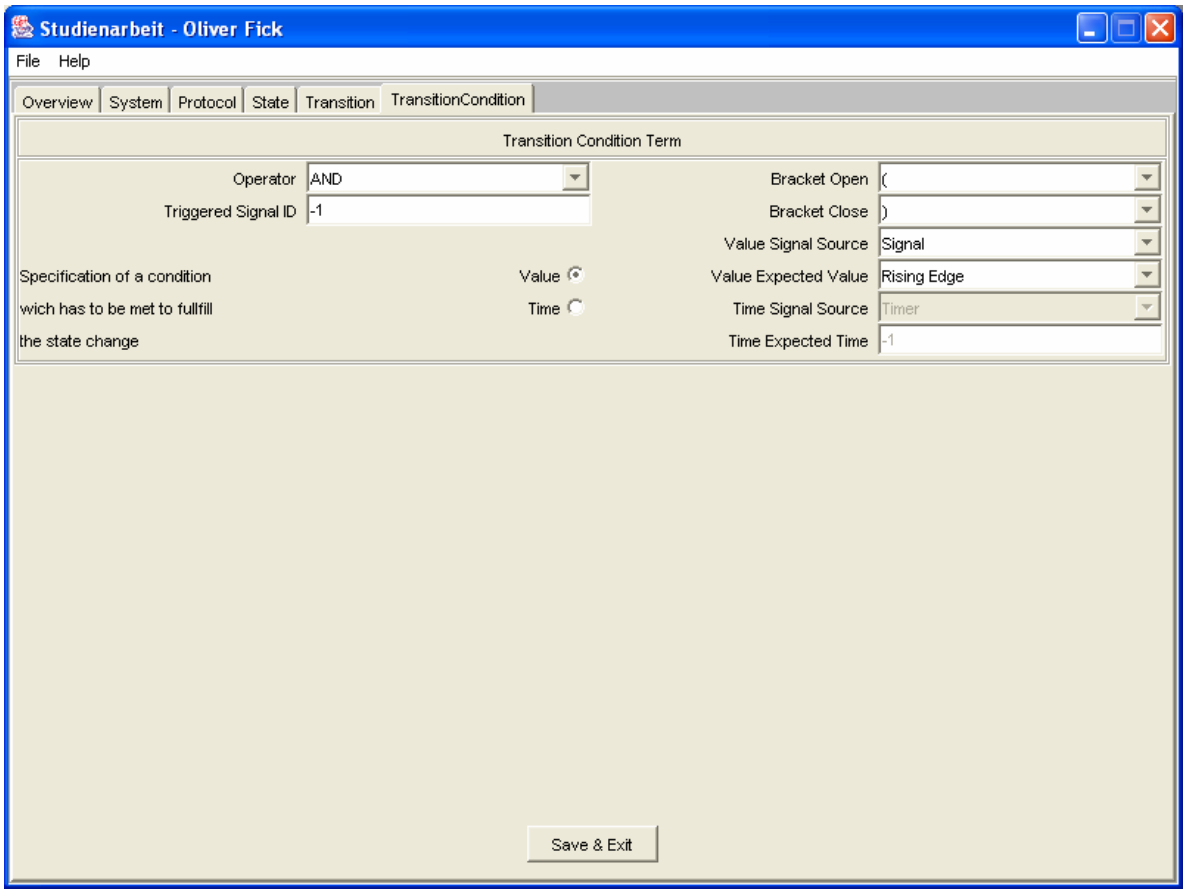

Abbildung 5-16: Der *TransitionCondition-View*

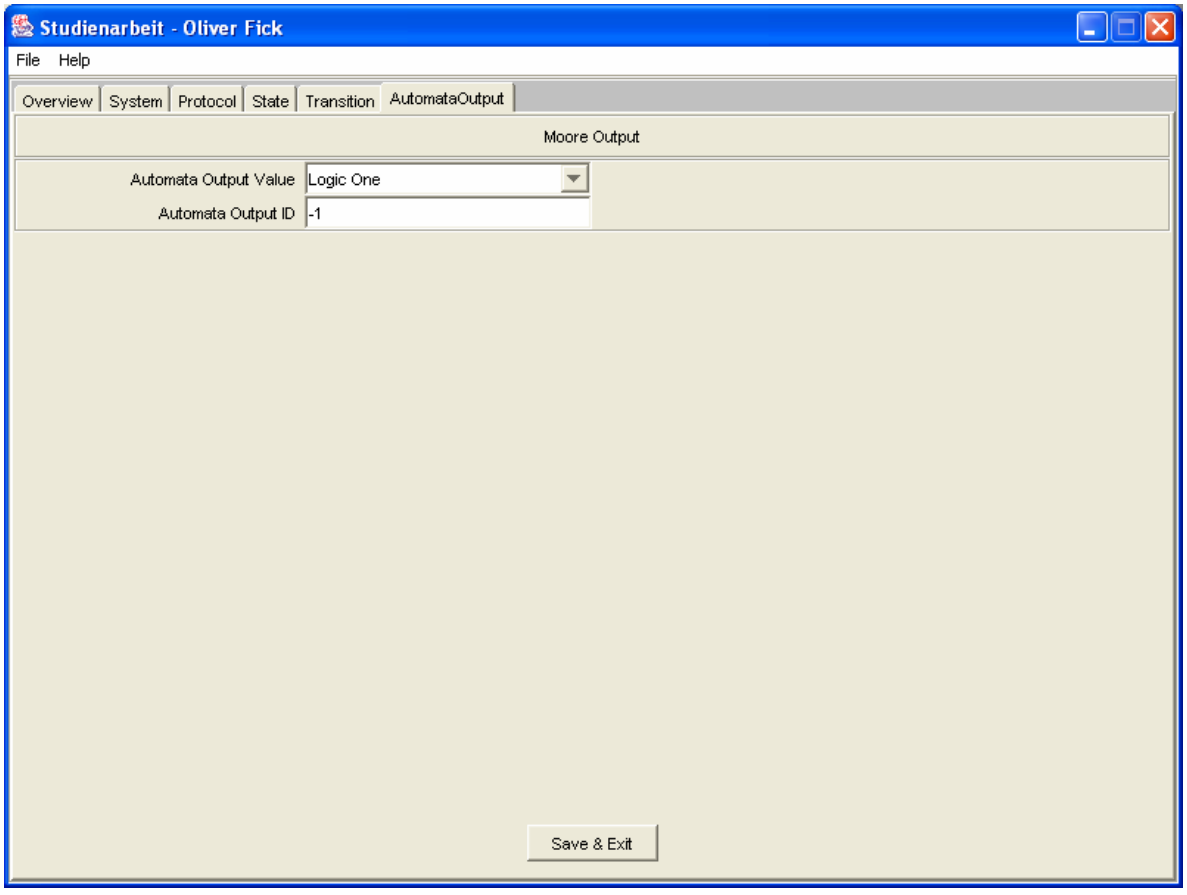

Abbildung 5-17: Der *AutomataOutput-View*

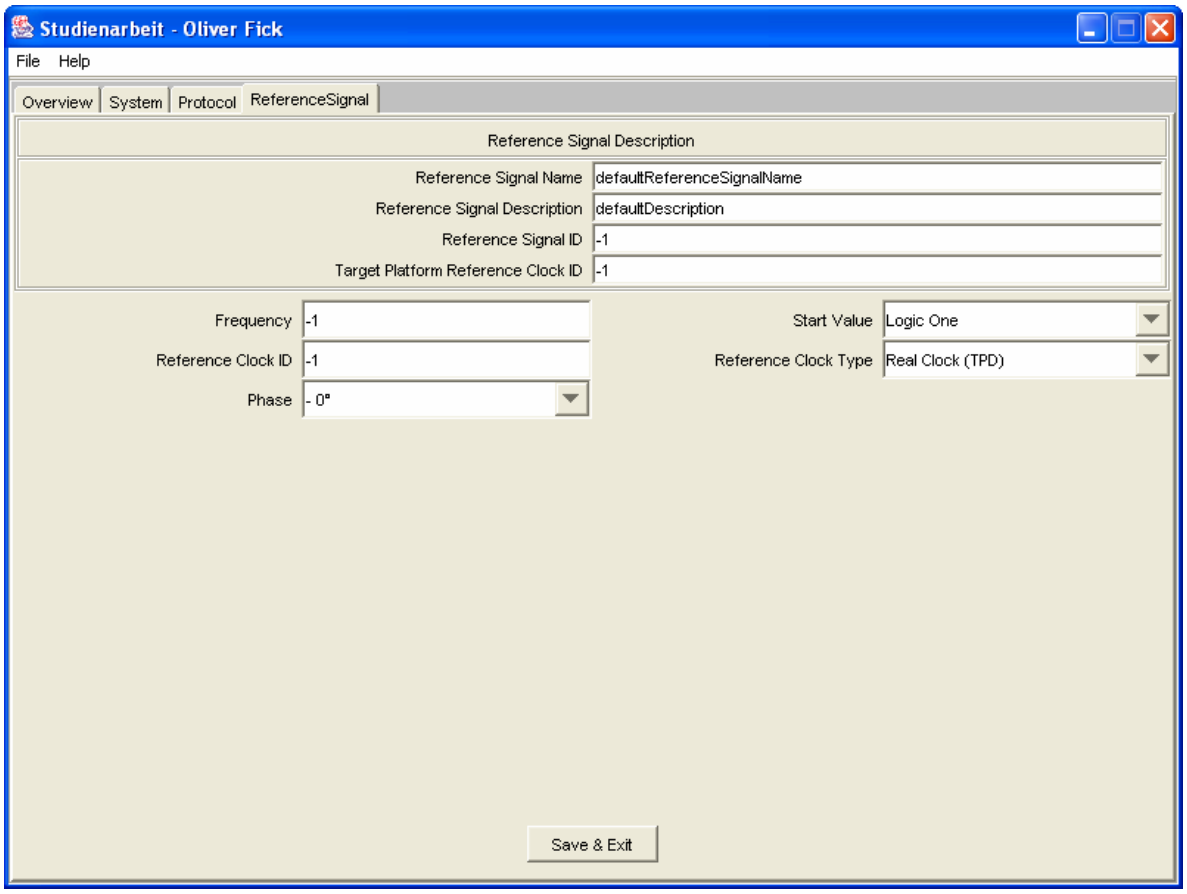

Abbildung 5-18: Der *ReferenceSignal-View* (Clock)

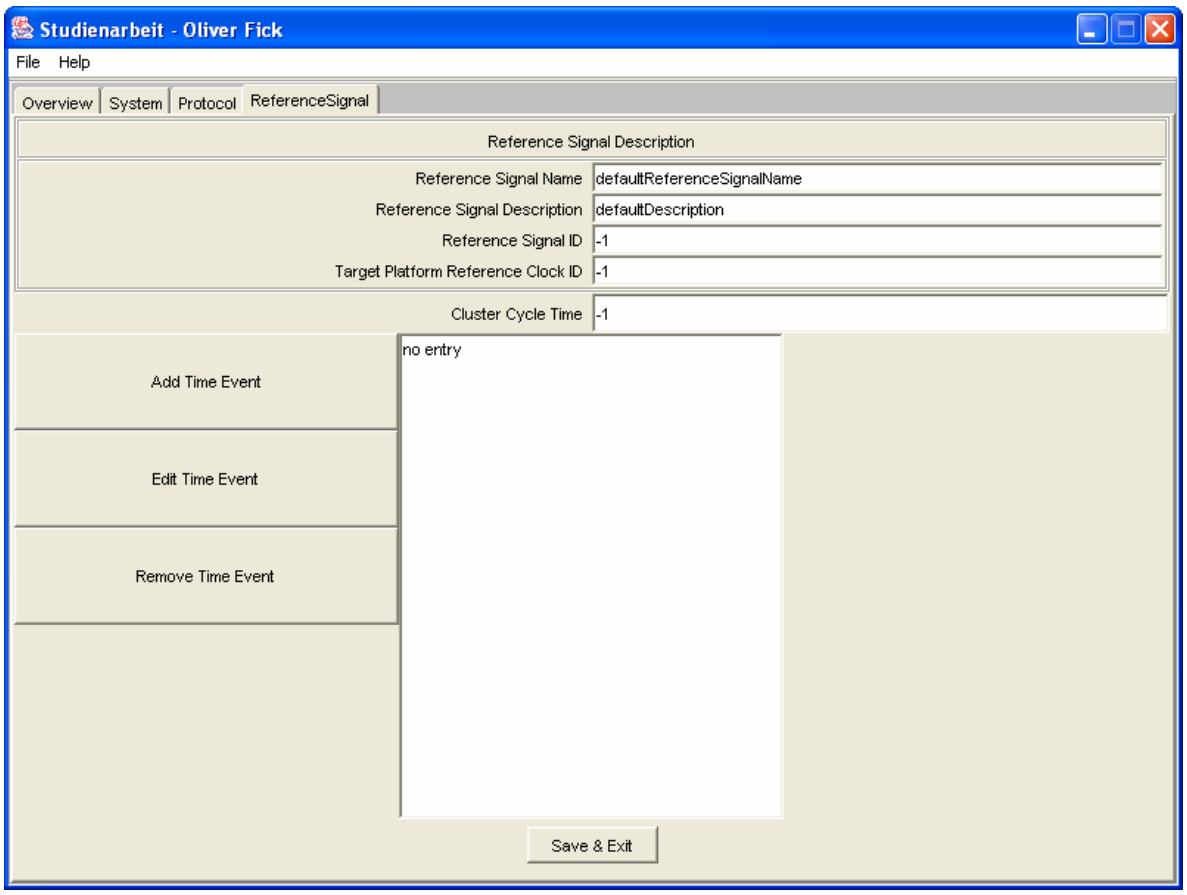

Abbildung 5-19: Der *ReferenceSignal-View* (GlobalDate)

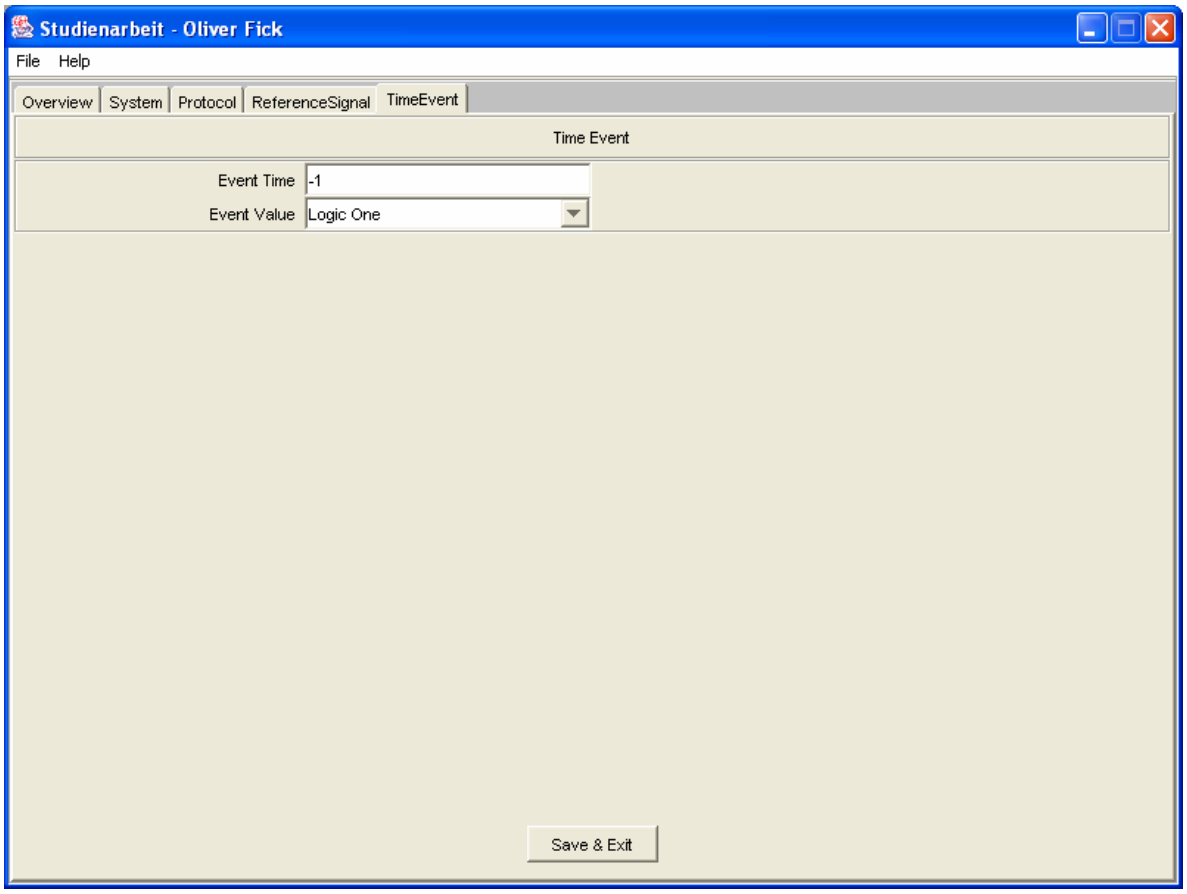

Abbildung 5-20: Der *TimeEvent-View*

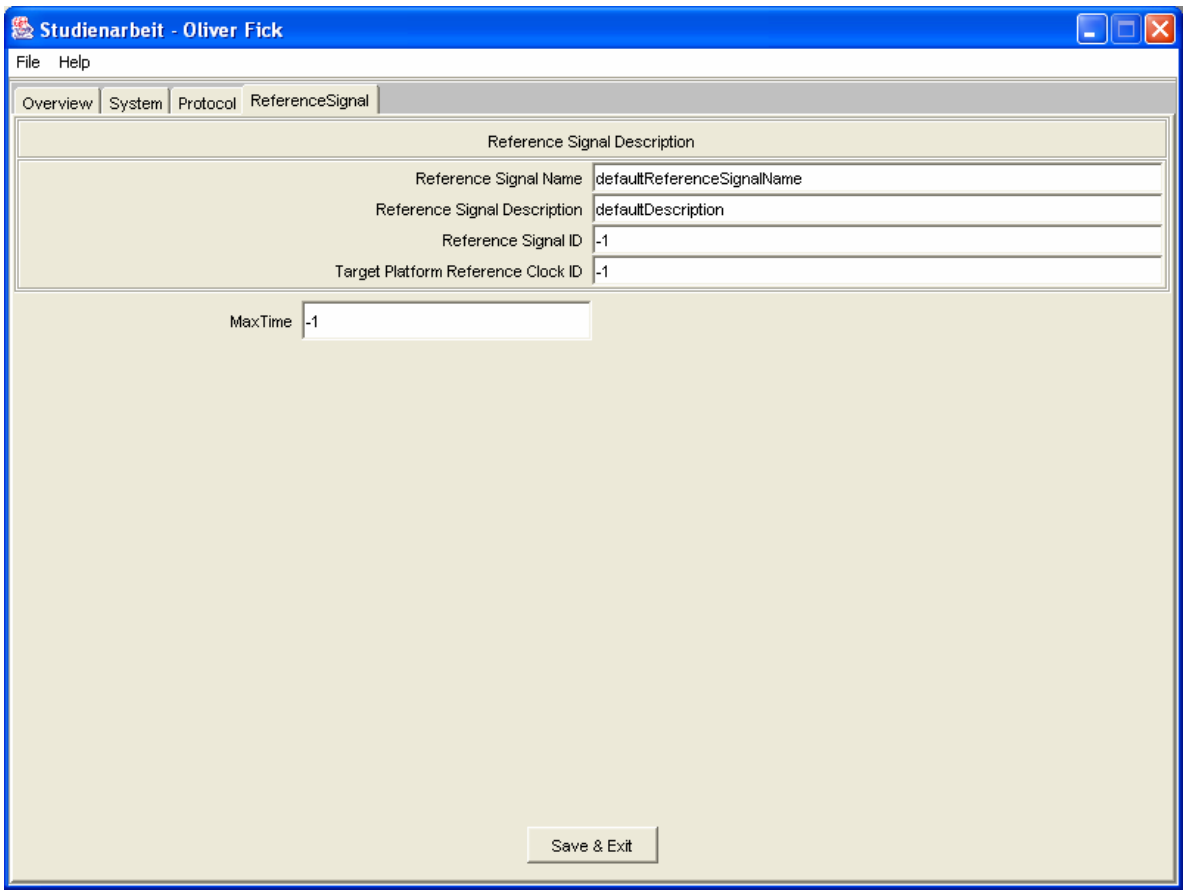

Abbildung 5-21: Der *ReferenceSignal-View* (TimerOrDeadline)

5.2.5. Verdrahtung

Die Verdrahtung zwischen mehreren Schnittstellen wird durch den *InterfaceMap-View* erstellt. Im oberen Bereich kann einer Verdrahtung (*InterfaceMap*) mit Namen, ID und Beschreibung versehen werden. Darunter werden die beiden zu verbindenden Schnittstellen ausgewählt. Mithilfe des Buttons „Take settings“ wird diese Auswahl bestätigt und die graphische Anzeige der Verdrahtung zwischen den beiden Schnittstellen im unteren Bereich aktualisiert.

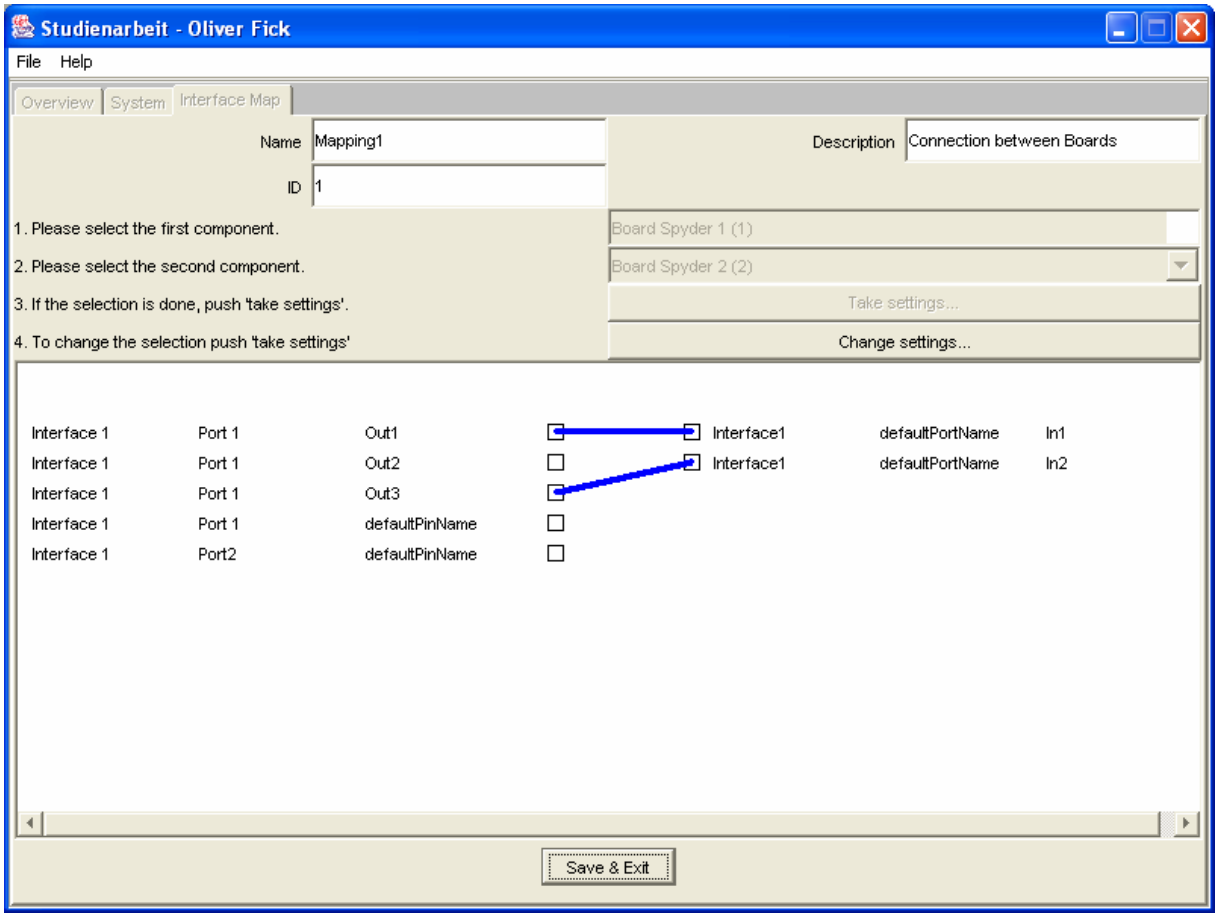

Abbildung 5-22: Der *InterfaceMap-View*

Die Verdrahtung kann mithilfe der Maus schnell erstellt werden. Wenn zwischen zwei Pins mit gerückter Maustaste eine Linie gezogen wird, werden automatisch die Eintträge zur Verdrahtung aktualisiert. Abbildung 5-22 zeigt beispelsweise die Verdrahtung der Pins Out1 und Out3 mit den Pins In1 bzw. In2.

6. Zusammenfassung und Ausblick

6.1. Das IFS Format

Das IFS Format bietet eine Möglichkeit, Chipdesigner bei ihrer Arbeit zu unterstützen. Eine Beschreibung der Schnittstellen integriert in die Modellierung der Systemarchitektur ermöglicht die automatisierte Generierung von Schnittstellenmodulen (Interface Blöcken). Diese Generierung ist allerdings Zukunftsarbeit, die innerhalb des Forschungsprojektes Interface Synthesis an der Universität Paderborn vorangetrieben wird. Weiterhin stellt das Format durch die Nutzung der XML Technologie eine Anbindung zum „IP-based Design“ her.

Die Integration des IFD Formates in eine komplexe Systemarchitektur soll die Verwendungsmöglichkeiten des IFS Formates ausbauen und gegebenenfalls zu Verbesserungen der Beschreibung führen. Eine sich anschließende Evaluierung der IFS Parameter wird zeigen, in wie weit sich diese zur Beschreibung von Schnittstellen eignen.

6.2. Der IFS Editor

Durch den IFS Editor wurde eine Möglichkeit geschaffen, die Menge an Parametern, die durch das IFS Format definiert wurden, zu bewältigen. Der IFS Editor hilft einem Entwickler komplexe Systeme aufzubauen und zu verwalten.

Die Funktionalität des Editors ist in Zukunft zu erweitern. Die Verknüpfung des IFS Projektes mit dem IPQ Projekt führte zur Nutzung gemeinsamer Datenstrukturen und APIs. Diese befinden sich teilweise noch im Entwicklungs- bzw. Weiterentwicklungsstadium. Dies bietet in Zukunft die Möglichkeit weitere Funktionalitäten in den IFS Editor zu integieren.

Folgende zentrale Funktionalitäten werden als nächstes implementiert:

- das Laden und Speichern erstellter Instanzen,
- die Überprüfung der Konsistenz der Daten.

7. Literaturangaben

[1] XML Schema Part 0: Primer
http://www.w3.org/TR/2001/REC-xmlschema-0-20010502/

[2] XML Schema Part 1: Structures
http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/

[3] XML Schema Part 2: Datatypes
http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

[4] Das Web automatisieren mit XML
http://members.aol.com/xmldoku/

[5] Marcel Tiemeyer, Seminarvortrag "Datentypen und Constraints in XML-Schemata" aus dem Seminar "Mobile Computing" von Prof. Dr. Stefan Böttcher, Paderborn, Deutschland, 2003

[6] JavaTM 2 Platform, Standard Edition, v 1.4.1 API Specification
http://java.sun.com/j2se/1.4.1/docs/api/index.html

[8] W. Hardt, M. Visarius, S. Ihmor, Rapid Prototyping of Real-Time Interfaces, FPL – Field Programmable Logic conference in Belfast, 2001

[9] S. Ihmor, Entwurf von Echtzeitschnittstellen am Beispiel interagierender Roboter, Master Thesis, University of Paderborn, 2001

[10] S. Ihmor, M. Visarius, W. Hardt, A Design Methodology for Application-specific Real-Time Interfaces, Proceedings of 2002 IEEE International Conference on Computer Design: VLSI in Computers & Processors, Colorado, USA, May 2002

[11] S. Ihmor, M. Visarius, W. Hardt, A Consistent Design Methodology for Configurable HW/SW Interfaces in Embedded Systems, Int. IFIP WG 10.3 / WG 10.5 Workshop on Distributed and Parallel Embedded Systems (DIPES'02), Montreal, Canada, August 2002

[12] Frank Oppenheimer, Dongming Zhang Wolfgang Nebel, *Modeling Communication Interfaces with ComiX*, Oldenburg, Germany

8. Anhang

8.1. Das IFS Format

8.1.1. Datentypen in XML Notation

ipq:M..16

```
<xs:simpleType name="M..16">
  <xs:annotation>
    <xs:documentation>Multiple 16</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:maxLength value="16"/>
  </xs:restriction>
</xs:simpleType>
```

ipq:M..32

```
<xs:simpleType name="M..32">
  <xs:annotation>
    <xs:documentation>Multiple 32</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:maxLength value="32"/>
  </xs:restriction>
</xs:simpleType>
```

ipq:M..256

```
<xs:simpleType name="M..256">
  <xs:annotation>
    <xs:documentation>Multiple 256</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:maxLength value="256"/>
  </xs:restriction>
</xs:simpleType>
```

ipq:NR1..8

```
<xs:simpleType name="NR1..8">
  <xs:annotation>
    <xs:documentation>Numeric 8</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:unsignedInt">
    <xs:minInclusive value="0"/>
    <xs:totalDigits value="8"/>
  </xs:restriction>
</xs:simpleType>
```

ipq:NR1..16

```
<xs:simpleType name="NR1..16">
  <xs:annotation>
    <xs:documentation>Numeric 16</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:unsignedInt">
    <xs:minInclusive value="0"/>
    <xs:totalDigits value="16"/>
  </xs:restriction>
</xs:simpleType>
```

ipq:NR2S..3.3

```
<xs:simpleType name="NR2S..3.3">
  <xs:annotation>
    <xs:documentation>Numeric Real</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:decimal">
    <xs:totalDigits value="6"/>
    <xs:fractionDigits value="3"/>
  </xs:restriction>
</xs:simpleType>
```

ipq:MinTypMaxNR2S..3.3

```
<xs:complexType name="MinTypMaxNR2S..3.3Type">
  <xs:complexContent>
    <xs:restriction base="ipq:MinTypMaxType">
      <xs:sequence>
        <xs:element name="Minimum" minOccurs="0">
          <xs:annotation>
            <xs:documentation>The minimum value of the
            superior attribute of the VC.</xs:documentation>
          </xs:annotation>
          <xs:complexType>
            <xs:simpleContent>
              <xs:extension base="ipq:NR2S..3.3">
                <xs:attribute name="weight" type="xs:float"
                              use="optional"/>
              </xs:extension>
            </xs:simpleContent>
          </xs:complexType>
        </xs:element>
        <xs:element name="Typical" minOccurs="0">
          <xs:annotation>
            <xs:documentation>The typical value of the
            superior attribute of the VC.</xs:documentation>
          </xs:annotation>
          <xs:complexType>
            <xs:simpleContent>
              <xs:extension base="ipq:NR2S..3.3">
                <xs:attribute name="weight" type="xs:float"
                              use="optional"/>
              </xs:extension>
            </xs:simpleContent>
          </xs:complexType>
        </xs:element>
        <xs:element name="Maximum" minOccurs="0">
          <xs:annotation>
            <xs:documentation>The maximum value of the
            superior attribute of the VC.</xs:documentation>
          </xs:annotation>
          <xs:complexType>
            <xs:simpleContent>
              <xs:extension base="ipq:NR2S..3.3">
                <xs:attribute name="weight" type="xs:float"
                              use="optional"/>
              </xs:extension>
            </xs:simpleContent>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>
```

8.1.2. Systemarchitektur

8.1.2.1. Version.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
  <xs:schema targetNamespace=http://www.upb.de/cs/ipl/ipq
             xmlns:xs=http://www.w3.org/2001/XMLSchema
             xmlns:ipq=http://www.upb.de/cs/ipl/ipq
             elementFormDefault="qualified"
             attributeFormDefault="unqualified">
  <xs:import namespace=http://www.upb.de/cs/ipl/ipq
             schemaLocation="./ISO6093Datatypes.xsd"/>
  <xs:complexType name="VersionType">
    <xs:annotation>
      <xs:documentation>
        General specification of designer specific data and version
        management
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="Design" type="ipq:M..256"
                  default="defaultDesignName">
        <xs:annotation>
          <xs:documentation>
            Name of this design part
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="Description" type="ipq:M..256"
                  default="defaultDescription">
        <xs:annotation>
          <xs:documentation>
            An informal user description of this design part
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="Designer" type="ipq:M..256"
                  default="defaultDesignerName">
        <xs:annotation>
          <xs:documentation>
            Name of the Designer
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="Company" type="ipq:M..256"
                  default="defaultCompanyName">
        <xs:annotation>
          <xs:documentation>
            Name of the Company
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="Release" type="ipq:NR1..8" default="0">
        <xs:annotation>
          <xs:documentation>
            The Release Number
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="Version" type="ipq:NR1..8" default="0">
        <xs:annotation>
```

```
            <xs:documentation>
               The Version Number
            </xs:documentation>
         </xs:annotation>
      </xs:element>
   </xs:sequence>
  </xs:complexType>
</xs:schema>
```

8.1.2.2. System.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace=http://www.upb.de/cs/ipl/ipq
          xmlns:xs=http://www.w3.org/2001/XMLSchema
          xmlns:ipq=http://www.upb.de/cs/ipl/ipq
          elementFormDefault="qualified"
          attributeFormDefault="unqualified">
  <xs:import namespace=http://www.upb.de/cs/ipl/ipq
             schemaLocation="../IPQ/Interface/Protocol.xsd"/>
  <xs:import namespace=http://www.upb.de/cs/ipl/ipq
             schemaLocation="../Maps/InterfaceMap.xsd"/>
  <xs:import namespace=http://www.upb.de/cs/ipl/ipq
             schemaLocation="../Architecture/Medium.xsd"/>
  <xs:import namespace=http://www.upb.de/cs/ipl/ipq
             schemaLocation="../Architecture/Board.xsd"/>
  <xs:import namespace=http://www.upb.de/cs/ipl/ipq
             schemaLocation="../IPQ/Common/Version.xsd"/>
  <xs:element name="System">
    <xs:annotation>
      <xs:documentation>
        The System contains everything needed for Interface Synthesis
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Identification">
          <xs:annotation>
            <xs:documentation>
              Parameters for identification of this System
            </xs:documentation>
          </xs:annotation>
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Name" type="ipq:M..256"
                          default="defaultSystemName">
                <xs:annotation>
                  <xs:documentation>
                    Name of this System
                  </xs:documentation>
                </xs:annotation>
              </xs:element>
              <xs:element name="ID" type="ipq:NR1..16" default="-1">
                <xs:annotation>
                  <xs:documentation>
                    Unique ID of this System
                  </xs:documentation>
                </xs:annotation>
              </xs:element>
              <xs:element name="Description" type="ipq:M..256"
                          default="defaultDesciption">
                <xs:annotation>
                  <xs:documentation>
```

```
                    An informal user description of this System
                  </xs:documentation>
                </xs:annotation>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="Version" type="ipq:VersionType">
          <xs:annotation>
            <xs:documentation>
              Version of this System
            </xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element name="InterfaceMapList" type="ipq:InterfaceMapType">
          <xs:annotation>
            <xs:documentation>
              Collection of InterfaceMaps
            </xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element name="BoardList" type="ipq:BoardType">
          <xs:annotation>
            <xs:documentation>
              Collection of Boards
            </xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element name="ProtocolList" type="ipq:ProtocolType">
          <xs:annotation>
            <xs:documentation>
              Colection of Protocols
            </xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element name="MediumList" type="ipq:MediumType">
          <xs:annotation>
            <xs:documentation>
              Collection of Mediums
            </xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

8.1.2.3. Board.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace=http://www.upb.de/cs/ipl/ipq
           xmlns:xs=http://www.w3.org/2001/XMLSchema
           xmlns:ipq=http://www.upb.de/cs/ipl/ipq
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
  <xs:import namespace=http://www.upb.de/cs/ipl/ipq
             schemaLocation="../IPQ/Interface/Interface.xsd"/>
  <xs:import namespace=http://www.upb.de/cs/ipl/ipq
             schemaLocation="../Maps/InterfaceMap.xsd"/>
  <xs:import namespace=http://www.upb.de/cs/ipl/ipq
             schemaLocation="../Architecture/Medium.xsd"/>
  <xs:import namespace=http://www.upb.de/cs/ipl/ipq
```

```
          schemaLocation="../Architecture/Chip.xsd"/>
<xs:import namespace=http://www.upb.de/cs/ipl/ipq
          schemaLocation="../IPQ/Common/Version.xsd"/>
<xs:complexType name="BoardType">
   <xs:annotation>
      <xs:documentation>
         Defines the componentes on Board level, especially Chips
      </xs:documentation>
   </xs:annotation>
   <xs:sequence>
      <xs:element name="Board" minOccurs="0" maxOccurs="unbounded">
         <xs:annotation>
            <xs:documentation>
               A Board is a part of the System
            </xs:documentation>
         </xs:annotation>
         <xs:complexType>
            <xs:sequence>
               <xs:element name="Identification">
                  <xs:annotation>
                     <xs:documentation>
                        Parameters for identification of this Board
                     </xs:documentation>
                  </xs:annotation>
                  <xs:complexType>
                     <xs:sequence>
                        <xs:element name="Name" type="ipq:M..256"
                                    default="defaultBoardName">
                           <xs:annotation>
                              <xs:documentation>
                                 Name of this Board
                              </xs:documentation>
                           </xs:annotation>
                        </xs:element>
                        <xs:element name="ID" type="ipq:NR1..16"
                                    default="-1">
                           <xs:annotation>
                              <xs:documentation>
                                 Unique ID of this Board
                              </xs:documentation>
                           </xs:annotation>
                        </xs:element>
                        <xs:element name="Description" type="ipq:M..256"
                                    default="defaultDescription">
                           <xs:annotation>
                              <xs:documentation>
                                 An informal user description of this Board
                              </xs:documentation>
                           </xs:annotation>
                        </xs:element>
                        <xs:element name="Category"
                                    default="Spyder Virtex XCV300">
                           <xs:annotation>
                              <xs:documentation>
                                 Category of this Board
                              </xs:documentation>
                           </xs:annotation>
                           <xs:simpleType>
                              <xs:restriction base="ipq:M..256">
                                 <xs:enumeration value="Altera UP1"/>
                                 <xs:enumeration value=
                                                "Spyder Virtex XCV300"/>
                              </xs:restriction>
```

```
                        </xs:simpleType>
                      </xs:element>
                    </xs:sequence>
                  </xs:complexType>
                </xs:element>
                <xs:element name="Version" type="ipq:VersionType">
                  <xs:annotation>
                    <xs:documentation>
                      Version of this Board
                    </xs:documentation>
                  </xs:annotation>
                </xs:element>
                <xs:element name="InterfaceList" type="ipq:InterfaceType">
                  <xs:annotation>
                    <xs:documentation>
                      Collection of Interfaces
                    </xs:documentation>
                  </xs:annotation>
                </xs:element>
                <xs:element name="InterfaceMapList"
                            type="ipq:InterfaceMapType">
                  <xs:annotation>
                    <xs:documentation>
                      Collection of InterfaceMaps
                    </xs:documentation>
                  </xs:annotation>
                </xs:element>
                <xs:element name="ChipList" type="ipq:ChipType">
                  <xs:annotation>
                    <xs:documentation>
                      Collection of Chips
                    </xs:documentation>
                  </xs:annotation>
                </xs:element>
                <xs:element name="MediumList" type="ipq:MediumType">
                  <xs:annotation>
                    <xs:documentation>
                      Collection of Mediums
                    </xs:documentation>
                  </xs:annotation>
                </xs:element>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:schema>
```

8.1.3. Interface Description

8.1.3.1. Interface.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace=http://www.upb.de/cs/ipl/ipq
           xmlns:ipq=http://www.upb.de/cs/ipl/ipq
           xmlns:xs=http://www.w3.org/2001/XMLSchema
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
  <xs:import namespace=http://www.upb.de/cs/ipl/ipq
           schemaLocation="../Common/ISO6093Datatypes.xsd"/>
  <xs:import namespace=http://www.upb.de/cs/ipl/ipq
```

```
        schemaLocation="./Properties.xsd"/>
<xs:import namespace=http://www.upb.de/cs/ipl/ipq
        schemaLocation="./Port.xsd"/>
<xs:import namespace=http://www.upb.de/cs/ipl/ipq
        schemaLocation="./Register.xsd"/>
<xs:complexType name="InterfaceType">
   <xs:annotation>
      <xs:documentation>
         General specification of I/O structure for components
      </xs:documentation>
   </xs:annotation>
   <xs:sequence>
      <xs:element name="Interface" minOccurs="0" maxOccurs="unbounded">
         <xs:annotation>
            <xs:documentation>
               An Interface is a part of Board, Chip, Task, Medium or an
               IFB
            </xs:documentation>
         </xs:annotation>
         <xs:complexType>
            <xs:sequence>
               <xs:element name="Identification">
                  <xs:annotation>
                     <xs:documentation>
                        Parameters for identification of this Interface
                     </xs:documentation>
                  </xs:annotation>
                  <xs:complexType>
                     <xs:sequence>
                        <xs:element name="Name" type="ipq:M..256"
                                    default="defaultInterfaceName">
                           <xs:annotation>
                              <xs:documentation>
                                 The name of this Interface
                              </xs:documentation>
                           </xs:annotation>
                        </xs:element>
                        <xs:element name="ID" type="ipq:NR1..16"
                                    default="-1">
                           <xs:annotation>
                              <xs:documentation>
                                 Unique ID of this Interface
                              </xs:documentation>
                           </xs:annotation>
                        </xs:element>
                        <xs:element name="Description" type="ipq:M..256"
                                    default="defaultDescription">
                           <xs:annotation>
                              <xs:documentation>
                                 An informal user description of this
                                 Interface
                              </xs:documentation>
                           </xs:annotation>
                        </xs:element>
                        <xs:element name="Connector" minOccurs="0">
                           <xs:annotation>
                              <xs:documentation>
                                 Connector specification, only used for Medium
                                 connection when there is a physical connector
                                 available
                              </xs:documentation>
                           </xs:annotation>
                           <xs:complexType>
```

```
                <xs:sequence>
                  <xs:element name="Category"
                       default="PIO (Extensionheader)">
                    <xs:annotation>
                      <xs:documentation>
                        Category of the connector
                      </xs:documentation>
                    </xs:annotation>
                    <xs:simpleType>
                      <xs:restriction base="ipq:M..32">
                        <xs:enumeration value=
                                    "PIO (Extensionheader)"/>
                        <xs:enumeration value="USB_A"/>
                        <xs:enumeration value="USB_B"/>
                        <xs:enumeration value=
                                    "IEEE 1394 (Firewire)"/>
                        <xs:enumeration value=
                                    "Wannenstecker"/>
                        <xs:enumeration value="Sub-D"/>
                        <xs:enumeration value="Undefined"/>
                        <xs:enumeration value="Centronics"/>
                        <xs:enumeration value="BNC"/>
                        <xs:enumeration value="RJ 45 (LAN)"/>
                        <xs:enumeration value=
                                    "RJ 11 (Phone)"/>
                        <xs:enumeration value="N-Connector"/>
                        <xs:enumeration value="DIN"/>
                        <xs:enumeration value="SCA"/>
                        <xs:enumeration value="PS2"/>
                        <xs:enumeration value=
                                    "DIN 41612 (IEC603-2)"/>
                        <xs:enumeration value="DIN 41617"/>
                        <xs:enumeration value="DIN 41622"/>
                        <xs:enumeration value="SCART"/>
                        <xs:enumeration value="VG64"/>
                        <xs:enumeration value="KEL100"/>
                      </xs:restriction>
                    </xs:simpleType>
                  </xs:element>
                  <xs:element name="Gender" default="MALE">
                    <xs:annotation>
                      <xs:documentation>
                        Gender of the connector
                      </xs:documentation>
                    </xs:annotation>
                    <xs:simpleType>
                      <xs:restriction base="ipq:M..16">
                        <xs:enumeration value="MALE"/>
                        <xs:enumeration value="FEMALE"/>
                      </xs:restriction>
                    </xs:simpleType>
                  </xs:element>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="Properties" type="ipq:PropertiesType">
        <xs:annotation>
          <xs:documentation>
            Properties of this Interface
          </xs:documentation>
```

```
            </xs:annotation>
          </xs:element>
          <xs:element name="PortList" type="ipq:PortListType">
            <xs:annotation>
              <xs:documentation>
                Collection of Ports
              </xs:documentation>
            </xs:annotation>
          </xs:element>
          <xs:element name="RegisterList" type="ipq:RegisterListType">
            <xs:annotation>
              <xs:documentation>
                Collection of Registers
              </xs:documentation>
            </xs:annotation>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
 </xs:complexType>
</xs:schema>
```

8.1.3.2. Properties.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace=http://www.upb.de/cs/ipl/ipq
          xmlns:ipq=http://www.upb.de/cs/ipl/ipq
          xmlns:xs=http://www.w3.org/2001/XMLSchema
          elementFormDefault="qualified"
          attributeFormDefault="unqualified">
  <xs:import namespace=http://www.upb.de/cs/ipl/ipq
          schemaLocation="../Common/ISO6093Datatypes.xsd"/>
  <xs:import namespace=http://www.upb.de/cs/ipl/ipq
          schemaLocation="../Common/Datatypes.xsd"/>
  <xs:complexType name="PropertiesType">
    <xs:annotation>
      <xs:documentation>
        General specification of common properties for all components
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="MaxFrequency" default="-1">
        <xs:annotation>
          <xs:documentation>
            This attribute specifies the maximum frequency.
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="ipq:NR1..16">
              <xs:attribute name="unit" type="xs:string"use="optional"
                            fixed="Hz"/>
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
      <xs:element name="MaxVoltage" default="-1">
        <xs:annotation>
          <xs:documentation>
            This attribute specifies the maximum voltage level
          </xs:documentation>
```

```
    </xs:annotation>
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="ipq:NR2S..3.3">
          <xs:attribute name="unit" type="xs:string" use="optional"
                        fixed="V"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="MaxCurrent" default="-1">
    <xs:annotation>
      <xs:documentation>
        This attribute specifies the maximum current level
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="ipq:NR2S..3.3">
          <xs:attribute name="unit" type="xs:string"
                        use="optional" fixed="A"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="IOTechnologyType" default="other">
    <xs:annotation>
      <xs:documentation>
        This attribute specifies the technology which realizes the
        I/O.
      </xs:documentation>
    </xs:annotation>
    <xs:simpleType>
      <xs:restriction base="ipq:M..32">
        <xs:enumeration value="DTL"/>
        <xs:enumeration value="TTL"/>
        <xs:enumeration value="PCI"/>
        <xs:enumeration value="LVDS"/>
        <xs:enumeration value="Differential"/>
        <xs:enumeration value="singleEnded"/>
        <xs:enumeration value="ECL"/>
        <xs:enumeration value="PECL"/>
        <xs:enumeration value="openCollector"/>
        <xs:enumeration value="other"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="MaxWireLength" default="-1" minOccurs="0">
    <xs:annotation>
      <xs:documentation>
        This attribute specifies the maximum length of connected
        cable or wires.
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="ipq:NR2S..3.3">
          <xs:attribute name="unit" type="xs:string"
                        use="optional" fixed="m"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
```

```
        <xs:element name="Latency" minOccurs="0">
          <xs:annotation>
            <xs:documentation>
              This attribute indicates the degree of latency on the actual
              level of abstraction.
            </xs:documentation>
          </xs:annotation>
          <xs:complexType>
            <xs:complexContent>
              <xs:extension base="ipq:MinTypMaxNR2S..3.3Type">
                <xs:attribute name="unit" type="xs:string"
                              use="optional" fixed="s"/>
              </xs:extension>
            </xs:complexContent>
          </xs:complexType>
        </xs:element>
        <xs:element name="Jitter" minOccurs="0">
          <xs:annotation>
            <xs:documentation>
              This attribute indicates the degree of jitter on the actual
              level of abstraction.
            </xs:documentation>
          </xs:annotation>
          <xs:complexType>
            <xs:complexContent>
              <xs:extension base="ipq:MinTypMaxNR2S..3.3Type">
                <xs:attribute name="unit" type="xs:string"
                              use="optional" fixed="s"/>
              </xs:extension>
            </xs:complexContent>
          </xs:complexType>
        </xs:element>
        <xs:element name="Sensitivity" minOccurs="0">
          <xs:annotation>
            <xs:documentation>
              This attribute indicates the degree of tolerance to
              interference such as noise.
            </xs:documentation>
          </xs:annotation>
          <xs:complexType>
            <xs:complexContent>
              <xs:extension base="ipq:MinTypMaxNR2S..3.3Type">
                <xs:attribute name="unit" type="xs:string"
                              use="optional" fixed="dB"/>
              </xs:extension>
            </xs:complexContent>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
</xs:schema>
```

8.1.3.3. Port.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace=http://www.upb.de/cs/ipl/ipq
           xmlns:ipq=http://www.upb.de/cs/ipl/ipq
           xmlns:xs=http://www.w3.org/2001/XMLSchema
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
  <xs:import namespace=http://www.upb.de/cs/ipl/ipq
             schemaLocation="../Common/ISO6093Datatypes.xsd"/>
```

```
<xs:import namespace=http://www.upb.de/cs/ipl/ipq
          schemaLocation="./Pin.xsd"/>
<xs:complexType name="PortListType">
   <xs:annotation>
      <xs:documentation>
         General specification of I/O structure at Port level
      </xs:documentation>
   </xs:annotation>
   <xs:sequence>
      <xs:element name="Port" minOccurs="0" maxOccurs="unbounded">
         <xs:annotation>
            <xs:documentation>
               A Port is a group of dependend Pins
            </xs:documentation>
         </xs:annotation>
         <xs:complexType>
            <xs:sequence>
               <xs:element name="Identification">
                  <xs:annotation>
                     <xs:documentation>
                        Parameters for identification of this Port
                     </xs:documentation>
                  </xs:annotation>
                  <xs:complexType>
                     <xs:sequence>
                        <xs:element name="Name" type="ipq:M..256"
                                    default="defaultPortName">
                           <xs:annotation>
                              <xs:documentation>
                                 The name of this Port
                              </xs:documentation>
                           </xs:annotation>
                        </xs:element>
                        <xs:element name="ID" type="ipq:NR1..16"
                                    default="-1">
                           <xs:annotation>
                              <xs:documentation>
                                 Unique ID of this Port
                              </xs:documentation>
                           </xs:annotation>
                        </xs:element>
                        <xs:element name="Description" type="ipq:M..256"
                                    default="defaultDescription">
                           <xs:annotation>
                              <xs:documentation>
                                 An informal user description of this Port
                              </xs:documentation>
                           </xs:annotation>
                        </xs:element>
                     </xs:sequence>
                  </xs:complexType>
               </xs:element>
               <xs:element name="Characterization">
                  <xs:annotation>
                     <xs:documentation>
                        Characterization of central properties of this Port
                     </xs:documentation>
                  </xs:annotation>
                  <xs:complexType>
                     <xs:sequence>
                        <xs:element name="VLogicOne" default="3.3">
                           <xs:annotation>
                              <xs:documentation>
```

```
                        Defines voltage level for a logical one
                        (high) for this Port
                     </xs:documentation>
                  </xs:annotation>
                  <xs:complexType>
                     <xs:simpleContent>
                        <xs:extension base="ipq:NR2S..3.3">
                           <xs:attribute name="unit" type="xs:string"
                                         use="optional" fixed="V"/>
                        </xs:extension>
                     </xs:simpleContent>
                  </xs:complexType>
               </xs:element>
               <xs:element name="VLogicZero" default="0.0">
                  <xs:annotation>
                     <xs:documentation>
                        Defines voltage level for a logical zero
                         (low) for this Port
                     </xs:documentation>
                  </xs:annotation>
                  <xs:complexType>
                     <xs:simpleContent>
                        <xs:extension base="ipq:NR2S..3.3">
                           <xs:attribute name="unit" type="xs:string"
                                          use="optional" fixed="V"/>
                        </xs:extension>
                     </xs:simpleContent>
                  </xs:complexType>
               </xs:element>
            </xs:sequence>
         </xs:complexType>
      </xs:element>
      <xs:element name="PinList" type="ipq:PinListType">
         <xs:annotation>
            <xs:documentation>
               Collection of Pins
            </xs:documentation>
         </xs:annotation>
      </xs:element>
   </xs:sequence>
</xs:complexType>
</xs:schema>
```

8.1.3.4. Pin.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace=http://www.upb.de/cs/ipl/ipq
           xmlns:xs=http://www.w3.org/2001/XMLSchema
           xmlns:ipq="http://www.upb.de/cs/ipl/ipq"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
   <xs:import namespace="http://www.upb.de/cs/ipl/ipq"
              schemaLocation="../Common/ISO6093Datatypes.xsd"/>
   <xs:complexType name="PinListType">
      <xs:annotation>
         <xs:documentation>
            General specification of I/O at Pin level
         </xs:documentation>
      </xs:annotation>
```

```
<xs:sequence>
   <xs:element name="Pin" minOccurs="0" maxOccurs="unbounded">
      <xs:annotation>
         <xs:documentation>
            A Pin is a member of a Port and specifies the properties of
            a single Pin
         </xs:documentation>
      </xs:annotation>
      <xs:complexType>
         <xs:sequence>
            <xs:element name="Identification">
               <xs:annotation>
                  <xs:documentation>
                     Parameters for Identification of this Pin
                  </xs:documentation>
               </xs:annotation>
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="Name" type="ipq:M..256"
                                 default="defaultPinName">
                        <xs:annotation>
                           <xs:documentation>
                              User name of this Pin
                           </xs:documentation>
                        </xs:annotation>
                     </xs:element>
                     <xs:element name="ID" type="ipq:NR1..16"
                                 default="-1">
                        <xs:annotation>
                           <xs:documentation>
                              Unique ID of this Pin
                           </xs:documentation>
                        </xs:annotation>
                     </xs:element>
                     <xs:element name="UserID" type="ipq:M..256"
                                 default="-1" minOccurs="0">
                        <xs:annotation>
                           <xs:documentation>
                              Alternative user ID for this Pin
                           </xs:documentation>
                        </xs:annotation>
                     </xs:element>
                     <xs:element name="Description" type="ipq:M..256"
                                 default="defaultDescription">
                        <xs:annotation>
                           <xs:documentation>
                              An informal user description of this Pin
                           </xs:documentation>
                        </xs:annotation>
                     </xs:element>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
            <xs:element name="Characterization">
               <xs:annotation>
                  <xs:documentation>
                     Characterization of central properties of this Pin
                  </xs:documentation>
               </xs:annotation>
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="Direction" default="BiDir">
                        <xs:annotation>
```

```
                        <xs:documentation>
                            Specifies signal direction of this Pin
                            </xs:documentation>
                        </xs:annotation>
                        <xs:simpleType>
                            <xs:restriction base="ipq:M..32">
                                <xs:enumeration value="Input"/>
                                <xs:enumeration value="Output"/>
                                <xs:enumeration value="Bidirectional"/>
                                <xs:enumeration value="Special (GND, VCC)"/>
                                <xs:enumeration value="Undefined"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:element>
                    <xs:element name="Type" default="Std_Logic">
                        <xs:annotation>
                            <xs:documentation>
                                Specifies possible signal states and values
                                of this Pin
                            </xs:documentation>
                        </xs:annotation>
                        <xs:simpleType>
                            <xs:restriction base="ipq:M..32">
                                <xs:enumeration value="Bit"/>
                                <xs:enumeration value="Std_Logic"/>
                                <xs:enumeration value="VCC"/>
                                <xs:enumeration value="Ground"/>
                                <xs:enumeration value="NC (Not Connected)"/>
                                <xs:enumeration value="Undefined"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:element>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

8.1.4. Protokolle

8.1.4.1. Protocol.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://www.upb.de/cs/ipl/ipq"
           xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:ipq="http://www.upb.de/cs/ipl/ipq"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
  <xs:import namespace="http://www.upb.de/cs/ipl/ipq"
           schemaLocation="./State.xsd"/>
  <xs:import namespace="http://www.upb.de/cs/ipl/ipq"
           schemaLocation="./ReferenceSignal.xsd"/>
  <xs:import namespace="http://www.upb.de/cs/ipl/ipq"
           schemaLocation="./ProtocolPin.xsd"/>
  <xs:import namespace="http://www.upb.de/cs/ipl/ipq"
           schemaLocation="./TransactionType.xsd"/>
  <xs:import namespace="http://www.upb.de/cs/ipl/ipq"
```

```
          schemaLocation="../Common/ISO6093Datatypes.xsd"/>
<xs:import namespace="http://www.upb.de/cs/ipl/ipq"
          schemaLocation="../Common/Version.xsd"/>
<xs:complexType name="ProtocolType">
   <xs:annotation>
      <xs:documentation>
         A behavior based, general description for various Protocols
      </xs:documentation>
   </xs:annotation>
   <xs:sequence>
      <xs:element name="Protocol" minOccurs="0"
                  maxOccurs="unbounded">
         <xs:annotation>
            <xs:documentation>
               Specification of one Protocol that can be mapped to an
               intercafe of the stuctural description
            </xs:documentation>
         </xs:annotation>
         <xs:complexType>
            <xs:sequence>
               <xs:element name="Identification">
                  <xs:annotation>
                     <xs:documentation>
                        Parameters for identification of this Protocol
                     </xs:documentation>
                  </xs:annotation>
                  <xs:complexType>
                     <xs:sequence>
                        <xs:element name="Name" type="ipq:M..256"
                                    default="defaultProtocolName">
                           <xs:annotation>
                              <xs:documentation>
                                 User Name of this Protocol
                              </xs:documentation>
                           </xs:annotation>
                        </xs:element>
                        <xs:element name="ID" type="ipq:NR1..16"
                                    default="-1">
                           <xs:annotation>
                              <xs:documentation>
                                 Unique ID of this Protocol
                              </xs:documentation>
                           </xs:annotation>
                        </xs:element>
                        <xs:element name="Description" type="ipq:M..256"
                                    default="defaultDescription">
                           <xs:annotation>
                              <xs:documentation>
                                 An informal user description of this Protocol
                              </xs:documentation>
                           </xs:annotation>
                        </xs:element>
                        <xs:element name="Category" default="Other">
                           <xs:annotation>
                              <xs:documentation>
                                 Category of this Protocol
                              </xs:documentation>
                           </xs:annotation>
                           <xs:simpleType>
                              <xs:restriction base="ipq:M..256">
                                 <xs:enumeration value="RS232"/>
                                 <xs:enumeration value="RS485"/>
                                 <xs:enumeration value="LVDS"/>
```

```
                  <xs:enumeration value="Firewire"/>
                  <xs:enumeration value="UserDefined"/>
                  <xs:enumeration value="Other"/>
               </xs:restriction>
            </xs:simpleType>
         </xs:element>
      </xs:sequence>
   </xs:complexType>
</xs:element>
<xs:element name="Version" type="ipq:VersionType">
   <xs:annotation>
      <xs:documentation>
         Version of this Protocol
      </xs:documentation>
   </xs:annotation>
</xs:element>
<xs:element name="Characterization">
   <xs:annotation>
      <xs:documentation>
         Parameters for identification of this Protocol
      </xs:documentation>
   </xs:annotation>
   <xs:complexType>
      <xs:sequence>
         <xs:element name="Control" default="Other">
            <xs:annotation>
               <xs:documentation>
                  Specifies who starts transaction Process
               </xs:documentation>
            </xs:annotation>
            <xs:simpleType>
               <xs:restriction base="ipq:M..32">
                  <xs:enumeration value="Responder"/>
                  <xs:enumeration value="Initiator"/>
                  <xs:enumeration value="Other"/>
               </xs:restriction>
            </xs:simpleType>
         </xs:element>
         <xs:element name="Data" default="Other">
            <xs:annotation>
               <xs:documentation>
                  Defines if protocol entity generates or
                  accepts data
               </xs:documentation>
            </xs:annotation>
            <xs:simpleType>
               <xs:restriction base="ipq:M..32">
                  <xs:enumeration value="Consumer"/>
                  <xs:enumeration value="Producer"/>
                  <xs:enumeration value="Other"/>
               </xs:restriction>
            </xs:simpleType>
         </xs:element>
         <xs:element name="UseCase" default="Other">
            <xs:annotation>
               <xs:documentation>
                  Category of this Protocol for special use
                  cases
               </xs:documentation>
            </xs:annotation>
            <xs:simpleType>
               <xs:restriction base="ipq:M..32">
                  <xs:enumeration value="Arbitration"/>
```

```
                  <xs:enumeration value="Control"/>
                  <xs:enumeration value="Data"/>
                  <xs:enumeration value="Interrupt"/>
                  <xs:enumeration value="Other"/>
               </xs:restriction>
            </xs:simpleType>
         </xs:element>
         <xs:element name="Flow" default="Other">
            <xs:annotation>
               <xs:documentation>
                  Implementation of the data flow model for
                  this protocol
               </xs:documentation>
            </xs:annotation>
            <xs:simpleType>
               <xs:restriction base="ipq:M..32">
                  <xs:enumeration value="Persistent"/>
                  <xs:enumeration value="Buffered"/>
                  <xs:enumeration value="FIFO"/>
                  <xs:enumeration value="LIFO"/>
                  <xs:enumeration value="Blocking"/>
                  <xs:enumeration value=
                                    "AssignedPortPriority"/>
                  <xs:enumeration value=
                                    "AssignedDataPriority"/>
                  <xs:enumeration value="MultiRate"/>
                  <xs:enumeration value="Pipelined"/>
                  <xs:enumeration value=
                                    "ExceptionsHandled"/>
                  <xs:enumeration value="Other"/>
               </xs:restriction>
            </xs:simpleType>
         </xs:element>
         <xs:element name="TransactionTypeList"
                     type="ipq:TransactionTypeListType">
            <xs:annotation>
               <xs:documentation>
                  Implemented functions which are supported by
                  the protocol entity
               </xs:documentation>
            </xs:annotation>
         </xs:element>
      </xs:sequence>
   </xs:complexType>
</xs:element>
<xs:element name="ProtocolPinList"
            type="ipq:ProtocolPinListType">
   <xs:annotation>
      <xs:documentation>
         Collection of Protocol Pin
      </xs:documentation>
   </xs:annotation>
</xs:element>
<xs:element name="StateList" type="ipq:StateType">
   <xs:annotation>
      <xs:documentation>
         Collection of States
      </xs:documentation>
   </xs:annotation>
</xs:element>
<xs:element name="ReferenceSignalList"
               type="ipq:ReferenceSignalType">
   <xs:annotation>
```

```
                    <xs:documentation>
                       Collection of Reference Signals
                    </xs:documentation>
                 </xs:annotation>
              </xs:element>
           </xs:sequence>
        </xs:complexType>
     </xs:element>
  </xs:sequence>
 </xs:complexType>
</xs:schema>
```

8.1.4.2. TransitionCondition.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 U (http://www.xmlspy.com) by Der Eine (-) -->
<xs:schema targetNamespace="http://www.upb.de/cs/ipl/ipq"
           xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:ipq="http://www.upb.de/cs/ipl/ipq"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
  <xs:import namespace="http://www.upb.de/cs/ipl/ipq"
             schemaLocation="../Common/ISO6093Datatypes.xsd"/>
  <xs:complexType name="TransitionConditionType">
     <xs:annotation>
        <xs:documentation>
           General specification of all protocols for components
        </xs:documentation>
     </xs:annotation>
     <xs:sequence>
        <xs:element name="TransitionCondition" maxOccurs="unbounded">
           <xs:annotation>
              <xs:documentation>
                 Defines condition rules for state changes
              </xs:documentation>
           </xs:annotation>
           <xs:complexType>
              <xs:sequence>
                 <xs:element name="Operator" default="AND" minOccurs="0">
                    <xs:annotation>
                       <xs:documentation>
                          Binary Operator to concatenate two signals
                       </xs:documentation>
                    </xs:annotation>
                    <xs:simpleType>
                       <xs:restriction base="ipq:M..16">
                          <xs:enumeration value="AND"/>
                          <xs:enumeration value="OR"/>
                          <xs:enumeration value="XOR"/>
                          <xs:enumeration value="NAND"/>
                          <xs:enumeration value="NOR"/>
                          <xs:enumeration value="XNOR"/>
                       </xs:restriction>
                    </xs:simpleType>
                 </xs:element>
                 <xs:element name="BracketOpen" default="(" minOccurs="0">
                    <xs:annotation>
                       <xs:documentation>
                          Opening Bracket to describe complex conditions
                       </xs:documentation>
                    </xs:annotation>
                    <xs:simpleType>
```

```
      <xs:restriction base="ipq:M..16">
        <xs:enumeration value="("/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="TriggeredSignalID" type="ipq:NR1..16"
              default="-1">
    <xs:annotation>
      <xs:documentation>
        Id of triggered Signal (ProtocolPin or
        ReferenceSignal)
      </xs:documentation>
    </xs:annotation>
  </xs:element>
  <xs:element name="Trigger">
    <xs:annotation>
      <xs:documentation>
        Specification of a condition which has to be met to
        fulfill the state change
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:choice>
        <xs:element name="Value">
          <xs:annotation>
            <xs:documentation>
              Signal is triggered by a Signal value
              constraint
              </xs:documentation>
          </xs:annotation>
          <xs:complexType>
            <xs:sequence>
              <xs:element name="ValueSignalSource"
                          default="Signal">
                <xs:annotation>
                  <xs:documentation>
                    Specifies if Signal is a Clock or a
                    on-Clock Signal
                  </xs:documentation>
                </xs:annotation>
                <xs:simpleType>
                  <xs:restriction base="ipq:M..16">
                    <xs:enumeration value="Clock"/>
                    <xs:enumeration value="Signal"/>
                  </xs:restriction>
                </xs:simpleType>
              </xs:element>
              <xs:element name="ValueExpectedValue"
                          default="Rising Edge">
                <xs:annotation>
                  <xs:documentation>
                    Sepcifies Value which has to be Met
                  </xs:documentation>
                </xs:annotation>
                <xs:simpleType>
                  <xs:restriction base="ipq:M..256">
                    <xs:enumeration value="Low Value"/>
                    <xs:enumeration value="High Value"/>
                    <xs:enumeration value="Falling Edge"/>
                    <xs:enumeration value="Rising Edge"/>
                    <xs:enumeration value="Signal
                                           Changed"/>
                  </xs:restriction>
```

```
                    </xs:simpleType>
                  </xs:element>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
            <xs:element name="Time">
              <xs:annotation>
                <xs:documentation>
                  Signal is triggered by a Timing value
                  Constraint
                </xs:documentation>
              </xs:annotation>
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="TimeSignalSource"
                              default="Timer">
                    <xs:annotation>
                      <xs:documentation>
                        Specifies if Signal is a Timer, Global
                        Date or Deadline
                      </xs:documentation>
                    </xs:annotation>
                    <xs:simpleType>
                      <xs:restriction base="ipq:M..16">
                        <xs:enumeration value="Timer"/>
                        <xs:enumeration value="Global_Time"/>
                        <xs:enumeration value="Deadline"/>
                      </xs:restriction>
                    </xs:simpleType>
                  </xs:element>
                  <xs:element name="TimeExpectedValue"
                              default="-1">
                    <xs:annotation>
                      <xs:documentation>
                        Specifies Time constraint which has to
                        be met
                      </xs:documentation>
                    </xs:annotation>
                    <xs:complexType>
                      <xs:simpleContent>
                        <xs:extension base="ipq:NR2S..3.3">
                          <xs:attribute name="unit"
                                        type="xs:string"
                                        use="optional"
                                        fixed="s"/>
                        </xs:extension>
                      </xs:simpleContent>
                    </xs:complexType>
                  </xs:element>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:choice>
        </xs:complexType>
      </xs:element>
      <xs:element name="BracketClose" default=")" minOccurs="0">
        <xs:annotation>
          <xs:documentation>
            Closing Bracket to describe complex conditions
          </xs:documentation>
        </xs:annotation>
        <xs:simpleType>
          <xs:restriction base="ipq:M..16">
```

```
                    <xs:enumeration value=")"/>
                  </xs:restriction>
                </xs:simpleType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
</xs:schema>
```

8.1.4.3. ReferenceSignal.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://www.upb.de/cs/ipl/ipq"
           xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:ipq="http://www.upb.de/cs/ipl/ipq"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
  <xs:import namespace="http://www.upb.de/cs/ipl/ipq"
             schemaLocation="../Common/ISO6093Datatypes.xsd"/>
  <xs:complexType name="ReferenceSignalType">
    <xs:annotation>
      <xs:documentation>
        General specification of all protocols for components
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="ReferenceSignal" minOccurs="0"
                  maxOccurs="unbounded">
        <xs:annotation>
          <xs:documentation>
            A ReferenceSignal is a member of a Protocol and specifies
            the properties of a additinal Reference Signal
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="Identification">
              <xs:annotation>
                <xs:documentation>
                  Parameters for Identification of this ReferenceSignal
                </xs:documentation>
              </xs:annotation>
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="Name" type="ipq:M..256"
                              default="defaultReferenceSignalName">
                    <xs:annotation>
                      <xs:documentation>
                        User name of this ReferenceSignal
                      </xs:documentation>
                    </xs:annotation>
                  </xs:element>
                  <xs:element name="ID" type="ipq:NR1..16"
                              default="-1">
                    <xs:annotation>
                      <xs:documentation>
                        Unique ID of this ReferenceSignal
                      </xs:documentation>
                    </xs:annotation>
                  </xs:element>
```

```
            <xs:element name="Description" type="ipq:M..256"
                        default="defaultDescription">
               <xs:annotation>
                  <xs:documentation>
                     An informal user description of this
                     ReferenceSignal
                  </xs:documentation>
               </xs:annotation>
            </xs:element>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:element name="TimeReference">
      <xs:annotation>
         <xs:documentation>
            Selection of timing behaviour
         </xs:documentation>
      </xs:annotation>
      <xs:complexType>
         <xs:choice>
            <xs:element name="Clock">
               <xs:annotation>
                  <xs:documentation>
                     Clock as Reference Signal
                  </xs:documentation>
               </xs:annotation>
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="Frequency" default="-1">
                        <xs:annotation>
                           <xs:documentation>
                              Frequency of the specified Clock
                           </xs:documentation>
                        </xs:annotation>
                        <xs:complexType>
                           <xs:simpleContent>
                              <xs:extension base="ipq:NR2S..3.3">
                                 <xs:attribute name="unit"
                                    type="xs:string" use="optional"
                                    fixed="Hz"/>
                              </xs:extension>
                           </xs:simpleContent>
                        </xs:complexType>
                     </xs:element>
                     <xs:element name="StartValue"
                                 default="Logic One">
                        <xs:annotation>
                           <xs:documentation>
                              Signal StartValue of the specified
                              Clock
                           </xs:documentation>
                        </xs:annotation>
                        <xs:simpleType>
                           <xs:restriction base="ipq:M..16">
                              <xs:enumeration value="Logic One"/>
                              <xs:enumeration value="Logic Zero"/>
                           </xs:restriction>
                        </xs:simpleType>
                     </xs:element>
                     <xs:element name="Phaseshift">
                        <xs:annotation>
                           <xs:documentation>
                              Specification of clock shifts towards
```

```
                other clocks.
              </xs:documentation>
            </xs:annotation>
            <xs:complexType>
              <xs:sequence>
                <xs:element name="ReferenceClockID"
                     type="ipq:NR1..16" default="-1">
                  <xs:annotation>
                    <xs:documentation>
                      Specifies to which clock ID
                      this Clock will be referenced
                    </xs:documentation>
                  </xs:annotation>
                </xs:element>
                <xs:element name="ReferenceClockType"
                             default="Real">
                  <xs:annotation>
                    <xs:documentation>
                      Specifies if the referenced
                      Clock is a Real or another
                      Logical Signal
                    </xs:documentation>
                  </xs:annotation>
                  <xs:simpleType>
                    <xs:restriction base="ipq:M..32">
                      <xs:enumeration value=
                              "Real Clock (TPD)"/>
                      <xs:enumeration value=
                              "Reference Clock"/>
                    </xs:restriction>
                  </xs:simpleType>
                </xs:element>
                <xs:element name="Phase" default="0°">
                  <xs:annotation>
                    <xs:documentation>
                      Phaseshift of actual clock
                      compared to referenced clock
                    </xs:documentation>
                  </xs:annotation>
                  <xs:simpleType>
                    <xs:restriction base=
                                  "ipq:M..256">
                      <xs:enumeration value="- 0°"/>
                      <xs:enumeration value=
                              "-  45°  ( Pi / 4 )"/>
                      <xs:enumeration value=
                              "-  90°  ( Pi / 2 )"/>
                      <xs:enumeration value=
                           "-135°  ( 3 Pi / 4 )"/>
                      <xs:enumeration value=
                              "-180°  ( Pi )"/>
                    </xs:restriction>
                  </xs:simpleType>
                </xs:element>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="TimerOrDeadline">
      <xs:annotation>
        <xs:documentation>
```

```
            Timer or Deadline as Reference Signal
         </xs:documentation>
      </xs:annotation>
      <xs:complexType>
         <xs:sequence>
            <xs:element name="MaxTime" default="-1">
               <xs:annotation>
                  <xs:documentation>
                     Specifies the amount of Time when the
                     Timer or the Deadline will exceed
                  </xs:documentation>
               </xs:annotation>
               <xs:complexType>
                  <xs:simpleContent>
                     <xs:extension base="ipq:NR2S..3.3">
                        <xs:attribute name="unit"
                           type="xs:string" use="optional"
                           fixed="s"/>
                     </xs:extension>
                  </xs:simpleContent>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:element name="GlobalDate">
      <xs:annotation>
         <xs:documentation>
            Global Date as Reference Signal
         </xs:documentation>
      </xs:annotation>
      <xs:complexType>
         <xs:sequence>
            <xs:element name="CycleTime" default="-1">
               <xs:annotation>
                  <xs:documentation>
                     Time of one complete Hyper Periode
                     (also cluster cycle)
                  </xs:documentation>
               </xs:annotation>
               <xs:complexType>
                  <xs:simpleContent>
                     <xs:extension base="ipq:NR2S..3.3">
                        <xs:attribute name="unit"
                           type="xs:string" use="optional"
                           fixed="s"/>
                     </xs:extension>
                  </xs:simpleContent>
               </xs:complexType>
            </xs:element>
            <xs:element name="TimeEventList"
                        default="-1">
               <xs:annotation>
                  <xs:documentation>
                     List of TimeEvents within one Hyper
                     Periode (also called cluster cycle)
                  </xs:documentation>
               </xs:annotation>
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="TimeEvent"
                                 minOccurs="0"
                                 maxOccurs="unbounded">
```

```
                                    <xs:annotation>
                                       <xs:documentation>
                                          Single TimeEvent within the
                                          HyperPeriode (also called
                                          cluster cycle)
                                       </xs:documentation>
                                    </xs:annotation>
                                    <xs:complexType>
                                       <xs:sequence>
                                          <xs:element name="EventTime"
                                                      default="-1">
                                             <xs:annotation>
                                                <xs:documentation>
                                                   Timepoints within the
                                                   ClusterCycle
                                                </xs:documentation>
                                             </xs:annotation>
                                             <xs:complexType>
                                                <xs:simpleContent>
                                                   <xs:extension base=
                                                         "ipq:NR2S..3.3">
                                                      <xs:attribute
                                                         name="unit"
                                                         type="xs:string"
                                                         use="optional"
                                                         fixed="s"/>
                                                   </xs:extension>
                                                </xs:simpleContent>
                                             </xs:complexType>
                                          </xs:element>
                                          <xs:element name="EventValue"
                                                      default="Logic One">
                                             <xs:annotation>
                                                <xs:documentation>
                                                   ActionValue to happen
                                                   at the given TimeEvent
                                                </xs:documentation>
                                             </xs:annotation>
                                             <xs:simpleType>
                                                <xs:restriction base=
                                                      "ipq:M..16">
                                                   <xs:enumeration value=
                                                         "Logic One"/>
                                                   <xs:enumeration value=
                                                         "Logic Zero"/>
                                                   <xs:enumeration value=
                                                         "High Peak"/>
                                                   <xs:enumeration value=
                                                         "Low Peak"/>
                                                </xs:restriction>
                                             </xs:simpleType>
                                          </xs:element>
                                       </xs:sequence>
                                    </xs:complexType>
                                 </xs:element>
                              </xs:sequence>
                           </xs:complexType>
                        </xs:element>
                     </xs:sequence>
                  </xs:complexType>
               </xs:element>
            </xs:choice>
         </xs:complexType>
```

```
            </xs:element>
            <xs:element name="TargetPlatformRefClockID"
                        type="ipq:NR1..16" default="-1">
              <xs:annotation>
                <xs:documentation>
                  Target Platform Reference Clock ID for selection of
                  clock source
                </xs:documentation>
              </xs:annotation>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

8.1.5. Verdrahtung / Maps

8.1.5.1. InterfaceMap.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://www.upb.de/cs/ipl/ipq"
          xmlns:xs="http://www.w3.org/2001/XMLSchema"
          xmlns:ipq="http://www.upb.de/cs/ipl/ipq"
          elementFormDefault="qualified"
          attributeFormDefault="unqualified">
  <xs:import namespace="http://www.upb.de/cs/ipl/ipq"
            schemaLocation="../IPQ/Common/ISO6093Datatypes.xsd"/>
  <xs:import namespace="http://www.upb.de/cs/ipl/ipq"
            schemaLocation="./InterfacePinMap.xsd"/>
  <xs:complexType name="InterfaceMapType">
    <xs:sequence>
      <xs:element name="InterfaceMap" minOccurs="0" maxOccurs="unbounded">
        <xs:annotation>
          <xs:documentation>
            An InterfaceMap is the desciption of the connection between
            two Port
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="Identification">
              <xs:annotation>
                <xs:documentation>
                  Parameters for identification of this InterfaceMap
                </xs:documentation>
              </xs:annotation>
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="Name" type="ipq:M..256"
                              default="defaultInterfaceMapName">
                    <xs:annotation>
                      <xs:documentation>
                        The name of this InterfaceMap
                      </xs:documentation>
                    </xs:annotation>
                  </xs:element>
                  <xs:element name="ID" type="ipq:NR1..16"
                              default="-1">
                    <xs:annotation>
```

```
            <xs:documentation>
              Unique ID of this InterfaceMap
            </xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element name="Description" type="ipq:M..256"
                    default="defaultDescription">
          <xs:annotation>
            <xs:documentation>
              An informal user description of this
              InterfaceMap
            </xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="InterfaceReferenceA">
    <xs:annotation>
      <xs:documentation>
        Parameters to reference to Interface A
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="DeviceCategoryA" default="Board">
          <xs:annotation>
            <xs:documentation>
              Category of device A
            </xs:documentation>
          </xs:annotation>
          <xs:simpleType>
            <xs:restriction base="ipq:M..16">
              <xs:enumeration value="Board"/>
              <xs:enumeration value="Chip"/>
              <xs:enumeration value="Task"/>
              <xs:enumeration value="Medium"/>
              <xs:enumeration value="IFB"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
        <xs:element name="DeviceIDA" type="ipq:NR1..16"
                    default="-1">
          <xs:annotation>
            <xs:documentation>
              ID of device A
            </xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element name="InterfaceIDA" type="ipq:NR1..16"
                    default="-1">
          <xs:annotation>
            <xs:documentation>
              ID of the Interface of device A
            </xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="InterfaceReferenceB">
    <xs:annotation>
      <xs:documentation>
```

```
                        Parameters to reference to Interface B
                    </xs:documentation>
                </xs:annotation>
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="DeviceCategoryB" default="Board">
                            <xs:annotation>
                                <xs:documentation>
                                    Category of device B
                                </xs:documentation>
                            </xs:annotation>
                            <xs:simpleType>
                                <xs:restriction base="ipq:M..16">
                                    <xs:enumeration value="Board"/>
                                    <xs:enumeration value="Chip"/>
                                    <xs:enumeration value="Task"/>
                                    <xs:enumeration value="Medium"/>
                                    <xs:enumeration value="IFB"/>
                                </xs:restriction>
                            </xs:simpleType>
                        </xs:element>
                        <xs:element name="DeviceIDB" type="ipq:NR1..16"
                                    default="-1">
                            <xs:annotation>
                                <xs:documentation>
                                    ID of device B
                                </xs:documentation>
                            </xs:annotation>
                        </xs:element>
                        <xs:element name="InterfaceIDB" type="ipq:NR1..16"
                                    default="-1">
                            <xs:annotation>
                                <xs:documentation>
                                    ID of the Interface of device B
                                </xs:documentation>
                            </xs:annotation>
                        </xs:element>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="InterfacePinMapList"
                        type="ipq:InterfacePinMapListType">
                <xs:annotation>
                    <xs:documentation>
                        Collection of PinMaps
                    </xs:documentation>
                </xs:annotation>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:schema>
```

8.1.5.2. InterfacePinMap.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://www.upb.de/cs/ipl/ipq"
           xmlns:ipq="http://www.upb.de/cs/ipl/ipq"
           xmlns:xs="http://www.w3.org/2001/XMLSchema"
           elementFormDefault="qualified"
```

```
        attributeFormDefault="unqualified">
  <xs:import namespace="http://www.upb.de/cs/ipl/ipq"
             schemaLocation="../IPQ/Common/ISO6093Datatypes.xsd"/>
  <xs:complexType name="InterfacePinMapListType">
    <xs:annotation>
      <xs:documentation>
        General specification of I/O at Pin level
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="InterfacePinMap" minOccurs="0"
                  maxOccurs="unbounded">
        <xs:annotation>
          <xs:documentation>
            A Pin Map is the connection betwee two Pins
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="PortIDA" type="ipq:NR1..16" default="-1">
              <xs:annotation>
                <xs:documentation>
                  ID of the Port of Interface A
                </xs:documentation>
              </xs:annotation>
            </xs:element>
            <xs:element name="PinIDA" type="ipq:NR1..16" default="-1">
              <xs:annotation>
                <xs:documentation>
                  ID of the Pin of Port A
                </xs:documentation>
              </xs:annotation>
            </xs:element>
            <xs:element name="PortIDB" type="ipq:NR1..16" default="-1">
              <xs:annotation>
                <xs:documentation>
                  ID of the Port of Interface B
                </xs:documentation>
              </xs:annotation>
            </xs:element>
            <xs:element name="PinIDB" type="ipq:NR1..16" default="-1">
              <xs:annotation>
                <xs:documentation>
                  ID of the Pin of Port B
                </xs:documentation>
              </xs:annotation>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```