RF communication is a cheap and – assuming protocols such as multi-hop – also one of the most dependable communication methods.
» Integration of highest-grade security features from existing "SmartCard" designs (a.k.a. crypto-cards) to guarantee information authenticity and privacy.

Especially the "System in Package" (SiP) integration is an ambitious goal that would – if done towards a working product – exceed financial resources and time frame of a STREP. However, being a research project, SmartCoDe aims at showing feasibility and analyzing and quantifying the benefit of the approach described. Therefore, the consortium will execute the SiP design at system level and estimate cost, feasibility and solve the respective problems such as packaging.

For demonstration, a prototype of the "SiP" that is built from legacy components and with restricted functionality (available microcontroller, improved RF transceiver, simplified protocol stack) is used. This will allow us to achieve our ambitious scientific and technological goals while remaining feasible within a STREP.

In short, SmartCoDe will …
» allow the local energy sources to dynamically announce the availability of energy via an appropriate channel (e.g. internet, change of power frequency, PCL, …),
» enable the energy management unit to react on this information in a smart and safe way,
» enable the local power network to balance the production/demand matching as far as possible by applying intelligent energy management methods to eliminate peaks in the local energy consumption (or generation) curve,
» set up an interface of the local energy network to the grid for changing variable pricing policies based on availability of energy,
» register and reward the owners and users of "smart controlled" devices, e.g. applying a power price depending on power used in different states of the power profile.

SmartCoDe finished its first Reporting & Review Period successfully on April 30, 2010.

vMAGIC – Automatic Code Generation for VHDL

Eine Bibliothek zur Automatisierung von VHDL-Entwurfsaufgaben

Wir stellen in diesem Artikel die Open Source Bibliothek vMAGIC [1] vor, mit deren Hilfe sich Werkzeuge für verschiedene Aufgaben im Bereich der VHDL-Entwicklung erzeugen lassen. Wie jede freie Software lebt vMAGIC vom Feedback der Nutzer, und wir hoffen auf diesem Wege das Interesse vieler VHDL-Entwickler wecken zu können. Der Fokus des Artikels liegt auf der grundlegenden Funktionalität der Bibliothek, die anhand eines Beispiels erläutert wird.

Einleitung

Produktivität und Verlässlichkeit, Wartbarkeit und Wiederverwendbarkeit sind Selbstverständlichkeiten für jeden nachhaltigen Entwicklungsprozess. Während Softwareentwickler durch vielfältige Werkzeuge, die die Einhaltung dieser Ziele vereinfachen und sicherstellen, Unterstützung erfahren, fehlt eine vergleichbar breite Unterstützung in der Hardwareentwicklung häufig. Insbesondere sind frei verfügbare Werkzeuge nur für bestimmte Nischen, wie beispielsweise FSM- oder ROM-Generatoren zu finden.

Ziel des Java-basierten vMAGIC-Projektes ist es, eine Plattform zu schaffen, die den VHDL-Entwickler im weitesten Sinne unterstützt und den Entwicklungsprozess einfacher und sicherer gestaltet. Dazu implementiert die vMAGIC-Bibliothek Funktionen, die den Zugriff auf VHDL-Code vereinfachen und ermöglicht damit Werkzeuge, wie Code- oder IP-Core-Generatoren, Code-Transformatoren und Code-Analysatoren. Die Grundidee der Bibliothek liegt darin, VHDL-Code „zugreifbar" zu machen, indem alles, was sonst z.B. für einen Codegenerator durch reine Textoperationen abgebildet werden müsste, auf eine formale Ebene gehoben wird. So ist es beispielsweise grundsätzlich möglich, ein Register mit Textoperationen in einen Quellcode einzufügen. Dazu muss ein entsprechender Prozess textuell in den Quellcode hineinkopiert werden, die Zuweisungen von Ein- und Ausgangssignalen müssen eingefügt und die ursprünglichen Verbindungen gelöscht werden. Spätestens aber, wenn der Quellcode dem Entwickler nicht im Vorhinein bekannt ist, wird das praktisch unmöglich. Außerdem müssen zahlreiche Randbedingungen, wie etwa die korrekte

Weitergehende Informationen gibt es auf der Projekt-Webseite http://vmagic.sf.net.

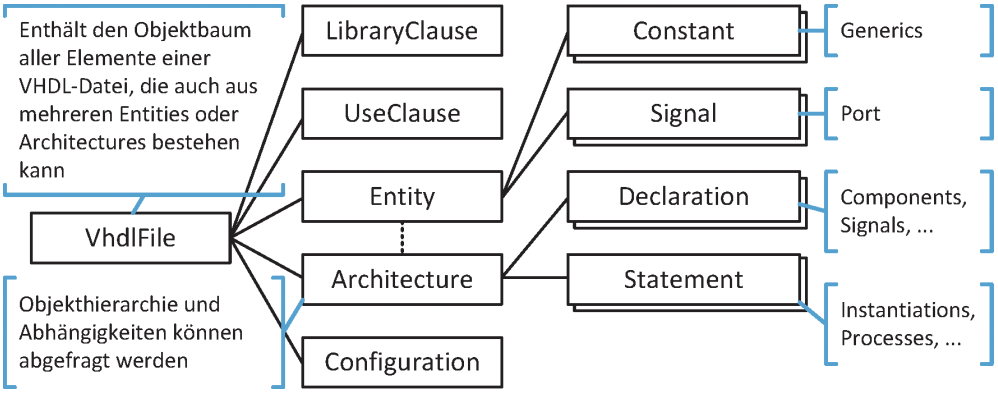

Abbildung 1.02: Die interne Baumstruktur von vMAGIC enthält alle Elemente des VHDL-Codes

etc.). Für das Registerbeispiel bedeutet dies, dass vMAGIC die Architektur findet und das Anwenderprogramm ein Register-Objekt und neue Ausgangssignale einfügt, und über vMAGIC die entsprechenden Signale umbenennt.

Um diese Funktionen zu erläutern gehen wir zunächst auf den Aufbau von vMAGIC ein, um dann den Programmablauf an einem einfachen Beispiel zu demonstrieren.

Aufbau

vMAGIC verwendet sogenannte Metaklassen, um VHDL-Code auf syntaktischer Ebene zu repräsentieren und zu manipulieren. Im Prinzip existieren für alle VHDL-Konstrukte, wie Deklarationen, Zuweisungen, Ausdrücke, Prozesse, etc. jeweils Metaklassen, die vom Parser (oder vom Anwenderprogramm) erzeugt werden. Entsprechend dem objektorientierten Ansatz gibt es in jeder dieser Klassen Funktionen, die den Zugriff auf die Elemente der Klassen ermöglichen. So enthält z. B. die Klasse für Zuweisungen eine Funktion, um auf das Ziel der Zuweisung zuzugreifen. Diese Klassen bzw. Objekte werden in einer Baumstruktur (Abbildung 1.02) abgelegt, womit dann beliebiger VHDL-Code in einer Form vorliegt, die sehr einfach manipuliert werden kann. Damit besteht die Bibliothek im Wesentlichen aus drei Teilen:

» Der VHDL-Parser kann bestehenden VHDL-Code in Meta-Objekte überführen und diese in einer Baumstruktur ablegen. Diese Struktur ähnelt dem Abstract Syntax Tree (AST), der einen typischen Zwischenschritt bei der Verarbeitung von Quellcode darstellt. Das Einlesen einer VHDL-Datei ist optional und vor allem dann wichtig, wenn ein bestehendes Template verarbeitet werden soll, oder Informationen aus einer bestehenden VHDL-Datei als Grundlage für die weitere Verarbeitung dienen sollen. Falls kein bestehender Quelltext verarbeitet werden soll, können alle Elemente einer VHDL-Datei auch ausschließlich über ein Anwenderprogramm erzeugt werden.

» Für jedes VHDL-Objekt definiert vMAGIC Operationen, mit denen die Ausdrücke, Zuweisungen, Prozesse, etc. entsprechend den Anforderungen eines Benutzers manipuliert werden können. Genauso lassen sich auch neue VHDL-Elemente erzeugen und in bestehenden Quellcode einfügen. Syntaktische Fehler sind durch diese Funktionen praktisch ausgeschlossen, alleine semantische Fehler können in der aktuellen Version noch erzeugt werden. Im Rahmen aktueller Arbeiten werden Funktionen erarbeitet, die semantische Fehler, wie z. B. das Nichtdeklarieren

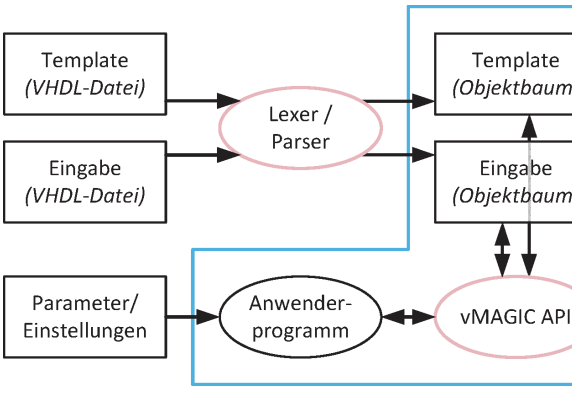

Abbildung 1.03: Entwurfsablauf beim Einsatz von vMAGIC