

Erzeugung einer ausführbaren Spezifikation für den WiMAX-Standard

Im Januar 2006 kündigte das WiMAX Forum die ersten Produkte an, die die strengen Testvorschriften für eine Zertifizierung nach dem IEEE 802.162004-Standard erfüllen. Dieser Standard garantiert die Kompatibilität und Interoperabilität von Breitband-Funkkomponenten. Das fast 900 Seiten lange Dokument beschreibt sowohl die Physikalische Schicht als auch die Media Access Control-Schicht von WiMAX-Systemen.

Der enorme Umfang und die Komplexität dieses Standards stellt für die Entwickler WiMAX-konformer Komponenten eine große Herausforderung dar. Eine Möglichkeit, dem zu begegnen, ist der Einsatz von Model-Based Design mit Simulink®. Im Mittelpunkt des gesamten Entwicklungsprozesses steht dabei ein Systemmodell, das von der Festlegung der Anforderungen bis hin zur fertigen Implementierung und den Systemtests durchgehend als ausführbare Spezifikation und als interaktive Testumgebung dient. Bei einem dokumentenbasierten Entwicklungsansatz sind die Anforderungen in der Produktspezifikation niedergelegt, die nach und nach ausgearbeitet, partitioniert und in Teilspezifikationen für die einzelnen Entwicklungsteams übersetzt wird. Jedes neue Dokument stellt jedoch eine Übersetzung der Anforderungen dar, die Fehler enthalten kann oder in der etwas vergessen wurde. Diese Mängel werden unter Umständen erst während der Konformitätstests entdeckt. Eine ausführbare Spezifikation dagegen ermöglicht es, das System durchgängig zu testen und zu verifizieren, wodurch Übertragungsfehler

leichter aufgedeckt und korrigiert werden können. In diesem Artikel wird eine ausführbare Spezifikation für einen WiMAX-Sender entwickelt. Wir konzentrieren uns dabei auf die Kanalkodierung, die in Abschnitt 8.8.3. des Standards beschrieben wird.

Aufbau und Test des Kanalmodells

Da viele Anforderungen für Kommunikationsstandards mathematischer Natur sind, eignen sich MATLAB® und Simulink besonders gut zum Aufbau einer ausführbaren Spezifikation. Weil aber auch die Erzeugung eines Simulink-Modells eine Übersetzung des WiMAX-Standards darstellt, müssen wir sehr sorgfältig vorgehen.

In Abschnitt 8.8.3. sind praktischerweise Testfälle aufgeführt, die zum schrittweisen Aufbau eines Modells genutzt werden

können. Die Kanalkodierung des WiMAX-Standards besteht aus fünf Schritten: Zufällige Anordnung der Daten (Randomisierung), Forward Error Correction (FEC), Verschachtelung (Interleaving), Modulation und Erzeugung der OFDM-Symbole. Wir beginnen mit einem Gerüst aus leeren Subsystemen (Abb. 1) und bauen den Sender Block für Block auf, wobei wir mit Hilfe der Daten aus dem Standard jeden Block testen, bevor wir weitermachen. Der Eingabevektor für den ersten WiMAX-Testfall ist 35 Bytes lang und in Hex-Notation angegeben. Mit der MATLAB-Funktion `sscanf` lässt er sich leicht als Zeilenvektor darstellen:

```
input_data =
sscanf(['45 29 C4 79 AD 0F 55 28 AD \...
      '87 B5 76 1A 9C 80 50 45 1B \...
      '9F D9 2A 88 95 EB AE B5 2E \...
      '03 4F 09 14 69 58 0A 5D'], '%x');
```

Dieser Zeilenvektor liefert die Quelldaten, mit denen wir unser Modell während des schrittweisen Aufbaus testen. Die anderen Testvektoren importieren wir mit der gleichen Methode nach MATLAB.

Beim Aufbau jedes einzelnen Blocks und der Simulation des Gesamtmodells vergleichen wir die erhaltenen Ausgaben mit den zugehörigen Testvektoren. Obwohl das sehr einfach erscheint, stoßen wir an zwei Stellen auf Probleme. Wenn wir diese nicht

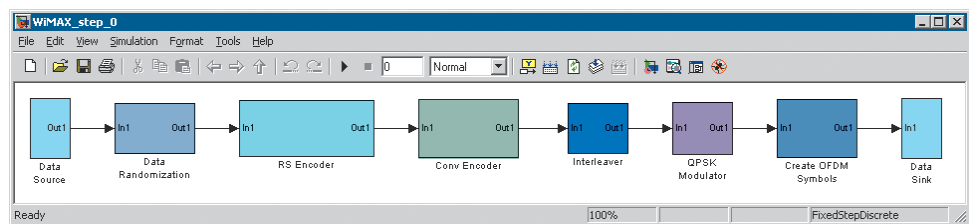


Abb. 1: Das Grundgerüst für den WiMAX-Sender.

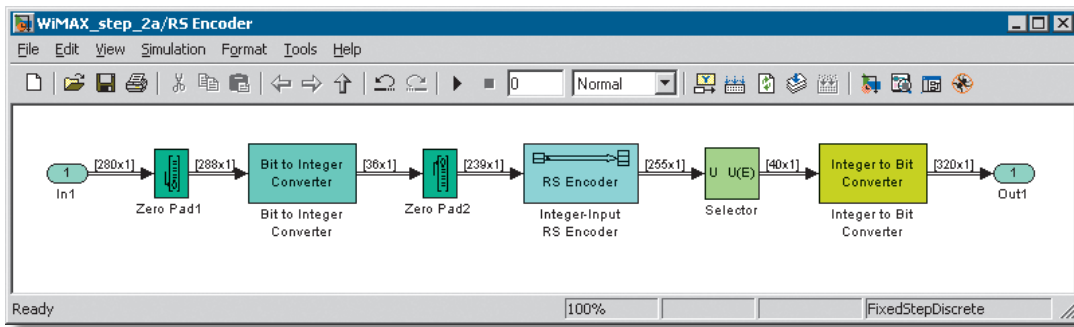


Abb. 2: Der Reed-Solomon-Encoder.

lösen, arbeitet der Sender später nicht mit anderen Systemen zusammen.

Der erste Problempunkt taucht bei der Implementierung des Reed-Solomon- (RS-) Encoders auf. Der Standard legt fest, dass der RS-Encoder mit Hilfe verschiedener Verkürzungs- und Punktierungs-Methoden „von einem systematischen RS(N=255, K=239, T=8)-Code abgeleitet wird“, um die verschiedenen Raten erzeugen zu können. Der erste Testfall verlangt einen RS(40,36,2)-Code.

Wir implementieren den Encoder nach dieser Beschreibung als Subsystem (Abb. 2) und stellen fest, dass unsere Ergebnisse nicht mit dem angegebenen Testvektor übereinstimmen. Der erneute Blick in den Standard zeigt, dass dort das Generatorpolynom für den RS(255,239,8)-Code explizit definiert ist. In unserem Entwurf hatten wir einfach angenommen, dass es mit dem voreingestellten Wert unseres Blocks identisch ist. Wir ersetzen darum diesen Standardwert durch das korrekte Generatorpolynom. Wenn wir nun das Modell simulieren und die Ergebnisse überprüfen, stimmen sie überein.

Der zweite Fehler tritt beim Aufbau des Interleavers auf, der die Daten verschachtelt, um die Fehlertoleranz zu erhöhen. Der Standard definiert den Interleaver mit Hilfe des Gleichungspaares:

$$m_k = (N_{\text{cbps}} / 12) * k_{\text{mod}12} + \text{floor}(k/12)$$

und

$$j_k = s * \text{floor}(m_k / s) = (mk + N_{\text{cbps}} - \text{floor}(12 * m_k / N_{\text{cbps}}))_{\text{mod}(s)}$$

für die gilt:

$$k = 0, 1, \dots, N_{\text{cbps}} - 1$$

Diese Gleichungen lassen sich leicht in MATLAB formulieren:

```
k = 0:Ncbps-1;
mk = (Ncbps/12)*mod(k,12)+...
    floor(k/12);
s = ceil(Ncpc/2);
jk = s*floor(mk/s)+...
    mod(mk+Ncbps-...,
        floor(12*mk/Ncbps),s);
```

Wenn wir nun die Interleaver-Tabelle mit dem Vektor jk spezifizieren und unser Modell simulieren, erhalten wir ein falsches Ergebnis. Wir sehen uns darum den Standard noch einmal an und stellen fest, dass darin mk und jk Schreibadressen sind, während der Interleaver-Block in unserem Modell Leseadressen verlangt. Dieser Fehler lässt sich leicht korrigieren, indem wir den Vektor mit der `Sort`-Funktion von MATLAB passend permutieren.

Beide Fehler waren reine Übersetzungsfehler und wurden nicht durch Mehrdeutigkeiten oder unvollständige Angaben im veröffentlichten Standard hervorgerufen – trotzdem hätten wir sie ohne die Testvektoren leicht übersehen können. Unsere erste Implementierung hätte bei der Bitfehlerrate und bei anderen Tests sogar annehmbare Ergebnisse geliefert. Dennoch hätte sie eine absolut inkompatible Hardware erzeugt.

Aufbau des Empfängers mit Hilfe der ausführbaren Spezifikation

Wie die meisten Kommunikationsstandards spezifiziert auch WiMAX nur die Signalverarbeitung im Sender. Dieser Ansatz garantiert einerseits die Interoperabilität, lässt aber andererseits den Herstellern freie Hand bei der

Wahl der Implementierung ihres Empfängers. Mit dem fertigen Simulink-Modell des WiMAX-Senders können wir einen standardkonformen Empfänger entwickeln. Das Modell sorgt dabei für Kontinuität, vermeidet Fehlinterpretationen und stellt gleichzeitig einen natürlichen Prüfstand für den Empfängerentwurf dar. Auf diese Weise haben wir nicht nur eine kleine Zahl von Testvektoren zur Hand, sondern können praktisch unendlich viele Testfälle mit dem Modell erzeugen.

Eine weitere Anwendung ausführbarer Modelle

Die für die Verabschiedung von Standards zuständigen Einrichtungen stehen vor den gleichen Problemen wie die Entwickler, die die Standards später implementieren. Im ersten Abschnitt des WiMAX-Standards heißt es beispielsweise, dass er eine Vereinigung dreier früherer 802.16-Standards darstellt und dass darin Veränderungen angebracht wurden, die „unkorrektes, mehrdeutiges und unvollständiges Material ersetzen“. Hätte man bei der Entwicklung und Veröffentlichung dieser älteren Standards ausführbare Modelle eingesetzt, dann könnten sie eindeutiger sein, würden weniger manuelle Übersetzungsarbeit erfordern, und die Modelle könnten gleichzeitig die Grundlage für kritische Konformitätstests bilden. ◀

QUELLEN

- ▶ **WiMAX Forum**
www.wimaxforum.org
- ▶ **Webinar: From a Wireless Standards Document to an Executable Model Using MATLAB and Simulink**
www.mathworks.com/res/executablemodel