BY MIKE MULLIGAN

Creating an Executable Specification *for the* WiMAX Standard

In January 2006, the WiMAX Forum announced the first products to complete the rigorous test procedures required for IEEE 802.16-2004-compliant certification. The IEEE 802.16 standard ensures compatibility and interoperability between broadband wireless access components. Nearly 900 pages long, it is a comprehensive document that describes both the physical layer and the media access control layer for WiMAX systems.

The scope and complexity of the standard creates challenges for designers of standard-compliant components. Model-Based Design with Simulink® addresses these challenges by placing a system model at the center of the development process—from requirements capture to implementation and test. The model is an executable specification that functions as an interactive test harness throughout the design workflow.

In a document-based design approach, requirements are assembled in a product specification that is subsequently elaborated, partitioned, and translated into subspecifications for each design team. Every new document is a translation of the requirements, and can introduce errors or omissions that may not be discovered until compliance testing. An executable specification enables continuous test and verification, making errors of translation much easier to identify and correct.

In this article we use Model-Based Design to build an executable specification for a WiMAX transmitter, focusing on the channel coding described in section 8.3.3. of the standard.

Building and Testing the Channel Coding Model

Since many of a communication standard's technical requirements are mathematical, MATLAB® and Simulink are well suited to the task of creating an executable specification. Creating a Simulink model is itself a translation of the WiMAX standard, however, and so we must proceed carefully. Fortunately, section 8.3.3 includes test cases that we can use to build and test our model step by step.

The channel coding section of the WiMAX standard includes five steps: data randomization, forward error correction (FEC), interleaving, modulation, and OFDM symbol creation. Starting with a framework of empty subsystems (Figure 1) we build up the transmitter block by block, using data from the standard to check each block before proceeding.

In the first WiMAX test case, the input data vector is 35 bytes long and given in hex notation. It is easily expressed in MATLAB using the `sscanf` function to create a row vector:

```
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');
```

This row vector is the source data that we will use to test our model as we incrementally build it. We use a similar technique to import the other test vectors into MATLAB. As we build each block and run the Simulink model, we compare the generated output with the corresponding test vector. This is a straightforward procedure, but

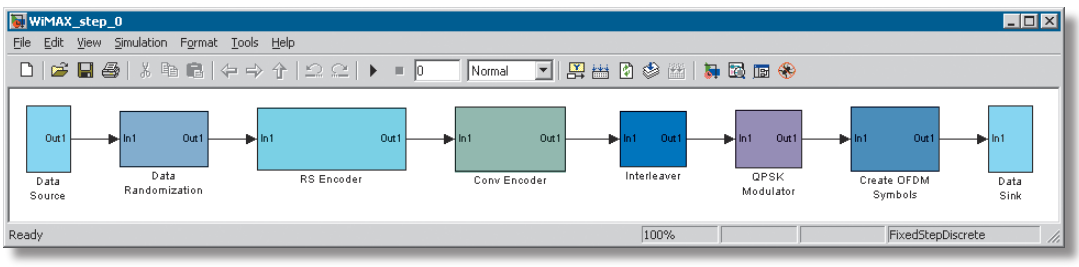

Figure 1. The framework for the WiMAX transmitter.

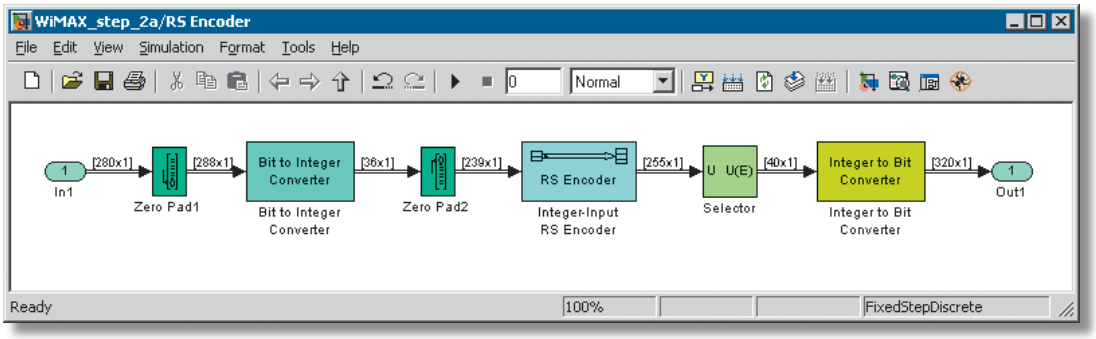

Figure 2. Reed-Solomon encoder.

in two places we encounter problems that if unresolved would result in a non-interoperable design.

The first trouble spot comes in the implementation of the Reed-Solomon encoder. The standard states that the Reed-Solomon encoder is "derived from a systematic RS(N=255,K=239,T=8) code" using a variety of shortening and puncturing techniques to achieve different rates. The first test case calls for a RS(40,36,2) code.

Based on the description in the standard, we implement the encoder as a subsystem (Figure 2).

Our results, however, do not match the published test vector. Returning to the standard we see that the generator polynomial for the RS(255,239,8) code is explicitly defined. In our design we had assumed that it was the same as the default in our block. We replace the default value with the explicit definition of the generator polynomial in the standard. When we rerun our model and check the results in the MATLAB workspace, the results match.

The second error occurs while we are building the interleaver. The standard defines the interleaver using a pair of equations:

$$m_k = (N_{cbps} / 12) * k_{mod12} + floor(k/12)$$

and

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

where

$$k = 0, 1, ..., N_{cbps} - 1$$

The equations are easily translated into MATLAB as

```
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);
```

Using the vector *jk* to specify the interleaver table, we run our Simulink model and get the wrong result. Upon returning to the standard we see that *mk* and *jk* are write addresses, whereas the interleaver block in our model requires read addresses. The error is easily corrected using the MATLAB `sort` function to obtain the required permutation vector.

While these were errors in translation, and not the result of ambiguity or omission in the published standard, without the test vectors to check against we could easily have overlooked them. In that case, while the initial implementation would have performed satisfactorily in bit error rate analysis and other system-level tests, it would still have produced a non-interoperable piece of equipment.

Using the Executable Specification to Design the Receiver

Like most communication standards, WiMAX specifies the signal processing in the transmitter only. This approach guarantees interoperability while letting manufacturers select their own receiver implementation.

Once we have a complete Simulink model of the WiMAX transmitter, we can use it to design a standard-compliant receiver. Our model not only provides continuity and prevents misinterpretation of the standard, it is also a natural test harness for the receiver design—and rather than relying on the handful of test vectors supplied in the printed standard, we can use the model to generate an inexhaustible supply of test cases.

A Further Use for Executable Models

The institutions that set standards confront the same challenge of translation as the designers who implement them. For example, the first paragraph of the WiMAX standard indicates that it is a consolidation of three earlier 802.16 standards, noting that changes were made to "replace incorrect, ambiguous or incomplete material." If an executable model were part of the standards development and publishing process, it could reduce ambiguity in the standard, minimize the need for error-prone translation, and provide the basis for the critical compliance testing phase.

RESOURCES

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