製造までの4つのステップ：
モデル・ベース設計で実現するソフトウェア無線

Part 2：MATLABとSimulinkによるモードS信号の検出とデコード

著者：Mike Donovan/Andrei Cozma/Di Pu

ADS-Bの概要

検出やデコード（復号化）の対象となるワイヤレス信号は至るところに存在します。そうした信号には、今日提供されているソフトウェア無線（SDR：Software-defined Radio）対応のハードウェアを使えば簡単にアクセスすることができます。代表的なハードウェアとしては、アナログ・デバイセズ（ADI）が提供するRFアジャイル・トランシーバ（RF Agile Transceiver™）IC「AD9361」、「AD9364」が挙げられます[1, 2]。例えば、AD9361とXilinx®社の「Zynq®-7000 All Programmable SoC（以下、Zynq SoC）」を組み合わせることで、SDR向けのラピッド・プロトタイピングを実現できます。プロトタイプのデモやテストには、民間航空機からワイヤレスで送信されるADS-B（Automatic Dependent Surveillance-Broadcast：放送型自動従属監視）信号を使用できます。民間航空機は、ADS-Bに対応するトランスミッタを使用して自らの位置、速度、高度、航空機IDを航空交通管制局に報告します[3]。飛行情報のデータ形式は、国際民間航空機関（ICAO）の規格「Mode S Extended Squitter（モードS拡張スキッタ）」によって定義されています[4]。ADS-Bは、航空交通管制システムや衝突回避システムの改善を目的として世界中で導入されつつあります。欧州ではすでに採用されており、米国でも徐々に導入が始まっています。

モードS拡張スキッタ規格では、RF伝送の形式と、デコードの対象となるデータ・フィールドの詳細を定めています。トランスポンダを利用した伝送の仕様は以下の通りです。

- 送信周波数：1090MHz
- 変調方式：PPM（パルス位置変調）
- データレート：1Mbps
- メッセージ長：56μsまたは112μs
- 誤り検出：24ビットのCRC（巡回冗長検査）チェックサム

チューニング周波数と帯域幅は、十分にAD9361の対象範囲内にあります。受信したI/Q信号のサンプル（サンプリングしたデータ）は、さまざまなソフトウェアや組み込みプラットフォームによって検出し、デコードすることができます。

本稿では、AD9361をベースとするレシーバ用プラットフォームによってモードSの信号を取得し、「MATLAB」と「Simulink®」を使用して、そのメッセージをデコードするためのアルゴリズムを開発する方法を説明します。アルゴリズムは、Avnet社の「PicoZed™ SOM（SDR System on Module）」をはじめとするZynq SoC対応のプラットフォームに配備（デプロイ）することを最終的な目標として開発します。

レシーバの設計における課題

モードSのメッセージには、ショート形式（56μs）とロング形式（112μs）の2種類があります。ショート形式のメッセージは、メッセージの種類、航空機の識別番号、CRCチェックサムで構成されます。ロング形式のメッセージには、さらに高度、位置、速度、飛行状態の情報が含まれます。どちらの形式でも、モードSの信号は8μsのプリアンブルで始まります。レシーバは、このプリアンブル・パターンを使用して、有効なメッセージが送信されていることを確認するとともにメッセージ・ビットの開始位置を判断します。詳細については図1をご覧ください[5]。

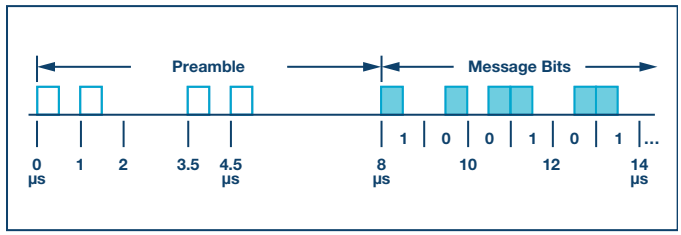

図1. モードSのメッセージの構造

モードSの信号は非常にシンプルです。しかし、送信されたメッセージを正しく受信してデコードするには、以下に示すような複数の課題を解決する必要があります。

- 一般的な受信環境では、長いアイドル期間の中に非常に短いメッセージが散在する状態になる。また、送信元の航空機がレシーバから遠く離れている場合、受信信号は非常に弱くなる可能性がある。加えて、従来から使われていない信号が同じ1090MHzで送信されることもある。レシーバは、混雑した周波数帯の中でプリアンブルを活用し、大振幅、小振幅のモードS信号を検出しなければならない
- ビット区間は1μsであり、その区間のビット・パターンは2種類のうちどちらかになる。前半の0.5μsがオンで後半の0.5μsがオフの場合が論理レベルの1を意味し、前半の0.5μsがオフで後半の0.5μsがオンの場合が論理レベルの0を意味する。ビットの判定は、時間をベースとしたパターンに基づいて行われる。そのため、レシーバはプリアンブルを使用し、メッセージ・ビットの開始位置となるI/Qサンプルを正確に検出する必要がある
- モードSのメッセージは、88ビットの情報と24ビットのチェックサムで構成される。レシーバは、レジスタのクリア、ビットの判定、チェックサムの計算、チェックサム・レジスタの読み出しの各動作を正しいタイミングで行わなければならない。レシーバが正しく機能するためには、タイミング制御が必須となる
- 組み込み設計の場合、サンプル単位でデコード処理を実行する必要がある。大量のデータを保存しておき

一括で処理するのは、組み込みシステムにおけるレシーバの設計として現実的ではない

AD9361のような高性能のRFフロント・エンドと、MATLAB®のようなテクニカル・コンピューティング言語を組み合わせることにより、上述したような信号の検出/デコードに伴う問題が大幅に簡素化されます。MATLABや「Signal Processing Toolbox」の機能を利用すれば、同期パターンの識別、ノイズフロアの計算、ビットの判定、チェックサムの計算を容易に行うことができます。また、MATLABの条件制御や実行制御の機能によって制御ロジックを簡素化することが可能になります。テスト用のデータへのアクセスも容易であり、バイナリ形式/テキスト形式のファイルから読み出すか、AD9361をベースとするSDR用プラットフォームを使用してMATLABに直接ストリーミングすることができます。MATLABはインタプリタ方式で実行されるのでデータを操作したり別のアプローチを試したりすることが容易であり、インタラクティブにソリューションを開発できます。

MATLABによるモードS用レシーバのアルゴリズムのモデル化と検証

以下で説明する内容を実装したMATLABソース・コードは、GitHubリポジトリから入手できます。エントリ・レベルの関数はad9361_ModeS.mであり、この関数から呼び出されるファイルも提供されています。

レシーバ用のアルゴリズムにおける最初のステップは、ソースとなるデータにアクセスすることです。現在では、多くの航空機がモードS用のトランスポンダを備えています。そのため、1090MHzの送信周波数に対してレシーバをチューニングするだけで近くの信号を捉えることができます。これについては、Zynq SoCを利用したSDR向けのラピッド・プロトタイピング・プラットフォームを使用します。ADIは、Ethernetを介して「AD-FMCOMMSx-EBZ」（FMCOMMSプラットフォーム）からのデータを受信できるMATLAB対応の「System Object™」を提供しています[6]。このSystem Objectにより、チューニング周波数とサンプリング・レートを選択し、無線用のハードウェアで受信サンプルを収集して、MATLABのワークスペースにMATLAB変数としてそれらを直接取り込むことができます。記述しなければならないコードはほんのわずかです。MATLAB対応のSystem Objectをセットアップするために数行、FMCOMMSプラットフォーム（AD-FMCOMMS3-EBZ）の設定を行うために数行、I/Qサンプルを取得してMATLAB変数に書き込むために数行を記述するだけです。図2、図3、図4にMATLABのサンプル・コードを示しました。

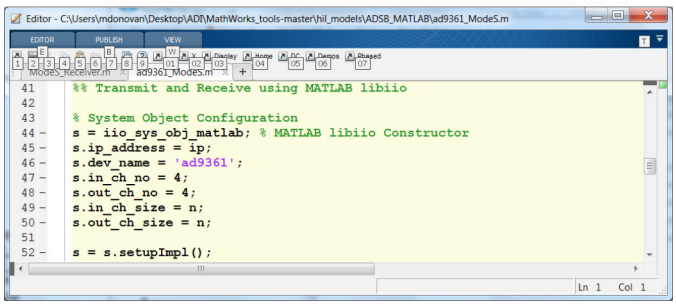

図2. System Objectを設定するためのサンプル・コード

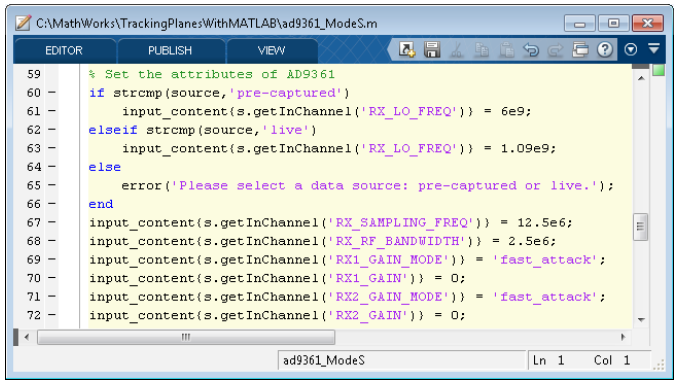

図3. AD-FMCOMMS3-EBZのボードを構成するためのサンプル・コード

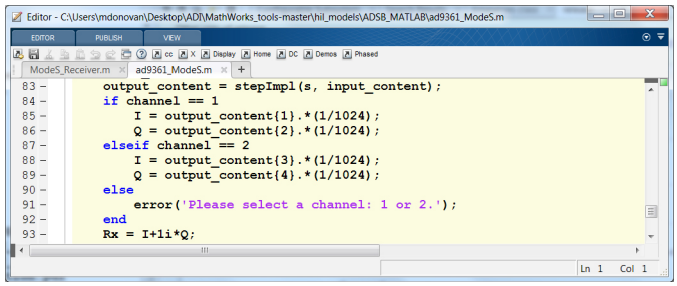

図4. I/Qサンプルを取得して受信変数に書き込むためのサンプル・コード

コマンドに基づくこれらのコードにより、12.5MHzのサンプル・レートで複数のデータ・セットを取得しました。12.5MHzのサンプル・レートを選択したのは、十分な数のサンプルを取得し、プリアンブルの位置をメッセージの先頭のビットに正確に合わせて、ビットの判定に用いるサンプル内の一部のノイズを平均化するためです。100万個のサンプルを取得した結果を図5に示しました。

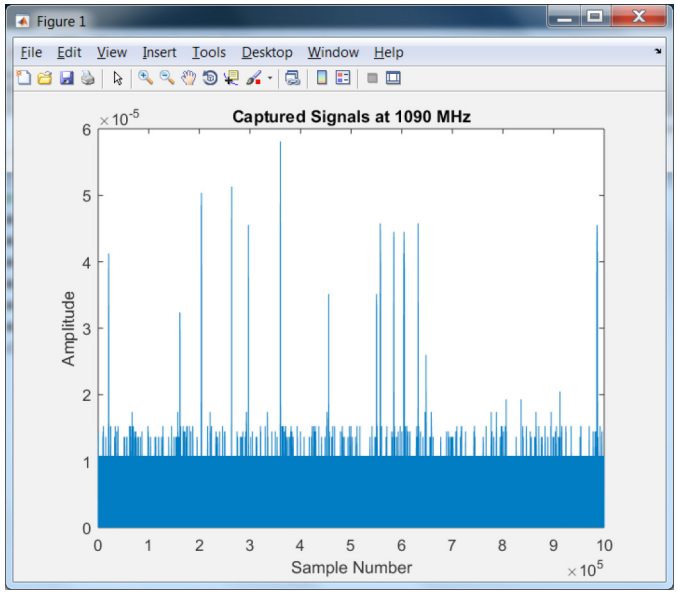

図5. 1090MHzの送信周波数に対応して取得したサンプル・データ

このデータ・セットの中には、ノイズフロアを上回るレベルの信号が14個あります。これらのうち2個がモードSのメッセージです。残りは従来から使われていた別の信号やスプリアス信号であり、除去する必要があります。サンプル番号が604000付近の領域を拡大すると、有効なメッセージのうちの1つを確認することができます（図6）。

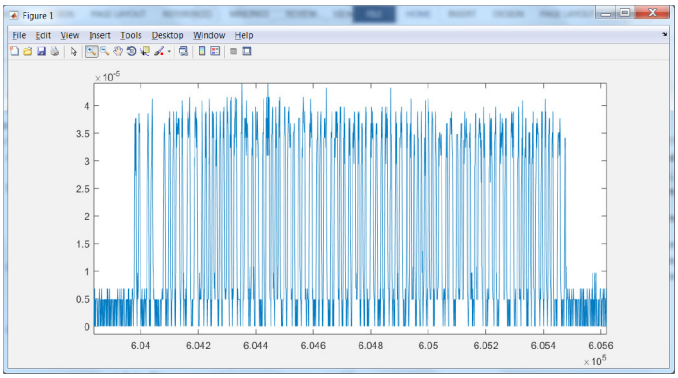

図6. モードSのメッセージの1つ

このグラフでは、プリアンブルを明瞭に確認することができるほか、PPMによるビットの移動もはっきりと見てとれます。このように信号がクリーンである場合でも、目視でビットのデコードを行うには、優れた視力とかなりの忍耐が必要になります。当然のことながら、メッセージのデコードにはそれを自動的に行うプログラムが必須だということです。MATLABはそのプログラムを開発するための適切なソリューションです。

モードSのメッセージを受信し、それをデコードするためのMATLABのコードは、以下のような流れになります。

1. 短い時間窓にfilter関数を適用し、ノイズフロア、プリアンブルのコリレーションを計算する。この例では、6μsに相当する75サンプルを使用した

2. プリアンブルのコリレーションがノイズフロアを大きく上回った時点で、メッセージの先頭ビットに当たるサンプルを検出するためのロジック処理を開始する
   a. この処理の閾値は、状況に応じて選択する。小さすぎると弱い信号を検出できないが、大きすぎると誤検出が多発してしまう。この例では、デコードが可能なほとんどのメッセージを取得できる妥当な閾値として、ノイズフロアの10倍の値を選択した
   b. プリアンブルのパターンにより、ピークが複数生成される。最大のピークは最初の6μsの間に現れるので、最初のピークの値を保存してメッセージの先頭ビットの検索を開始し、次の3μsの間にさらに大きな別のピークが現れるかどうかを確認する。大きな別のピークが現れたら、その値を保存してメッセージの先頭ビットの検索をやり直す
   c. 最大のピークが現れたら、その2μs後からメッセージ・ビットのデコードを開始する
   d. 図7は、ノイズフロア（緑の線）と共に、入力データに対する理想的なプリアンブルとのコリレーションの結果を示したものである。ノイズフロアを上回る複数のピークが存在するが、ここでは最大振幅を持つピークに注目する。そのピークの2μs後に、メッセージの先頭ビットのサンプルが到達する

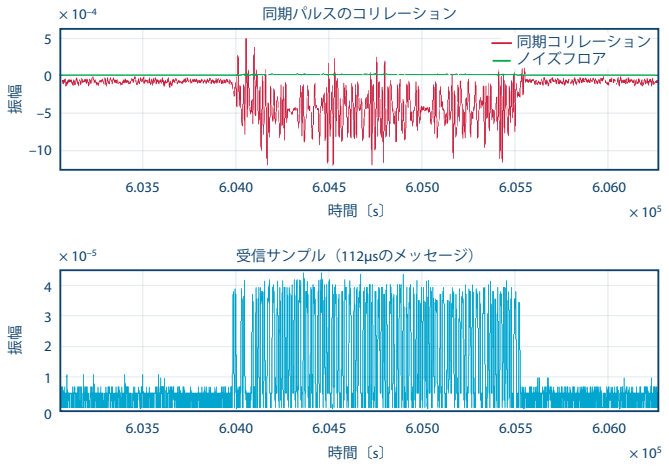

図7. ノイズフロアとプリアンブルのコリレーションの算出結果

3. 個々のビットに対し、最初の0.5μsと次の0.5μsでサンプルの振幅を合算する。どちらの合計値が大きいかによって、そのビットが論理値の1であるか0であるかを判定する

4. ビットの判定に基づいてチェックサムを計算する。それには、最初のビットが到達したときにCRCレジスタをリセットし、88ビットに対してチェックサムを計算して、最後の24ビットでCRCレジスタを空にする制御ロジックが必要になる。受信ビットがチェックサムに一致すれば、そのADS-Bメッセージは有効である

5. モードSの規格に従って、メッセージ・ビットをパースする（図8）

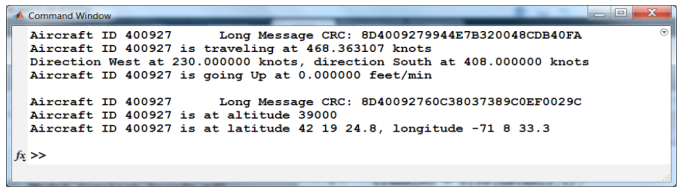

図8. デコードされたモードSのメッセージ

図8に示したのは、MATLABのコマンド・ウィンドウです。100万サンプルから成るデータ・セットを基に正しくデコードされた2つのメッセージが示されています。88ビットのメッセージと24ビットのチェックサムが16進数で表示され、デコードの結果として、航空機ID、メッセージの種類、飛行速度、高度、位置が示されています。

MATLABは演算と信号処理のための強力な言語であり、このような処理を容易に実現することができます。MATLABで記述したわずか200行のコードにより、サンプル・データに処理を施し、最終的にメッセージをデコードすることが可能になります。また、MATLABはインタプリタ方式で実行されるので、設計上のアイデアをインタラクティブに試し、実現可能なソリューションを素早く見極めることができます。この例では、さまざまなデータ・セットに対し、複数のタイミング構成、閾値、ノイズ・レベルを試すことで、最終的に適切なプログラムを開発することができました。

MATLABで開発したコードのテストは、近くの空域を飛行する航空機からの信号を使って実施しました。受信した信号をデコードして得たメッセージを、airframes.

orgやflightaware.comなどから入手した情報と照らし合わせることで、正しく処理が行えたことを確認しました。ハードウェアとコードは適切に動作し、50マイル（約80km）離れた航空機からの信号を正しくデコードすることができていました。

実装に向けた作業

以下で説明する内容を実装したSimulinkモデルのファイルはGitHubリポジトリから入手できます。

https://github.com/analogdevicesinc/MathWorks_tools/tree/master/hil_models/ADSB_Simulink

MATLABは、設計上のアイデアをテストするためにPC上でアルゴリズムを実行するには最適な環境です。ただ、最終的な目標が、Zynq SoCなどの組み込みプラットフォーム上で使用するソフトウェアやHDLコードを生成することであるならば、Simulinkの使用が適しています。Simulinkは、プログラマブルなデバイスをターゲットにする際に必要なハードウェアに固有の処理をモデル化する場合に最適です。推奨されるワークフローは、MATLABでアルゴリズムの開発/検証を行い、その設計をSimulinkのモデルに変換して、ハードウェアへの実装に向けて開発を進めるというものです。

この例のアルゴリズムを実現したMATLABのコードは、データをサンプル単位で処理します。そのため、このコードは、かなり簡単にSimulinkのモデルに変換することができます。SimulinkのモデルではMATLABで記述した200行のコードがシンプルに表示/表現されます（図9）。

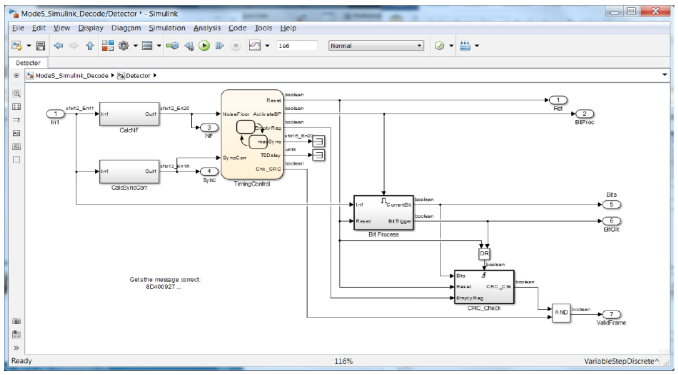

図9. モードSの信号を検出/デコードするアルゴリズムに対応したSimulinkのモデル

図9からは、デコードのための最初のステップが、ノイズフロアと、プリアンブルのコリレーションの算出であることが見てとれます。これらの計算には、デジタル・フィルタ・ブロックが使用されています。またタイミング制御用のブロックは「Stateflow®」で実装されています。Stateflowはステート・マシン・ツールであり、モデルにおいて制御ロジックをデータ・フローと分離したい場合に非常に有用です。この例では、デコード用のアルゴリズムにおけるその他の部分に向けて、タイミング信号、リセット信号、制御信号を生成するために使用しています。タイミングとトリガがアクティベートされると、「BitProcess」という名前のブロックが、入力されたI/Qのサンプルを受け取ってデータ・ビットを計算します。併せて、「CRC_Check」というブロックがチェックサムを計算します。なお、メッセージのパースは、このSimulinkモデルによって駆動されるMATLABのスクリプトによって行われます。

モデルをさらに詳しく見てみると、Simulinkが組み込み開発に適していることを示すいくつかの特徴に気づきます。特に、Zynq SoCをターゲットとして機能を実現するために、適切に設計の分割を行い、それぞれに対応するHDLのコードとC言語のコードを生成するうえで非常に有用です。

1. Simulinkは、固定小数点のサポートに優れているため、ビット精度が設計と一致する回路を構築し、テストを行うことができる。また個々のブロックで、モデルにおける算術演算のワード長と小数点以下の桁数を設定可能である。その良い例として、プリアンブルのコリレーションの計算に使用するデジタル・フィルタ・ブロックが挙げられる（図10）。演算における丸めのモードとオーバーフローが発生した際の動作も設定できる（HDLコードで行われる演算については、FloorとWrapが最も簡単な選択肢となる）。また、フィルタの乗算器やアキュムレータの演算に対しては異なるワード長や小数精度を指定できる（図11）。レシーバのA/Dコンバータに適合するようワード長を選択することにより、ハードウェア乗算器（例えばZynq SoCが備えるDSP48スライス内の18ビット×25ビットの乗算器）を活用することが可能になる

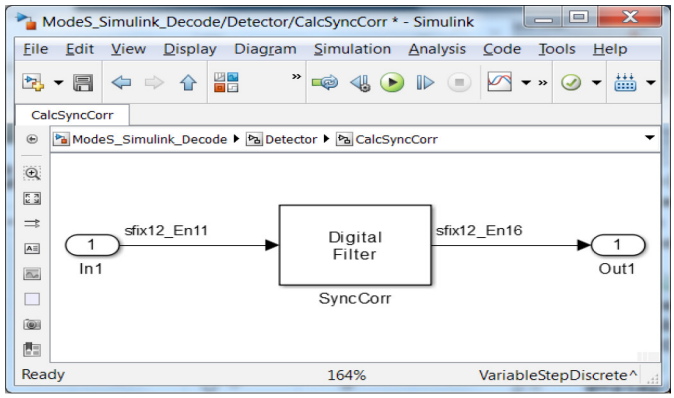

図10. Simulinkのデジタル・フィルタ・ブロック（12ビットのデータ型、プリアンブルのコリレーションの算出に使用）

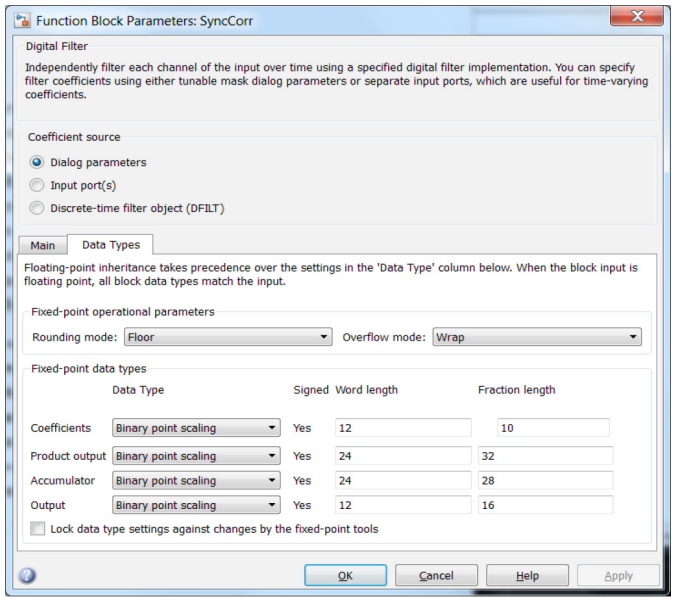

図11. 固定小数点のデータ型の設定

2. 多くの場合、組み込み設計では多数の動作モードが設けられる。各モードには、条件に応じて実行されるアルゴリズムが対応づけられる。Stateflowは、このような制御信号の管理に特に適している。この例で言えば、Stateflowを使うことで、モードSのメッセージの検出/デコードに必要な制御ロジックを視覚的に確認することができる。図12には、このロジックにおける以下のステートが示されている
   - SyncSearch: 取得したサンプル内のプリアンブルを検索する
   - WaitForT0: メッセージの先頭ビットの開始位置を検索する
   - BitProcess: ビット処理を有効化する
   - EmptyReg: チェックサム・レジスタを空にし、ビット・データをビット処理の出力と比較する

異なるステート間を遷移する検出/デコード用アルゴリズムの流れに従って、Stateflowのブロックは、ビット処理を有効化する信号を生成し、ビット判定用のカウンタとチェックサム・レジスタをリセットし、モードSのメッセージの最後に配置されたチェックサム・ビットを読み取ります。

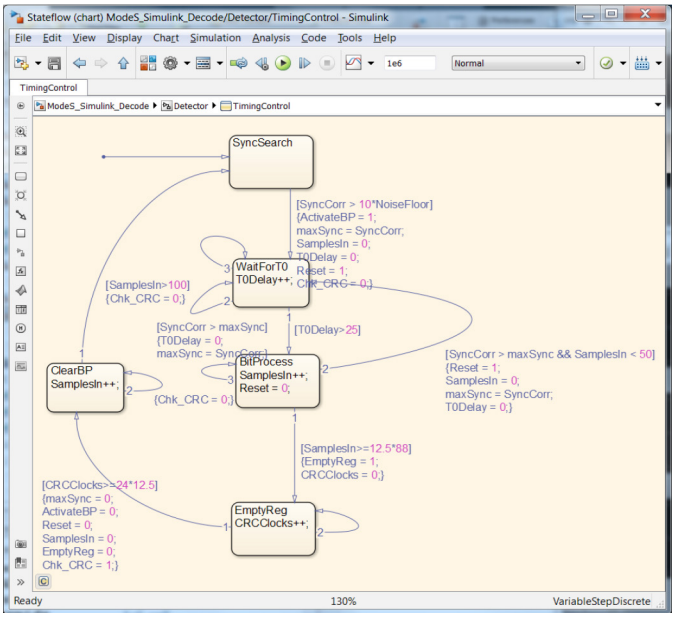

図12. モードSのメッセージをデコードするための Stateflowのチャート

3. Simulinkのブロック・ライブラリを利用することにより、技術者は非常に高いレベル、あるいは非常に細かいレベルで作業を行うことが可能になる。Simulinkの高レベル・ブロックとしては、信号処理回路の構築を簡素化するDigital Filter、FFT、Numerically Controlled Oscillatorなどが挙げられる。速度や面積の最適化など、よりきめ細かく回路を制御する必要がある場合には、Unit Delay、Logic Operator（XORなど）、Switchといった低レベル・ブロックを使用すればよい。この例のモデルでは、24ビットのチェックサムが、低レベル・ブロックで構築されたフィードバック・シフト・レジスタによって計算される（図13）

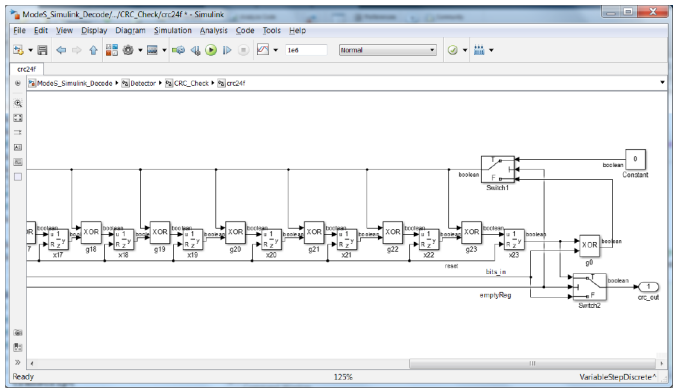

図13. モードSのチェックサムを計算するための フィードバック・シフト・レジスタ

このSimulinkモデルは、モードSのメッセージを検出/デコードするMATLABアルゴリズムのハードウェア向けバージョンだと言えます。Simulinkは、MATLABで記述された動作アルゴリズムと組み込みハードウェア用の実装コードの間のギャップを埋めるために役立つツールです。ハードウェア固有の処理をSimulinkモデルとして構築して実行することで、加えた変更がデコード用のアルゴリズムに影響を与えないことを確認できます。

まとめ

Zynq SoCを利用したSDR向けのラピッド・プロトタイピング・プラットフォームとMathWorks社のソフトウェアを組み合わせることにより、ワイヤレス・レシーバの設計においてアイデアを基に直ちにプロトタイプを開発するための柔軟かつ新たな手段が得られます。その中心にあるのは広帯域に対応するRFアジャイル・トランシーバであるAD9361/AD9364です。両製品が備える高いプログラマビリティや性能に加え、MATLAB環境とハードウェアの間のシンプルな接続性によって、広範なワイヤレス信号を対象として扱うことができます。MATLABを使用すれば、設計上のさまざまなアイデアを直ちに試し、有望なソリューションを見極めることが可能になります。設計の最終的なターゲットが組み込みプロセッサである場合、Simulinkを使用すれば、そのハードウェアに適したアイデアに基づいて設計を改良し、最終的にはそのプロセッサで使用するコードを生成できます。このワークフローによって、ワイヤレス・レシーバの設計に必要なスキルが軽減され、概念設計から実際に動作するプロトタイプを開発するまでの時間を短縮することが可能になります。

本稿の続編となるPart 3では、HIL（Hardware in the Loop）を使用してレシーバの設計を検証する方法を取り上げます。検証に使用するSimulinkのホスト上で信号処理システムのモデルを実行しつつ、ターゲットとなるトランシーバで信号を取得する方法を説明します。

参考文献

1 「AD9361」 Analog Devices

2 「AD9364」 Analog Devices

3 「960-1164 MHz」 National Telecommunications and Information Administration

4 「Technical Provisions for Mode S Services and Extended Squitter」 International Civil Aviation Organization

[5] 「Surveillance and Collision Avoidance Systems」 Aeronautical Telcommunications, Volume IV. International Civil Aviation Organization

[6] Di Pu, Andrei Cozma, Tom Hill「製造までの4つのステップ：モデル・ベース設計で実現するソフトウェア無線、Part 1：ADI/Xilinx社のSDR向けラピッド・プロトタイピング用プラットフォーム――その機能、メリット、開発ツールについて学ぶ」Analog Dialogue 49-09

ソース・コードとモデルへのリンク

MATLABによるモードS信号のデコード用アルゴリズム:
https://github.com/analogdevicesinc/MathWorks_tools/blob/master/hil_models/ADSB_MATLAB/

Simulinkによるモードs信号のデコード用モデル:
https://github.com/analogdevicesinc/MathWorks_tools/tree/master/hil_models/ADSB_Simulink

謝辞

本稿の例に使用されているMATLABコードの一部を作成していただいたMathWorks社のMike Mulligan氏に感謝いたします。

著者：

Mike Donovan（mike.donovan@mathworks.com）は、MathWorks社のアプリケーション・エンジニアリング・グループでマネージャーを務めています。バックネル大学で電気工学の学士号を、コネチカット大学で修士号を取得しています。MathWorks社に入社する前は、レーダー/衛星通信システム分野、ブロードバンド通信分野の業務に従事していました。

Mike Donovan

Andrei Cozma（andrei.cozma@analog.com）は、ADIのエンジニアリング・マネージャーとしてシステム・レベルのリファレンス設計の開発を支援しています。産業用オートメーションと情報科学に関する学士号に加えて、電子工学と電気通信工学の博士号を取得しています。モーター制御、産業用オートメーション、ソフトウェア無線（SDR）、電気通信など、さまざまな分野にわたって設計/開発プロジェクトに従事した経験を持ちます。

Andrei Cozma

この著者が執筆した他の技術文書

FPGAベースのシステムで、モーター制御の性能を向上

Analog Dialogue 49-03

Di Pu（di.pu@analog.com）は、ADIのシステム・モデリング・アプリケーション・エンジニアです。ソフトウェア無線（SDR）に対応するプラットフォームやシステムの設計/開発をサポートしています。特にMathWorks社と密に連携することで、両社の顧客が抱える課題の解決に取り組んでいます。2007年に中国南京にある南京理工大学（NJUST）で理学士の学位を取得しています。また、マサチューセッツ州ウースターにあるウースター工科大学（WPI）で、2009年に電気工学の修士号、2013年に博士号を取得しました。WPIでは、2013年の「Sigma Xi Research Award for Doctoral Dissertation」を受賞しています。

Di Pu