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 3：HILによるモードS信号のデコード用アルゴリズムの検証

著者：Di Pu/Andrei Cozma

はじめに

本シリーズのPart 2では、「MATLAB」と「Simulink」を使って信号処理用のアルゴリズムを実装する方法について説明しました。次のステップでは、そのアルゴリズムを実行するソフトウェア無線（SDR：Software-defined Radio）対応のハードウェア・システムによって実際にデータを取得します。システムから取得した複数の入力データ・セットを使用し、アルゴリズムの機能検証を実施するということです。これにより、アルゴリズムが正しく機能することが確認できます。ただし、それだけではデータを取得したときとは異なる環境条件においてもアルゴリズムが期待どおりに動作するということまでは保証されません。また、SDRシステムのアナログ・フロントエンドとデジタル・ブロックを異なる設定にした場合に、どのような動作と性能が得られるのかということもわかりません。このようなすべての側面を含めて検証を行うためには、アルゴリズムをオンラインで実行し、ライブ・データを入力として受け取って、最適な性能が得られるようにSDRシステムの設定を調整できるようにすべきです。本シリーズ[1]のPart 3となる本稿では、まずMATLABやSimulinkのモデルと、SDR対応のラピッド・プロトタイピング用プラットフォーム「AD-FMCOMMSx-EBZ（以下、FMCOMMS）」との間の直接的なやり取りを可能にするためにアナログ・デバイセズ（ADI）が提供するソフトウェア・ツールについて説明します。そのうえで、各ツールを使用して本シリーズのPart 2[2]で示したADS-Bモデルの検証方法を紹介します。

MATLABとSimulinkのIIO System Object

ADIは、MATLABやSimulinkのモデルと、Linuxが稼働するFPGA/SoCシステムに接続されたFMCOMMSプラットフォームとの間で相互かつリアルタイムにやりとりできるようにするために、完全なソフトウェア・インフラを提供しています。それが本稿の主要なテーマとなる「IIO System Object™」[3]です。IIO System Objectは、TCP/IPを介してハードウェア・システムとデータ交換するように設計されています。これによって、ターゲットとの間でデータをストリーミングし、ターゲットの設定を制御したり、RSSI（受信信号強度）などのさまざまなパラメータを監視したりすることが可能になります。図1に、このソフトウェア・インフラの高レベル・アーキテクチャと、システム内のコンポーネント間におけるデータの流れを示しました。

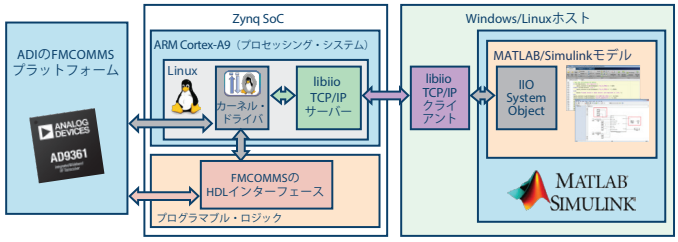


図1. ソフトウェア・インフラのブロック図

IIO System Objectは、MathWorks社が定めた「System Object」[4]の仕様をベースにしています。これを利用することにより、データを公開し、MATLAB/SimulinkモデルがIIO（Linux Industrial I/O）ベースのプラットフォームと通信するためのインターフェースを制御することができます。これらのインターフェースは、System Objectのインターフェースを IIOデータ・チャンネルまたはIIO属性にリンクさせる構成（コンフィギュレーション）ファイルによって定義されます。これにより、IIO System Objectの実装に汎用性がもたらされ、構成ファイルを変更するだけで任意のIIOプラットフォームに対応できるようになります。ADIは、GitHubリポジトリ[5]によって構成ファイルやサンプルを提供しています。対象とするプラットフォームとしては、「AD-FMCOMMS2-EBZ」、「AD-FMCOMMS3-EBZ」、「AD-FMCOMMS4-EBZ」、「AD-FMCOMMS5-EBZ」の各FMCOMMSプラットフォーム（ボード）や、高速データ・アクイジション用の「AD-FMCDAQ2-EBZ」などがあります。IIO System Objectとターゲットの間の通信は、サーバー/クライアント向けのインフラである「libiio」によって実現されます。libiioサーバーはLinuxが稼働する組み込みターゲット上で動作し、ターゲットとローカル/リモート・クライアントとの間のリアルタイムでのデータ交換を管理します。libiioのライブラリは、ハードウェアの低レベルの詳細情報を抽象化し、高度なプロジェクトにも利用できるシンプルでありながら完全なプログラミング・インターフェースを提供します。そのため、多様な言語（C言語、C++、C#、Python）とのバインディングが可能です。

次のセクションでは、IIO System Objectを使用して、MATLABとSimulinkで作成したADS-Bモデルを検証する例を紹介します。図2は、AD-FMCOMMS3-EBZ[6]を、ADIのLinuxディストリビューションを実行する「ZedBoard[7]」に接続した様子を表しています。このAD-FMCOMMS3-EBZが、ADS-B信号の検出/デコード（復号化）に使用するアルゴリズムの動作を検証するためのSDR対応ハードウェア・システムとして機能します。

図2. ADS-B信号用のアルゴリズムを検証するためのハードウェア

IIO System Objectを用い、MATLABベースのADS-Bアルゴリズムを検証

ここでは、AD-FMCOMMS3-EBZで取得したリアルタイム・データに、MATLABで開発したADS-B信号のデコード用アルゴリズムを適用し、その動作を確認します。それに向けて、以下の処理を実行するMATLABスクリプトを作成しました。

- ユーザーからの入力に基づき、地球上のどこに位置するのかを計算する
- IIO System Objectを生成して構成を行う
- IIO System Objectを介し、AD-FMCOMMS3-EBZのアナログ・フロントエンドとデジタル・ブロックの構成を行う
- IIO System Objectを使用して、AD-FMCOMMS3-EBZからデータ・フレームを受信する
- ADS-B信号を検出してデコードする
- デコードによって得られた情報を表示する

IIO System Object（のインスタンス）を生成したら、SDRシステムのIPアドレス、ターゲットとなるデバイスの名前、入出力チャンネルのサイズと番号を設定します。図3に、MATLABによりIIO System Objectを生成して設定する例を示しました。

```
% System Object Configuration
s = iio_sys_obj_matlab; % MATLAB libiio Constructor
s.ip_address = ip;
s.dev_name = 'ad9361';
s.in_ch_no = 4;
s.out_ch_no = 4;
s.in_ch_size = n;
s.out_ch_size = n;

s = s.setupImpl();
```

図3. MATLABによるIIO System Objectの生成と設定

次に、IIO System Objectを使用し、AD9361の属性を設定してADS-B信号を受信します。AD9361の属性は、以下に説明する点を考慮して設定します。

```
% Set the attributes of AD9361
if strcmp(source,'pre-captured')
    input_content{s.getInChannel('RX_LO_FREQ')} = 6e9;
elseif strcmp(source,'live')
    input_content{s.getInChannel('RX_LO_FREQ')} = 1.09e9;
else
    error('Please select a data source: pre-captured or live.');
end
input_content{s.getInChannel('RX_SAMPLING_FREQ')} = 12.5e6;
input_content{s.getInChannel('RX_RF_BANDWIDTH')} = 4e6;
input_content{s.getInChannel('RX1_GAIN_MODE')} = 'fast_attack';
input_content{s.getInChannel('RX1_GAIN')} = 0;
input_content{s.getInChannel('RX2_GAIN_MODE')} = 'fast_attack';
input_content{s.getInChannel('RX2_GAIN')} = 0;
input_content{s.getInChannel('TX_LO_FREQ')} = 6e9;
input_content{s.getInChannel('TX_SAMPLING_FREQ')} = 12.5e6;
input_content{s.getInChannel('TX_RF_BANDWIDTH')} = 4e6;
```

図4. MATLAB libiioによるAD9361の属性の設定

AD9361をベースとするプラットフォームにおいて、サンプリング・レートをどう設定するかは単純明快です。通常、送信データレートは受信データレートと同じであり、最終的にはベースバンド部のアルゴリズムに依存します。この例では、デコード用のアルゴリズムが12.5MSPSのサンプリング・レートで動作するように設計されています。したがって、AD9361のデータレートもそれに合わせて設定します。そうすることで、デシメーションやインターポレーションの処理を追加することなく、受信信号から得たサンプルをそのままデコード用のアルゴリズムに引き渡すことができます。

AD9361のRx（受信）側アナログ・ベースバンド部では、アンチエイリアシング（折返し誤差防止）と帯域外信号の除去を行うためにローパス・フィルタを使用します。RF帯域幅を制御することで、同フィルタの帯域幅も決まります。受信信号を正しく復調するには、システムのS/N比を最大化する必要があります。これを実現するためには、平坦性と帯域外成分除去の仕様を満たしつつ、RF帯域幅をできるだけ狭くします。それにより、帯域内のノイズとスプリアス信号のレベルを最小限に抑えることができます。RF帯域幅を必要以上に広く設定すると、ノイズが増加し、A/Dコンバータ（ADC）の線形ダイナミック・レンジ性能が低下します。同様に、帯域外成分の除去性能が低下するため、ADCのSFDRが低下し、レシーバの全体的なダイナミック・レンジが低下します。つまり、RF帯域幅を最適な値に設定することが、帯域内の信号を正確に受信し、帯域外成分を除去するうえで非常に重要だということです。この例の場合、受信信号のスペクトルを観測した結果から、RF帯域幅の適切な値は4MHzであることがわかりました。

IIO System Objectを使えば、RF帯域幅の属性によってAD9361のアナログ・フィルタの設定を行うだけでなく、FIR型デジタル・フィルタを有効にすることでデコード性能を高めることができます（図5）。

```
s.writeFirData('adsb.ftr');
```

図5. libiioによりAD9361のFIRフィルタを有効にするためのコード

図5では、adsb.ftrというファイルを指定しています。このファイルには、ADIのMATLABアプリケーション「AD9361 Filter Wizard」[8]を用いて設計したFIRフィルタの係数が含まれています。ADS-B信号のスペクトル特性に基づき、データレートを12.5MSPS、通過帯域周波数を3.25MHz、阻止帯域周波数を4MHzとしてFIRフィルタを構成しました。これによって、対象帯域幅をさらに絞

り込むことができます。AD9361 Filter Wizardを使えば、汎用ローパス・フィルタの設計だけでなく、信号パスの他のステージに対する振幅と位相のイコライズも行えます。

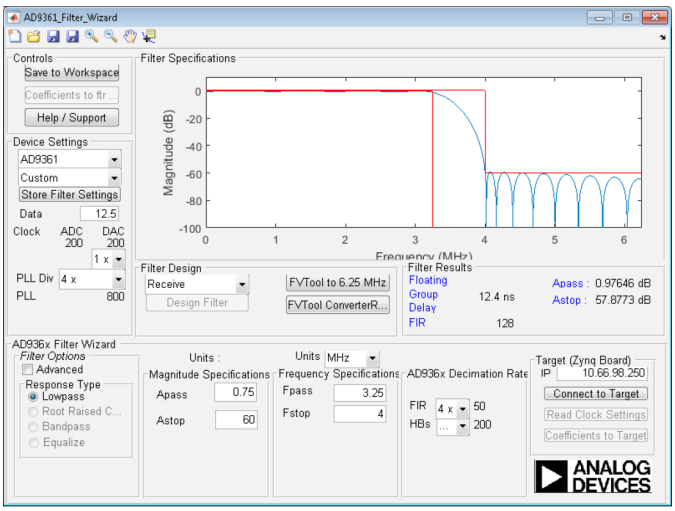


図6. MATLABベースのAD9361 Filter Wizardを用いてADS-B信号用に設計したFIRフィルタ

AD9361は汎用性と構成可能性（コンフィギュラビリティ）に優れるトランシーバICです。ゲイン制御用のモードを複数種備えているため、多様なアプリケーションに対応できます。IIO System Objectでは、Gain Mode（ゲインのモード）というパラメータにより、manual、slow_attack、hybrid、fast_attackのうちいずれかのモードを選択します。よく使われるのは、manual、slow_attack、fast_attackの各モードです。manualモードを選択した場合、ベースバンド・プロセッサによってゲインを制御することが可能です。slow_attackモードは変化の遅い信号用であり、fast_attackモードは急なオン/オフが発生する信号用です。ゲインのモードをどのように設定すべきなのかということは、受信信号の強度に大きく依存します。信号強度が非常に強いか非常に弱い場合には、manualモードかslow_attackモードを使用するとよいでしょう。それ以外の場合は、fast_attackモードを選択するべきです。ADS-B信号の場合、急激に変化が生じるため、ゲインのモードとしてはfast_attackを選択した場合に最良の結果が得られます。ADS-B信号にはプリアンブルがあり、先頭ビットを取得するためにはAGC（自動利得制御）が十分に速く反応する必要があります。このことから、fast_attackモードが適切な選択肢になるということです。信号が存在しない期間では、アタック・タイム（ゲインを低下させるまでにかかる時間）とディケイ・タイム（ゲインを増加させるまでにかかる時間）の間に差があります。ここでの目標は、ビットからビットへの遷移時間内にゲインを増加させることなく、先頭ビットとして有効な1を観測できるように、ゲインを素早く低下させることです。

最後に、TX_LO_FREQとRX_LO_FREQを設定します。両者をどのように設定するかによって、取得済みのデータを使用する方法（RFループバック）と、エア中のライブ・データを使用する方法のうちどちらでこのモデルを使用するのかを選択できます。

取得済みデータの使用法

ここでは、AD-FMCOMMS3-EBZを使用して、取得済みのADS-B信号を送受信します。信号のデータは、newModeSという名前の変数に保存します。

```
input_content{1} = (2^13).*newModeS./sqrt(2);
input_content{2} = (2^13).*newModeS./sqrt(2);
input_content{3} = (2^13).*newModeS./sqrt(2);
input_content{4} = (2^13).*newModeS./sqrt(2);
```

図7. 取得済みのADS-B信号を入力にするための定義

ここでの要件は、「TX_LO_FREQ=RX_LO_FREQ」とすることです。その値としては、AD-FMCOMMS3-EBZがサポートする任意の局部発振周波数の値を設定できます。取得済みのデータを使う場合、有効なADS-Bデータが多数存在するので、ハードウェアの設定が適切か否かを十分に検証することができます。

ライブ・データの使用法

次に、AD-FMCOMMS3-EBZによって送信された信号ではなく、エア中のADS-B信号をリアルタイムで受信する方法を説明します。まず、ADS-Bの仕様によると、送信信号の中心周波数は1090MHzです。このことから、この例における要件は次のようになります。

- RX_LO_FREQの値は1090MHz。干渉を避けるために、TX_LO_FREQの値としては1090MHzから十分に離れた周波数を選択する
- レシーバ側では、1090MHz帯をカバーする適切なアンテナを使用する（「ADS-B Double 1/2 Wave Mobile Antenna」[9]など）。チューニングが適切でなかったり、性能の低いアンテナを使用したりすると、航空レーダーの探知範囲が狭くなる

すべてを正しく設定したら、次の簡単なコマンドによってMATLABモデルを実行することができます。

```
[rssi1,rssi2]=ad9361_ModeS('ip','data source',channel);
```

ここで、ipはFPGAを搭載するボードのIPアドレス、data sourceは受信信号に対応するデータ・ソースです。現在、このモデルは「pre-captured」（取得済み）と「live」（ライブ）の両データ・ソースをサポートしています。channelは、AD-FMCOMMS3-EBZのチャンネル1とチャンネル2のうちどちらで信号を受信するのかを指定するためのものです。

例えば、次のコマンドであれば、チャンネル2で、取得済みのデータを受信するという意味になります。

```
[rssi1,rssi2]=ad9361_ModeS('192.168.10.2','pre-captured',2);
```

シミュレーションが終了すると、両方のチャンネルのRSSIの値と共に、図8に示した結果が表示されます。

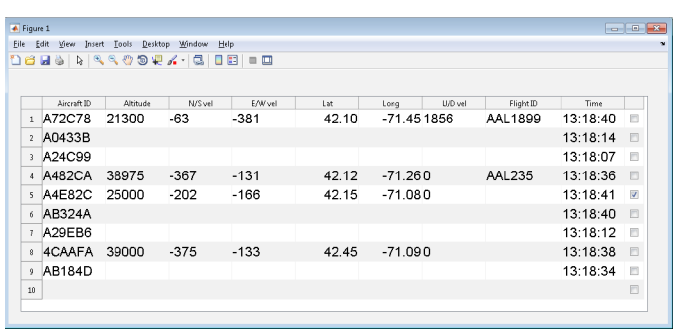


図8. シミュレーションの終了時に表示される結果

図8の表には、シミュレーション中に取得した航空機の情報が表示されています。適切なアンテナを使用した場合、このモデルによってAD-FMCOMMS3-EBZから80マイル（約130km）の範囲内にある航空機からの信号を取得/デコードすることができます。モードSのメッセージは2種類（56µsまたは112µs）あり、含まれる情報の量に違いがあります。

このモデルを実際のADS-B信号に対して試用する場合、正しいデコードの実現に向けては信号強度が非常に重要な要素になります。そのことから、アンテナは航空機に対して良好な見通し線が得られる位置に配置してください。受信信号強度は、両方のチャンネルのRSSIの値によって確認できます。例えば、チャンネル2で信号を受信する場合、チャンネル2のRSSIがチャンネル1のRSSIよりもかなり高くなっていなければなりません。有用なデータが存在するか否かは、スペクトル・アナライザで観測することによって確認できます。

RF信号の品質

本稿の例のようなシステムでは、任意のRF信号に対する品質基準が必要になります。例えば、QPSK（4位相偏移変調）のような信号ではエラー・ベクトル振幅（EVM）が重要な意味を持ちます。ADS-B信号の場合、図8に示したような正しいメッセージのスライサの出力を確認するだけでは十分ではありません。どちらの設定の方が優れているかといったことを判断できるように、ADS-BとPPM（パルス位置変調）の品質を定める基準が必要になります。

ModeS_BitDecode4.mという関数には、そのような基準として使用できるdiffValsという変数が定義されています。これは112×1のベクトル変数であり、1つのモードSのメッセージに含まれるデコード後の各ビットが、閾値と比べてどれだけの差を持つかを表します。つまり、デコード後の各ビットが、判定に使う境界値に対してどれだけのマージンを持っているのかを表すということです。当然のことながら、各ビットのマージンが大きいほど、デコード結果の信頼性は高くなります。逆に、マージンが小さい場合には、境界付近で判定が行われたということを意味します。この場合、デコード結果は誤っている可能性が高いと言えます。

図9、図10は、FIRフィルタを適用した場合と適用しなかった場合のそれぞれについて、ADS-B用のレシーバから得られたdiffValsの値を示したものです。Y軸方向に着目すると、diffValsの値は、高いか低いか、あるいは平均レベルであるかにかかわらず、FIRフィルタを適用した場合の方が大きいことがわかります。一方、FIRフィルタを適用しない場合、複数のビットでdiffValsの値が0に近づきます。つまり、デコード結果が誤っている可能性があるということです。これらの結果から、適切なFIRフィルタを適用することが、信号のデコードにおける品質の向上につながるということを確認できます。

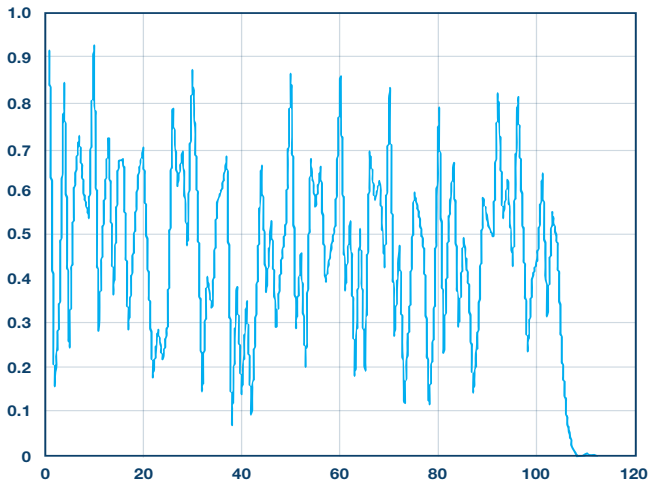


図9. FIRフィルタを適用した場合にADS-B用レシーバから得られたdiffValsの値

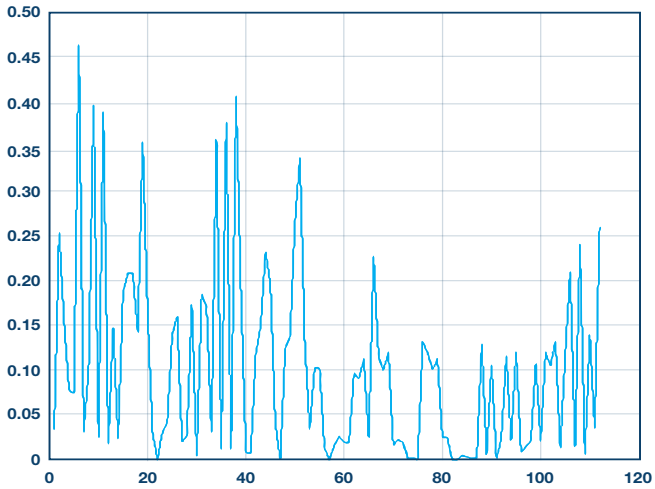
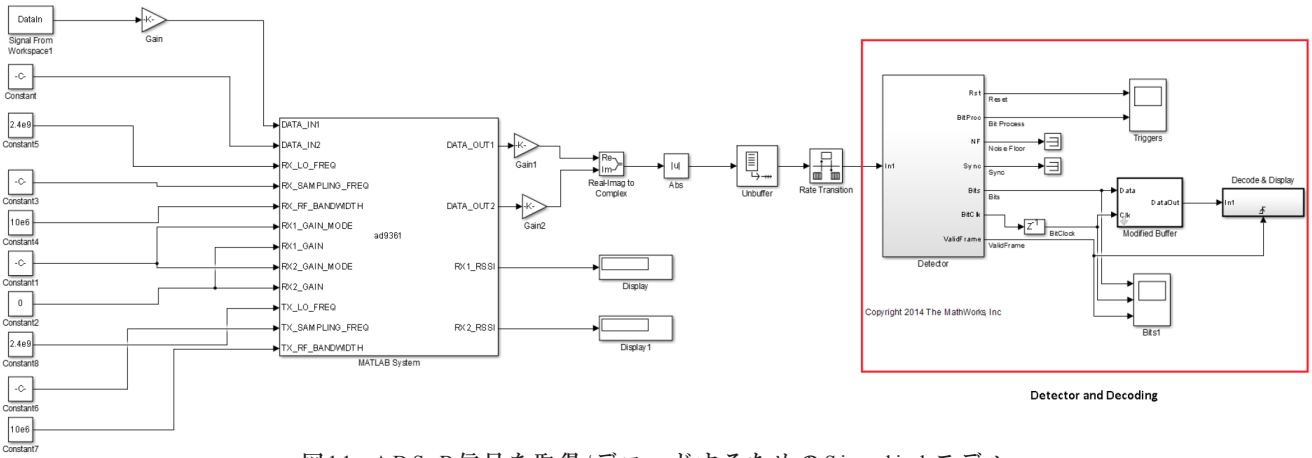


図10. FIRフィルタを適用しない場合にADS-B用レシーバから得られたdiffValsの値



Detector and Decoding

図11. ADS-B信号を取得/デコードするためのSimulinkモデル

ADIは、MATLABによりIIO System Objectを用いて記述したADS-B用のアルゴリズムを、GitHubリポジトリで提供しています[10]。

IIO System Objectを用い、Simulinkベースの ADS-Bアルゴリズムを検証

Simulinkモデルは、本シリーズのPart 2[2]で示したモデルに基づいて構築します。検出とデコードの部分は元のモデルのままで、信号の受信とHIL（Hardware in the Loop）シミュレーションを行うためにSimulinkのIIO System Objectを追加しました。

元のモデルでは、サンプル時間は1、フレーム・サイズも1でした。しかしSimulinkのIIO System Objectは、多数のサンプルを蓄積してから処理するバッファ・モードで動作します。この例では、元のモデルをSystem Objectに対応させるために、両者の間に2つのブロックを追加しました。フレーム・サイズを1にするためのUnbufferと、サンプル時間を1にするためのRate Transitionです。こうすることで、元のモデルを変更せずにそのまま使うことができます。

SimulinkのIIO System Objectは、MATLABの場合と同様に設定します。つまり、System Objectを生成し、IPアドレス、デバイスの名前、入出力チャンネルの番号とサイズを定義します。

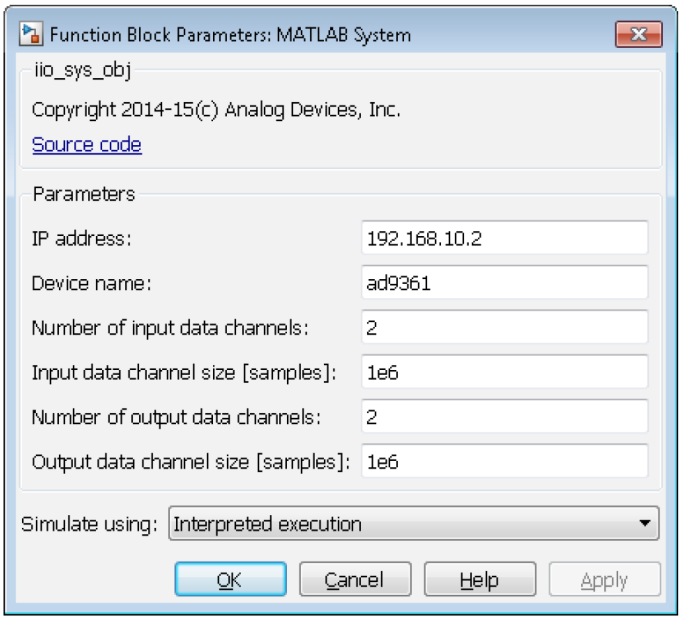


図12. SimulinkのIIO System Object.

IIO System Objectに対応するSimulinkブロックの入出力ポートについては、そのオブジェクトのブロックのプロパティ・ダイアログと、ターゲットであるFMCOMMSプラットフォーム用の構成ファイルで定義します。この入出力ポートは、データ用のポートと制御用のポートに分類されます。データ用のポートは、フレーム単位の処理モードにおいて、ターゲットとなるシステムとの間で連続してデータを受信/送信するためのバッファとして使用します。一方、制御用のポートは、ターゲットのシステムが備えるさまざまなパラメータの構成と監視に使用します。データ・ポートの番号とサイズは、ブロックの構成ダイアログで設定するのに対し、制御ポートは構成ファイルで設定します。AD9361の属性を設定する際に検討すべき事柄は、MATLABモデルの場合と同じです。MATLABモデルで使ったすべての理論と手法を、同様に適用できます。

このSimulinkモデルでは、TX_LO_FREQとRX_LO_FREQをどのように設定するかによって、取得済みのデータ（DataIn）を使用するモードとライブ・データを使用するモードのうちどちらで実行するのかが決まります。例えば、取得済みのデータを使用した場合、シミュレーションの終了時には図13のような結果がコマンド・ウィンドウに表示されます。

```
Aircraft ID 400927      Long Message CRC: 8D4009276OC38037389C0EF0029C
Aircraft ID 400927 is at altitude 39000
Aircraft ID 400927 is at latitude 42 19 24.8, longitude -71 8 33.3
Aircraft ID 400927      Long Message CRC: 8D4009279944E7B320048CDB40FA
Aircraft ID 400927 is traveling at 468.363107 knots
Direction West at 230.000000 knots, direction South at 408.000000 knots
Aircraft ID 400927 is going Up at 0.000000 feet/min
```

図13. 取得済みのデータを使用した場合に、シミュレーションの終了時にコマンド・ウィンドウに表示される結果

MATLABモデルの場合、結果は表として示されましたが、このSimulinkモデルではテキスト形式で表示されます。ADIは、IIO System Objectを用いて構築したADS-BのSimulinkモデルを、GitHubリポジトリで提供しています[11]。

まとめ

本稿では、ADIが提供するlibiioによって実現したHILシミュレーションについて解説しました。このlibiioというインフラを用いることで、MATLABとSimulinkによるADS-B信号の検出/デコード用アルゴリズムを、実際の信号と実際のハードウェアを使って検証することができます。属性の設定は、アプリケーションと信号に大きく依存するため、ある信号に対して適切な設定が、別の信号でも有効であるとは限りません。アルゴリズムの検証は、SDRシステムのアナログ・フロントエンドとデジタル・ブロックが、対象とする信号とアルゴリズムに合わせて適切にチューニングされていることを確認するために不可欠な作業です。また、検証は、アルゴリズムが十分に堅牢であり、多様な環境条件の下で取得される実際のデータに対して期待どおりに動作することを保証するためにも必須の作業です。アルゴリズムの検証が完了すれば、次のステップに進むことができます。つまり、MathWorks社の自動コード生成ツールによってアルゴリズムをHDLのコードやC言語のコードに変換し、それらをSDRシステムのプログラマブル・ロジックとプロセッサに適用するということです。本稿の続編となるPart 4では、まずコードを生成してハードウェアに配備（デプロイ）する方法について説明します。続いて、空港で実際のADS-B信号に対してこのプラットフォームを動作させた結果を示します。このPart 4によって、SDRシステムのプロトタイプ開発から製造までのすべてのステップが完結します。

References

[1] 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

[2] Mike Donovan、Andrei Cozma、Di Pu 「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」Analog Dialogue 49-10

[3] Analog Devices「IIO System Object」

[4] MathWorks「What Are System Objects?」

[5] Analog Devices「Mathworks_tools」GitHub repository.

[6] Analog Devices「AD-FMCOMMS3-EBZ User Guide」

[7] ZedBoard

[8] Analog Devices「MATLAB AD9361 Filter Design Wizard」

[9] ADS-B Double 1/2 Wave Mobile Antenna

[10] MATLAB ADS-B Algorithm Using The IIO System Object Source Code

[11] Simulink ADS-B Model Using The IIO System Object Source Code

謝辞

本稿で使用したMATLAB/SimulinkによるADS-B信号の検出/デコード用アルゴリズムの開発にご協力いただいたMathWorks社のMike Donovan氏に感謝いたします。

著者：

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

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