LabVIEW & LabVIEW FPGA モジュール入門

my recommendations on research & development tools.

堀田 育志†

キーワード：LabVIEW，グラフィカルプログラミング，FPGA，並列処理，確率共振，非線形システム

1. ま え が き

本誌の読者の多くは、研究・開発に関わっている方々でしょう。研究・開発では、取り扱う問題を設定し、実験またはシミュレーションからデータを収集し、データの解析と考察によって問題解決の糸口を探っていくというのが、どの分野でも共通するプロセスではないでしょうか。このプロセスにおいて時間は、できるだけ問題設定や考察に使いたいところですが、実際にはシステムの構築・改良とデータの収集・解析に費やさなければなりません。ツールは、このようなシステム改良やデータ収集といった作業の効率化と時間短縮に大変有効なものです。

現在では、さまざまなツールが世に出回っており、各使用目的において便利なものを選ぶことができます。しかし、いざツールを導入しようとする、数あるものの中からどれを使えばよいかわからないところ、筆者も、自身の研究においてどのツールを用いればよいかわからないと悩ませていました。

では、研究・開発の現場において、ツールに求められるものとは何でしょうか。まず、使用するツールに関して、より汎用性の高いツールを一つ選びたいと思うことでしょう。しかしツールを選んだとしても、実際にはなかなかじっくりと使い方を学ぶ時間がとれません。さらに、いざツールが使えるようになって、研究ではシステムを頻繁に改良する必要があり、それに余計な時間を取られてしまいます。そこで筆者が考えたのが、① 汎用性と自由度が高いこと、② ツールの使い方を覚えるのにそれ程時間をかけなくてよいこと、③ 短時間で実験システムの構築と改良ができること、の三つです。

今回は、これらのニーズを満たしてくれるツールとしてLabVIEWとLabVIEW FPGAモジュールを紹介したいと思います。

2. ツールの概要

2.1 LabVIEWの概要

LabVIEW (Laboratory Virtual Instrumentation

Engineering Workbench)は、NI(National Instruments社)が提供するグラフィカルプログラミング言語であり、データ収録、信号処理、画像処理、計測器制御、自動テストシステムの構築などの幅広い分野で利用されています。温度計のデータ収録から、世界最大の加速器CERNの大型ハドロン衝突型加速器の制御にまで利用されていることから¹⁾、その使用範囲の広さと信頼性の高さが伺えます。また最近では、LabVIEWは国内外の企業の開発現場や大学の研究・教育現場で使用するプログラミングツールとして採用されています¹⁾。その理由は、LabVIEWのグラフィカルなプログラム開発環境にあり、これによって誰もが短時間で必要とするシステムを構築できるようになるからです。

では、LabVIEWのグラフィカルプログラミンとはどういうものか見ていきましょう。LabVIEWでは、実行可能なプログラムユニットをVI (Virtual Instruments) と呼びます。VIは、図1に示すように、ユーザインタフェースを構成するフロントパネルとプログラムを記述するブロックダイアグラムで構成されます。LabVIEWプログラミングでは、関数パレットから使用する関数のアイコンをブロックダイアグラムに配置し、それらアイコンの入出力端子間をワイヤーで配線することで、データフローを作っていきます。これによってフローチャートを書くように直感的にプログラミングができる訳です。また、VIは再利用性やオブジェクト性に優れています。作成したVIを、他のVIからサブVIとして呼び出して再利用したり、メインVIの一部をサブVIに置き換えたりすることも簡単にできます。

LabVIEWは、プログラム開発環境としても優れています。システムの開発では、プロジェクトエクスプローラというツールによって複数のVIやハードウェア構成をプロジェクト単位でまとめて管理することができます(図1)。また、サンプルプログラム、関数やサンプルプログラムの検索機能、ヘルプ機能、追加モジュール等が充実しています。多くの場合は、すでに準備されているサンプルプログラムを加工するだけで充分実用的なプログラムを作成することができます。さらに、インターネット上にディスカッションフォーラムが立ち上がっており、フォーラムを通してLabVIEWに関する情報収集を行うことができます²⁾。

2.2 グラフィカルであることのメリット

LabVIEWは、プログラミングの初心者に限らず、誰も

† 独立行政法人理化学研究所 基幹研究所

"Introduction to LabVIEW and LabVIEW FPGA Module" by Yasushi Hotta (Advanced Science Institute, the Institute of Physical and Chemical Research (RIKEN), Saitama)

図1 LabVIEWの開発画面。VIはサンプルプログラム1の正弦波発生器のもの

が入門し易いというメリットがあります。これは、LabVIEWではテキストベースの言語のように文法を覚える必要がないため、このメリットにより誰もが直ぐにプログラミングを行えるようになります。

筆者が考えるもう一つメリットは、プログラムの構造を直感的に理解できることです。これは、プログラムをブロックダイアグラム平面上に配置することで、プログラムの階層や機能ブロックを視覚的に分けて考えることができるためです。また、テキストベース言語では、プログラムフローが上から下に向かうだけなのに対し、LabVIEWでは上下左右さまざまな方向のフローをグラフィカルに表現できます。これによって、使用する機能ブロックに直ぐにたどり着くことができたり、プログラムの並列処理や主従関係を視覚的に表現できたりします。このように、構造やフローを視覚的に表現できることで、プログラムの規模が大きくなっても容易に全体像を把握することができます。

2.3 LabVIEW FPGAモジュールの概要

FPGA (Field-Programmable Gate Array) は、再構成可能なプログラマブルロジックデバイスのことです。ユーザが定義したカスタムの論理ロジックを、現場レベルでハードウェア化できるデバイスです。一般的には、アナログI/O、トリガリング、クロック発生、カウンタ、タイミング同期、PWM制御、PID制御などのカスタムハードウェア構築に利用されています。

ハードウェアベースのシステムは、ソフトウェアベースのものと比較して、高速性と並列性に優れています。また、

処理にPCのリソースを使用しないことから、システムの安定性が向上します。FPGAは、ハードウェアベースシステムをより身近に利用できるようにしたデバイスといえます。しかし、FPGAを利用するためには専門のハードウェア言語の使い方を新たに覚える必要があります。これが専門家以外の利用者にとっての障壁となります。

そこで登場するのが、LabVIEW FPGAモジュールです。これは、LabVIEWのグラフィカルプログラミング環境を用いて、FPGAの論理ロジックの定義が行えるモジュールで、これによってハードウェア言語が使えなくても、FPGAを利用できるようになります。LabVIEW FPGAモジュールでは、LabVIEW上で論理ロジックの定義を行い、それをコンパイルしたビットファイルをFPGAデバイスにダウンロードして使用します。もちろんFPGA上のビットファイルは書換えが可能であり、一度デバイスに実装した後でも、別のビットファイルを再ダウンロードすることで機能の変更が行えます。また、FPGAのプログラミングにはいくつかのハードウェア言語特有の制約がありますが、LabVIEW FPGAモジュールでは、利用できる関数ははじめから制限されており、自ずと制約を守ったコードの作成が可能になっています。

3. LabVIEWプログラミング

それでは、実際にLabVIEWを使用してみましょう。NIのウェブサイトから、最新のLabVIEW 2010の評価版がダウンロードできます³⁾。これは、すべての機能が30日間無

料で利用できるものです。一方、LabVIEW FPGAモジュールですが、実際に動作させるためには、FPGAが搭載されたハードウェアが必要となるため、今回は説明だけに留めます。評価版のインストールが終了すると、LabVIEW 2010を起動します。

VI作成は、フロントパネルの作成、ブロックダイアグラムへの関数やサブVIの配置とそれらの配線、コンパイルと実行、そしてデバッグという流れで行います。

3.1 サンプルプログラムの説明

筆者は、確率共振と呼ばれる現象を用い、ノイズで音声信号をリアルタイムに検出するシステムの開発を行っています⁴⁾。確率共振とは、ニューロンのような閾値をもつ非線形システムにノイズが加えられたとき、そのシステムの信号伝達性能が向上するという現象です。この現象は、われわれ人間を含めたさまざまな生体システムに見られます。皆さんも、静かな図書館よりも少し騒がしい喫茶店の方が、仕事がかどるといった経験がありませんか。これも一種の確率共振(?)かもしれません。今回は、この現象によって閾値システムに入力された閾値以下の微弱信号をノイズによって復元するプログラムを作ってみましょう。

確率共振現象に必要な要素は、微弱信号源、非線形閾値判別器、そしてノイズ(ホワイトノイズ)源の三つになります。そこで、これら三つの機能をもつVIを作成し、統合することにします。さらに信号を復元するためには、閾値判別器を並列に並べ、それらの平均を取る必要がありますので、その作業を行うためのVIも作成します。

これからご紹介するサンプルプログラムのVIIは、学会のウェブサイトからダウンロードすることができます。皆さんサンプルを参考にして実際にVIを作成してみてください。

3.2 正弦波発生器

本サンプルプログラムでは、微弱信号として正弦波を使用します。まずは、簡単な正弦波発生器を作成してみましょう。もちろん、正弦波発生の関数はLabVIEWにデフォルトで用意されていますが、ここでは練習も含めて後のプログラムで使いやすいVIを作成します。

先ほどLabVIEWの概要説明に使った図1が、正弦波発生器VIのフロントパネルとブロックダイアグラムの一例になっています。フロントパネルには、制御器として“振幅”、“位相”、“周波数”を制御する三つのコントロールと“停止”ボタン、そして結果を表示するための“波形チャート”を配置しています。これらは、フロントパネルの制御器パレットからドロップアンドドラッグするだけで作成することができます。

正弦波は、Whileループのなかで“振幅” $\times \sin(2\pi \times \text{“周波数”} \times \text{“時間”} + \text{“位相”})$ を計算することで、発生させます。ブロックダイアグラムの関数パレットからWhileループ、正弦値関数、加算、積算、正弦値のアイコンをドロップアンドドラッグして配置し、図1のブロックダイアグラムを

参考にして、上の式を満たすように配線していきます。“時間”は1回のWhileループを10msとして、反復端子を利用して増加させます。最後に計算された正弦値を波形チャートの入力に配線します。以上でプログラミングは終了です。これだけで、正弦波発生器を作成することができました。このVIIは、サンプルプログラム1に収録されています。試しに、“振幅”=10、“位相”=0 deg、“周波数”=5HzとしてVIを実行してみてください。VIを実行するには、ボタンメニュー左端の矢印ボタンをクリックします。波形チャートに5Hzの正弦波が出力されるのが確認できると思います。

続いて、ブロックダイアグラムウィンドで正弦波発生部分(図1の破線枠内)を選択し、“メニューの編集>選択範囲をサブVIに変換”を選んで選択範囲をサブVI化してみましょう。このとき、配線のコネクタも自動的に作成されます。これで、プログラムの構造が理解し易くなり、またサブVI化した部分は再利用性が向上します。サンプルプログラムでは、このサブVIを“正弦波発生器.vi”として収録しています。また、フロントパネルの制御器をダイヤル式ボリュームに変更して、操作性を良くしたVIをサンプルプログラム2に収録しました。

3.3 ノイズ源を内蔵した非線形閾値判別器

ここでは、非線形閾値判別器としてヒステリシスコンパレータのVIを作成します。このVIのフロントパネルとブロックダイアグラムの例を図2に示します。また、このVIをサンプルプログラム3に収録しました。

フロントパネルの制御器は、“出力レベル”、“閾値”、“入力信号”、“ノイズレベル”、表示器は“波形チャート”です。入力するパラメータが多くなってきたので、ここではクラスタ変数を使っています。クラスタ変数を使うことで、多変数の受け渡しが一本の配線できるようになります。

続いて、ブロックダイアグラムウィンドに移動して、プログラミングを行います。コードは、先ほど正弦波発生器より複雑ですが、同様に関数アイコンを配置して配線していきます。閾値判別器の状態を表すのにブール変数とシフトレジスタを使っていますが、これはWhileループの外側に“False”定数を作成してWhileループに接続し、接続したコネクタを右クリックして“シフトレジスタと置換”を選択することで使用できます。コードができあがると、ノイズ源と閾値判別器部分をサブVI化しましょう。サンプルプログラムでは、このサブVIを“閾値判別器.vi”としました。また、フロントパネルの制御器をダイヤル式ボリュームに変更したVIをサンプルプログラム4に収録しました。

3.4 閾値判別器ユニットの並列化

ここでは、Forループを使って閾値判別器ユニットの並列化を行います。ブロックダイアグラムの例を図3に示し、VIをサンプルプログラム5に収録しました。作成手順ですが、先ほどサブVI化した“閾値判別器.vi”のアイコンをFor

図2 ノイズ源を内蔵した非線形閾値判別器のフロントパネルとブロックダイアグラム

図3 閾値判別器ユニットを並列化したVIのブロックダイアグラム

ループで囲みます。閾値判別器の状態を表すブール変数が自動的に配列化されますが、このままでは、配列のサイズが未定義のため、サブVIにうまく値が引き渡されません。そこで、フロントパネルに並列数を入力するための“並列ユニット数”制御器を追加し、Forループの反復回数“N”に配線します。続いて、Whileループの外側に配列サイズを定義する関数を配置して“並列ユニット数”の値と“False”定数を配線します。配列をForループに配線する場合、自動指標付けを有効にすることでForループが配列の指標要素を自動的に読取ってループ処理することができます。

このプログラムでは、実行中に“並列ユニット数”制御器の値を変えてもForループの回数を変更することはできません。これは、制御器がWhileループの外側にあるためです。VI実行中に値を変更した場合は、一度プログラムを停止し、再実行したときにその値が反映されます。

“並列ユニット数”を増やしてこのVIを実行すると、入力信号の各時間における一点のデータに対して、“閾値判別器.vi”がForループで反復して実行されます。これで、閾値判別器が並列に並べられているのと同じ処理になります。反復実行された結果は、このサブVIの二つの出力である“波形チャート”と“ブール2”の配列として出力されます。これらのうち、“波形チャート”の値を格納した配列の平均値を求めることで波形再現ができるようになります。

3.5 信号復元確率共振システム

最後は、正弦波発生器と並列化されたノイズ源付閾値判別器ユニットを統合したVIを作成します。図4に、それぞれを統合したVIのフロントパネルとブロックダイアグラムを示します。このVIIは、サンプルプログラム6に収録しました。

図4のブロックダイアグラムの作成手順ですが、図3のブロックダイアグラムに“正弦波発生器.vi”を呼び出し、その出力と“入力信号”制御器の値を加算してクラス変数の“入力信号”に配線します。これで、正弦波発生器の出力が入力されるようになります。以上で、ノイズで信号を復元する確率共振システムの完成です。

試みに、正弦波発生器の“振幅”=2、“位相”=0°、“周波数”=5Hzとし、閾値判別器の“出力レベル”=2、“閾値”=5、“入力信号”=0、“ノイズレベル”=0、そして“並列ユニット数”=1として、このVIを実行します。この状態では、波形チャートの入力信号(白線)には正弦波が表示されますが、出力信号(赤線)には、入力正弦波の振幅が閾値を超えないため一定値しか表示されません。ここで“ノイズレベル”のボリュームを上げていくと、あるノイズレベル以上から5Hzの矩形波が現れるようになります。この現象は、入力の正弦波がノイズと確率的に共振して閾値を超えることで起こります。この現象が“確率共振”と呼ばれるので

図4 信号復元確率共振システムのフロントパネル（実行中）とブロックダイアグラム

す。続いて、一度プログラムを停止し、並列数を増やしてみましょう。“並列ユニット数”を100にしてVIを再実行します。今度は、矩形波ではなく、正弦波が波形チャートの出力信号に現れます。このように閾値判別器の並列数が増えると波形の再現性が向上していきます。これは、並列数が増えることでシステム全体の取り得る状態の数が増え、正弦波の形を閾値を超える閾値判別器の数で表現できるようになるからです。

3.6 FPGAを利用するメリット

先ほどのプログラムでは、並列数を増やすほど処理が重くなっていくことと思います。ソフトウェアベースのシステムでは、プログラムは逐次処理され、処理数が増えると当然遅れがでます。音声信号を確率共振処理で復元する場合、可聴域の20Hz～20kHzの信号に対して、100個程度の閾値判別器を並列に動作する必要があります。これをPCで処理するには、マルチスレッドのCPUを搭載したもので大変なことがわかります。

しかし、FPGAを利用したハードウェアシステムでは、並列化したブロックダイアグラムがFPGAの内部クロックで完全に並列に動作します。つまり、FPGAのゲート容量の許す限り閾値判別器を並列に並べると、そのすべてを同時並列に動作させることができます。私の研究では、FPGAを用いたことでリアルタイムの音声再現システムが実現できたという訳です。

4. む す び

本稿では、LabVIEWおよびLabVIEW FPGAモジュールを紹介しました。またサンプルプログラムを通して、

LabVIEWグラフィカルプログラミングの簡便性を取り上げて説明しましたが、このツールによって、短時間でシステムの構築と改良が行えることがわかっていただけたでしょうか。

筆者は、LabVIEWとLabVIEW FPGAモジュールを導入する前は電子回路を用いて研究を行っていました。電子回路で今回のサンプルプログラムのようなシステムを構築するには、当然ながら並列数分の回路を作製する必要があり、また、システムに変更が生じる度にすべての回路を作り直さなければなりません。半田ごて片手に、四つ回路を作製・改良するだけで丸一日を費やしていました。一方、FPGAを導入してからは、1時間もあれば100個の閾値判別器の改良が終了します。もちろんLabVIEWのグラフィカルプログラミング環境が大いに役立っていることは言うまでもありません。簡単に実験システムの構築・改良ができるようになったことで、今後ますます試してみたい研究のアイデアが膨らんでいきます。（2010年11月2日受付）

〔文 献〕

- 1) <http://sine.ni.com/cs/app/main/>
- 2) <http://zone.ni.com/dzhp/app/main/>
- 3) <http://www.ni.com/trylabview/ja/>
- 4) <http://sine.ni.com/cs/app/doc/p/id/cs-12408/>

堀田 育志 2004年、大阪大学大学院理学研究科博士後期課程修了。現在、(独)理化学研究所基幹研究所研究員。2009年、日本ナショナルインスツルメンツ・アプリケーションコンテスト2009 一般部門 最優秀賞受賞。博士(理学)。