MISH TECH JOURNAL（ミッシュ・テックジャーナル）は、最新の情報をいち早くお届けする技術情報誌です。

Powered by MISH

MISH TECH JOURNAL

2025 Summer

Vol.18

https://www.mish.co.jp

CONTENTS

‘25夏号特集

Direct Digital Synthesizer

AWGを使用したマルチトーン信号生成

企画・編集・発行/株式会社ミッシュインターナショナル
〒190-0004 東京都立川市柏町4-56-1 グローバルビル
TEL：042-538-7650 Email：sales@mish.co.jp

SPECIAL FEATURE

AWGを使用したマルチトーン信号生成

はじめに

ここではSpectrum Instrumentation製AWGボードを使用したマルチトーン信号の生成について詳しく説明します。

DDS は、Direct Digital Synthesizerの略で複数のDDSコアを使用して複数の異なる周波数の正弦波を生成することができます。

従来のAWGモード

DDSモードのさまざまな側面を詳しく説明する前に、まず従来の任意波形発生器 (AWG) モードを詳しく見てみます。このモードでは、ユーザーは出力波形を構成する各サンプルデータを事前に作成し、それをオンボードメモリに送信します。その後、このデータを標準再生モード、FIFOモード、シーケンスモードなどの選択した再生モードに応じて、ワンショット、ループ再生、または連続ストリームでアナログ出力に再生できます。原則として、このAWGモードではあらゆる種類の出力波形が可能なことから、多くのアプリケーションで柔軟なソリューションになります。

高いサンプリング周波数で長いAWGデータシーケンスを保存するには、大容量のオンボードメモリが必要です。そのため、たとえば最大サンプリング周波数が1.25GHzのAWGカードのM4i.66xxシリーズには、4GByte (2Gサンプル) のメモリが搭載されています。この大容量メモリにより、1秒を超えるシーケンスを保存できます。

AWGカードに連続的なデータストリームを転送するために、すべてのSpectrum製AWGカードには高速PCIeインターフェイスがあり、1秒あたり数ギガバイトのデータを簡単に転送できます。CPUを使用して必要なAWGデータを継続的に計算するのは非常に困難なため、Spectrum製カードはSCAPPオプションを使用して、NVIDIAの最先端のCUDA GPUアクセラレータカードからの転送もサポートしています。

ストリーミングFIFOモードでは、オンボードメモリが送信データのFIFOバッファとして使用されます。これにより、データを生成するユーザーアプリケーションが一瞬停止するイベントを補正できるため、ストリーミングの信頼性が非常に高くなります。ただし、

バッファリングによって遅延も発生するため、クローズドフィードバックループや一般的な制御システムなどの特定のアプリケーションでは不利になる場合があります。

応答時間を短縮し、データ転送を削減するために、AWGにはシーケンスモードがあります。このモードでは、事前に保存されたさまざまなシーケンスを構成可能な順序でループし、外部トリガー入力でシーケンス間の切り替えをトリガーできます。

ただし、シーケンスモードでは依然として小さなFIFOバッファが使用されます。したがって、外部入力でシーケンスの変更をトリガーする場合、FIFOバッファ内の古いデータを出力に転送する必要があり、FIFOバッファ内の古いデータの量が変化すると、この特定のモードとアプリケーションでの動的動作に不確実性が生じる可能性もあります。

DDSモード

正弦波信号のみが必要な場合はDDSモードを使用できます。ダイレクトデジタル合成 (DDS) を使用すると、必要な周波数 $\boldsymbol{fi}$、振幅 $\boldsymbol{Ai}$、初期位相 $\boldsymbol{\varphi i}$ を一度設定するだけで、AWG カードのハードウェアで連続正弦波を生成できます。実行時には、動的な動作を実現するために変更のみを送信する必要があります。

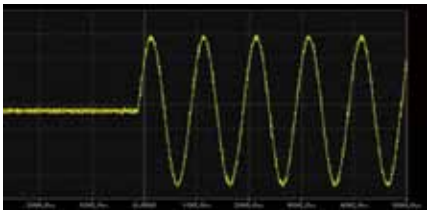
■ DDSコアとは？

計算式で表すと、DDSコア出力 $\boldsymbol{uCore,i}(t)$ は、サンプリング周波数 $\boldsymbol{fs}$ でサンプル $\boldsymbol{z}$ ごとに次の計算を実行します:

$$u_{Core,i}(t) = \sin(2\pi \cdot f_i \cdot t + \varphi_i) \cdot A_i$$
with
$$t = \frac{z}{f_s} \text{ and } z = 0, 1, 2, 3 \cdots$$

Example 1をご確認ください。

DDSコアを使用して、振幅90%の100MHz連続正弦波を出力する例です。このようにコマンドを送信するだけでDDSコアから任意の正弦波を出力することができます。



DDSコアで生成したアナログ信号出力波形

```
spcm_dwSetParam_d64(hCard, SPC_DDS_CORE0_AMP, 0.9);
spcm_dwSetParam_d64(hCard, SPC_DDS_CORE0_FREQ, 100.0e6);
spcm_dwSetParam_i32(hCard, SPC_DDS_CMD, SPCM_DDS_CMD_EXEC_AT_TRG);
```

Example 1: 100MHzでDACフルスケール範囲の90%の振幅を持つ連続正弦波を生成するだけの場合は、DDSモードを使用して3つの特定のコマンドのみをAWGカードに送信します

■ DDSモードの利点

DDSは、サンプルデータの計算タスクをコンピュータではなくAWGカード上のFPGAで実行します。これは時間とともに変化する閉じたフィードバックループや連続波形データに非常に有利です。カードに送信する必要があるのは設定値のみなので、転送する必要があるのは小さなデータのみであり、バッファ長を小さく保って応答遅延を減らすことができます。データはFPGA内でオンザフライで生成できるため、信号の位相を連続的に維持しながら、外部トリガーイベントに対して一定の遅延で反応できます。

さらに、さまざまな正弦波信号を生成するために必要な計算は、多くのユーザーにとって非常に難しくエラーが発生する場合があります。DDSは、わかりやすい高レベルのコマンドインターフェイスを提供することでこの複雑さを軽減し、プログラミングを容易にします。

■ DDSモードの欠点

DDSモードは周期的な正弦波信号を対象としており、他の信号形式を近似することもできますが、一般に正弦波以外の波形が必要な用途ではAWGモードの利用が推奨されます。

DDSモードには、非同期トリガー信号に対して6.6nsのジッターを伴う固定のタイミング遅延があります。対照的に、AWGモードには1.25GS/sで800psなどの単一サンプルのジッターしかありません。そのため、AWGモードと比較すると、DDSモードのトリガーから出力までのジッターは、1.25GS/sフルレートで8倍大きくなります。

マルチトーン信号生成

DDSモジュールは、1つの基本DDSコアの代わりに、最大N個のDDSコア1で構成され、その出力はすべて加算されて1つ以上のアナログチャネルに出力されます（Figure1参照）。したがって、いわゆるマルチトーンまたはマルチキャリア信号を生成できます。これは、たとえば量子研究の多くのアプリケーションに不可欠です。

```
spcm_dwSetParam_d64(hCard, SPC_DDS_CORE0_AMP, 0.05);
spcm_dwSetParam_d64(hCard, SPC_DDS_CORE0_FREQ, 100.0e6);
spcm_dwSetParam_d64(hCard, SPC_DDS_CORE1_AMP, 0.025);
spcm_dwSetParam_d64(hCard, SPC_DDS_CORE1_FREQ, 120.0e6);
spcm_dwSetParam_d64(hCard, SPC_DDS_CORE2_AMP, 0.05);
spcm_dwSetParam_d64(hCard, SPC_DDS_CORE2_FREQ, 140.0e6);
spcm_dwSetParam_d64(hCard, SPC_DDS_CORE3_AMP, 0.05);
spcm_dwSetParam_d64(hCard, SPC_DDS_CORE3_FREQ, 160.0e6);
//... and so on
spcm_dwSetParam_i32(hCard, SPC_DDS_CMD, SPCM_DDS_CMD_EXEC_AT_TRG);
```

Example 2: 115MHzから90MHzまでの20個の周波数を生成するには、各DDSコアパラメータを次々にプログラムするだけです。出力範囲の100%を超えないように、コア1を除くすべてのコアで出力振幅を5%に設定します。コア1では、異なる振幅設定の効果を示すために2.5%を選択しました。そのため、2番目のキャリアの振幅は、信号のスペクトルでわかるように、他のキャリアよりも少し低くなります（Figure1参照）

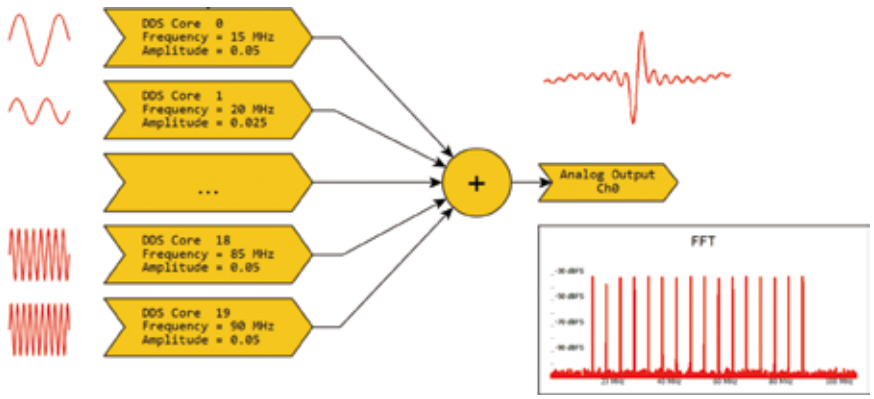


Figure1：複数のDDSコアを1つのアナログハードウェア チャネルに合計します

簡略化すると、出力 u(t) は次のように表すことができます。

$$u_{CH0}(t) = \sum_{i}^{i=N} u_{Core,i}(t)$$

With i = DDS core index

すべてのコア振幅値の合計は、最大値1.0または100%まで加算できます。たとえば、2つのコアを両方とも60%に設定すると、合計は120%になり、出力で整数オーバーフロー効果が発生します。

疑似コードの利用

Spectrum製カードはC++やPythonなどのさまざまなプログラミング言語をサポートしているため、ここでは擬似コードを使用してコードスニペットの可読性と汎用性を高めます。コードスニペット2は、C++コードスニペット1の結果の擬似コードを示しています。

```
spcm_dwSetParam_d64(hCard, SPC_DDS_CORE0_AMP, 1.0);
spcm_dwSetParam_d64(hCard, SPC_DDS_CORE0_FREQ, 100.0e6);
spcm_dwSetParam_i32(hCard, SPC_DDS_CMD, SPCM_DDS_CMD_EXEC_AT_TRG);
```

コードスニペット1：C++コード

```
CORE0_AMP    = 1.0
CORE0_FREQ   = 100.0e6
CMD          = EXEC_AT_TRG
```

コードスニペット2：疑似コード

■ DDS設定の変更

周波数、位相、振幅などのすべての設定は実行時に変更できます。変更は、EXEC_AT_TRIGコマンドを使用して順番にキューに入れられ、FIFOバッファリングされ、定義済みのトリガーイベントで同時に実行されます。可能なトリガーイベントは次のとおりです。

・カードトリガー：カードで使用可能なトリガーエンジン全体をサポートします（トリガーエンジンの設定については、ユーザーマニュアルを参照してください）。
・内部タイマー：定義済みの時間間隔で次のシーケンスを自動的にトリガーします。

さらに、EXECUTE_NOW コマンドを使用して、キューの最後から DDS モジュールに到達するとすぐに変更を実行できます。

トリガーソースとタイマー間隔は、他のすべてのパラメーターと同様に変更できるため、複雑なシーケンスの作成は簡単です。

DDS モジュールは、すべての操作に対して固定された時間ベースと固定されたタイミング解像度で動作します。

■ トリガー分解能とジッターの計算例

M4i.66xx カードのすべてのトリガーソースのタイミング分解能は t_{re} = 6.4ns で、156.25MHz に相当します。

たとえば、内部タイマーの最小値は 83.2ns ですが、6.4ns 単位で調整できます。したがって、タイマー値を（10us）に設定すると、正確に 10.0032us または 1563x6.4ns に設定されます。すべての AWG カードは外部クロックソースに同期できるため、タイマーの精度はクロックソースと同等になります。

外部トリガーを使用して次のキューに入れられたコマンドを実行する場合、トリガーも 6.4ns のタイミング分解能で検出されます。トリガーが AWG カードに対して非同期の場合、ジッターは +/-3.2ns、つまり合計 6.4ns になります。外部トリガーが AWG カードに同期され、位相がトリガーエンジンのサンプル＆ホールドウィンドウを常に満たすように正確に設定されている場合、ジッターを最小限に抑えることができます。

Spectrum 製 STAR-HUB モジュールを使用することで、複数ボード間のジッターを実質的に 0ns にすることができ、複数枚の AWG カードを簡単に同期することが可能になります。

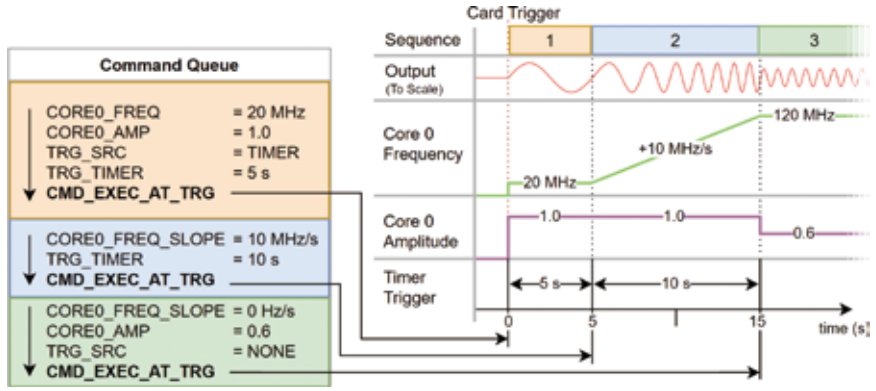


Figure 2：周波数の傾斜とシーケンス

固有周波数と振幅の傾斜

周波数、振幅、初期位相に加えて、DDS モジュールは動的パラメータとして線形周波数と振幅の傾斜もサポートします。

Example 3:

Figure 2 は線形傾斜を使用したシーケンスプログラミングの例を示しています。

シーケンスの順序：

リセット後、すべてのパラメータが 0 に設定され、トリガーソースは外部トリガーに設定されます。

1. トリガーイベントで、周波数は 20MHz に、振幅は 1.0 に設定されます。5 秒後に次のシーケンスに切り替えるには、トリガーソースを内部タイマーに設定します。内部タイマーは 5 秒のカウントを開始し、次のシーケンスがトリガーされます。
2. コアの周波数は、1 秒あたり 10MHz ずつ直線的に滑らかに増加します。したがって、10 秒後に最終周波数 120MHz に達します。
3. スロープを 0 に設定して停止し、同時に振幅を 0.6 に設定します。
4. 最後に、トリガーソースを NONE に設定して内部タイマーを無効にします。

■ 周波数と振幅スロープの量子化誤差

スロープは実質的に線形ですが、量子化されたタイミングと値の解像度を使用して、カードのロジック内で離散的なステップで計算されます。

M4i.66xx DDS モジュールの場合、最小スロープリフレッシュレート t_{re} は 6.4ns で、この値はほとんどのアプリケーションで十分な値です。

これらの量子化された計算により、スロープの増分とタイミング解像度は有限であり、目的の設定値と実際の値の間に一定の量子化誤差が発生する可能性があります。

通常、これらの影響は最小スロープステップに達する秒単位の非常に長いスロープ、またはタイミングの量子化が関係する短いスロープでのみ目立ちます。これらの影響を軽減するには、スロープの最後に最終周波数を手動で設定する必要があります。

■ タイミング量子化誤差

タイミング誤差は、タイマーの浮動小数点値を入力するときに生成され、ドライバーによって t_{re} ステップで完全なクロックサイクルに丸められます。約 1 ミリ秒の長い傾斜の場合、この量子化誤差は +/-3.2ns または 3.2ppm の範囲にあり、多くのアプリケーションでは無視できるほど小さいものです。アプリケーションとプログラミングでこの量子化を考慮し、t_{re} ステップの時間期間のみを使用することで、この誤差を軽減できます。

■ 動作タイミング誤差

さらに、発生する可能性のある別のタイミング誤差があります。実行イベントでスロープコマンドのみを使用すると、周波数がトリガーイベントと同

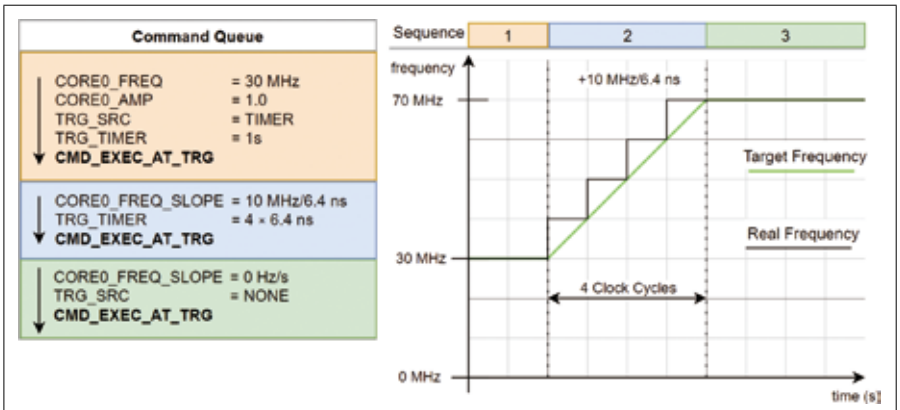


Figure 3：まず開始周波数を設定し、次にスロープを開始します

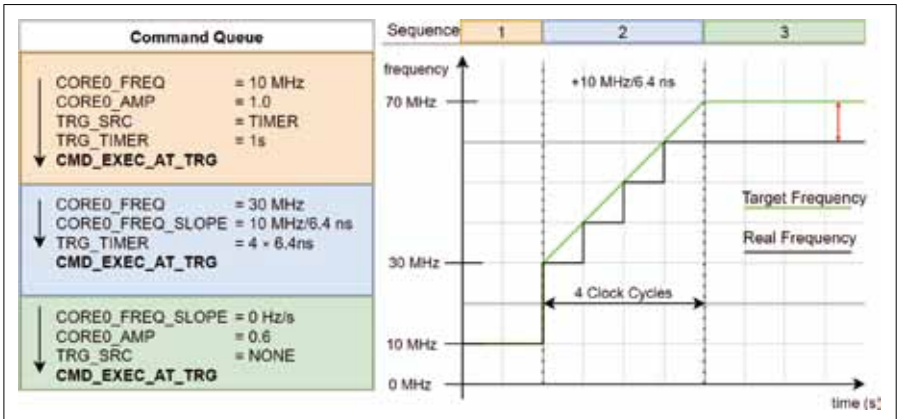


Figure 4：同じ実行イベントで周波数と傾斜を設定すると、最終値が異なります

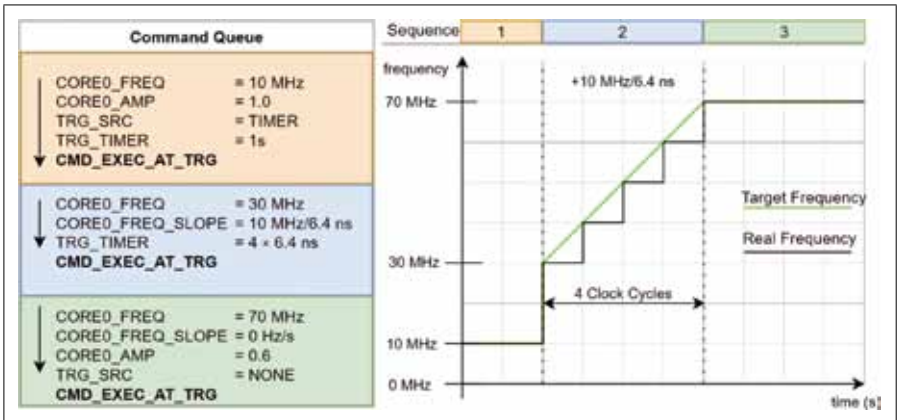


Figure 5：周波数と傾きは同じ実行イベントで設定され、終了値は明示的に設定されます

時に増加し（Figure3）、予想される終了周波数になります。

ただし、同じ実行イベントで周波数とスロープ速度を設定すると、Figure4に示すように、最初のクロックサイクルで周波数が設定され、次のクロックサイクルで周波数が増加します。

この動作は技術的には間違いではありませんが、スロープは予想されるよりも1クロックサイクル短くなります。この動作は、最終周波数を明示的に設定するか（Figure5）、開始周波数を1増分だけ調整することで簡単に説明できます。

表示目的で4クロックサイクルという非常に短いスロープ期間が選択されましたが、これは、最小タイマー時間が100nsの範囲であるため、EXEC_AT_TRIGコマンドではこれまで不可能でした。スロープ期間がミリ秒単位の場合、この影響は百万分の一単位になります。

■ 値量子化誤差

値誤差を減らすために、DDSの周波数分解能 f_{step} よりも高い勾配 f'_{step} 分解能で勾配が内部的に計算され量子化誤差が減ります。秒単位の長い期間の場合は、ステップディバイダを設定して分解能をさらに高めることができますが、通常ミリ秒単位のタイムスケールのアプリケーションでは必要ありません。

f'_{step} は、SPC_DDS_AVAIL_FREQ_SLOPE_STEPレジスタをAPIで読み取ることができ、f_{step} は、SPC_DDS_AVAIL_FREQ_STEPレジスタで読み取ることができます。

Example 4:

M4i.66xx DDSモジュールの周波数分解能 f_{step} は約0.3Hzです。スロープ速度を正確に f_{step}/t_{re} に設定すると、周波数は6.4nsごとに正確に0.3Hz増加します。これが最小周波数スロープ分解能 f'_{step} でもある場合、0.3Hz/6.4ns未満のスロープ値は不可能になり、スロープは実行されません。

1秒後、この最小値により周波数は47MHz変化します。したがって、1秒後には、0Hz、47MHz、2*47MHz、3*47MHzなどの量子化された周波数変更のみが可能になります。

要約すると、最大周波数勾配量子化誤差を計算するには、勾配分解能と勾配持続時間を掛けるだけです。中間の値では、より低い値またはより高い値に丸めるため、2で割ります。

$$f_{error} = \frac{f'_{step}\,\Delta t}{2}$$

実際に実装された周波数勾配分解能 f'_{step} は $f'_{step} = 694\text{Hz/s}$ の f_{step} よりも高くなります。したがって、1MHzから2MHzへの勾配を1ミリ秒で実現したい場合、最大誤差は

$\frac{f'_{step}\,\Delta t}{2} = 0.348Hz$ になりますが、これはほとんどのアプリケーションでは

```
// Set initial Values, keep them for one second
TRG_SRC           = TIMER
TRG_TIMER         = 1 s
CORE0_AMP         = 1.0
CORE0_FREQ        = 100 MHz
CMD               = EXEC_AT_TRG

// Start the Slope
TRG_TIMER         = 1 ms
CORE0_FREQ_SLOPE  = 1 MHz / 1 ms
CMD               = EXEC_AT_TRG

// Stop the Slope
CORE0_FREQ_SLOPE  = 0
TRG_SRC           = NONE
CMD               = EXEC_AT_TRG
```

コードスニペット 3：100MHz から 101MHz へのスロープを 1ms で実現

無視できる値です。

上記コードスニペット3は、1ミリ秒で100MHzから101MHzへの勾配を実現し、101.00000014 MHzで終了します。

位相の挙動

Figure2 からわかるように、周波数が変化しても位相は連続しています。これは、位相アキュムレータ $\boldsymbol{\theta_i(z)}$ を含むDDSコアのより正確な数学的表現によって説明できます。

$$u_{Core,i}(z) = \sin(\theta_i(z) + \varphi_i) \cdot A_i$$

$$\theta_i(z) = \theta_i(z - 1) + 2\pi \frac{f_i}{f_s}$$

リセットまたは位相ジャンプの場合:

$$\theta_i(0) = 0$$

式からわかるように、周波数 f_i を変更してもクロックサイクルあたりの位相増分 $2\pi \frac{f_i}{f_s}$ のみが変わり、位相アキュムレータの出力は時間とともに間接的にしか変化しません。たとえば、周波数を0に設定すると位相が停止しますが、位相コヒーレントDDSでは位相がジャンプします。

■ 位相シフトモード

デフォルトでは、初期位相 φ_i を 51° から52° に変更すると、出力信号の位相が前の状態に対して 1° 変化します。このモードは、ノブを回して位相を連続的に調整できる一般的な測定器のように動作します。考えられる用途としては、位相シフトキーイング (PSK) などの変調方式が挙げられます。

■ 位相コヒーレント動作/位相ジャンプモード

一部のアプリケーションでは、周波数が切り替わるたびに位相がジャンプする位相コヒーレント周波数スイッチングが行われます。この時間多重方式では、単一のDDSコアを使用して複数の周波数を同時に出力できます。実際には、すべての周波数を同時に合計する場合と比較して、アナログパスから必要なダイナミックレンジが低くなるという利点があります。

前述のように、位相は通常連続していますが、DDSモジュールにはコヒーレント動作用の位相ジャンプモードも備わっており、位相が設定されるたびに位相アキュムレータ $\theta_i(z) = 0$ がリセットされます。

Example 5:

Figure 6は、1つのDDSコアに時分割多重化された2つの異なる周波数を示しています。時間平均すると1つのコアのみが使用されていますが、周波数スペクトルには両方の周波数が表示されます。

実際のアプリケーションでは、スイッチングスプリアスの影響を減らすために、必要な信号周波数よりもはるかに低いスイッチング周波数を選択することをお勧めします。

この動作を実現するには、スイッチングポイントで周波数と位相の両方を設定する必要があります。スイッチングポイントで必要な位相を° 単位で計算するには、簡単な計算のみです。

$\varphi(t) = f{\cdot}t{\cdot}360^\circ mod\ 360^\circ$

$\varphi(1s) = 1.4MHz{\cdot}1\mu s{\cdot}360^\circ mod\ 360^\circ = 144^\circ$

$\varphi(2s) = 3MHz{\cdot}2\mu s{\cdot}360^\circ mod\ 360^\circ = 0^\circ$

$\varphi(3s) = 1.4MHz{\cdot}3\mu s{\cdot}360^\circ mod\ 360^\circ = 72^\circ$

中間のすべてはハードウェアで計算されます。したがって、すべてのDDSコアのスイッチング周波数を100kHzの範囲で簡単に切り替えることができ、DDSで生成できるトーンの数が実質的に増加します。

カスタム周波数スロープ

DDSモジュールは線形勾配のみをサポートしていますが、他の勾配タイプも目的の形状の区分線形補間を形成する有限数の線形周波数勾配を使用して簡単に近似できます。たとえば、冷却原子物理学の世界の多くのアプリケーションでは、正弦波、いわゆる「S字型」の周波数勾配が必要です (Figure 7)。使用用途で必要な滑らかさに応じて、ユーザーは特定の数のセグメントを選択できます。表示目的では、Figure 7に示すように、約7つのセグメントがターゲットに一致するようです。以下では、これらの勾配を計算してプログラムする方法について説明します。

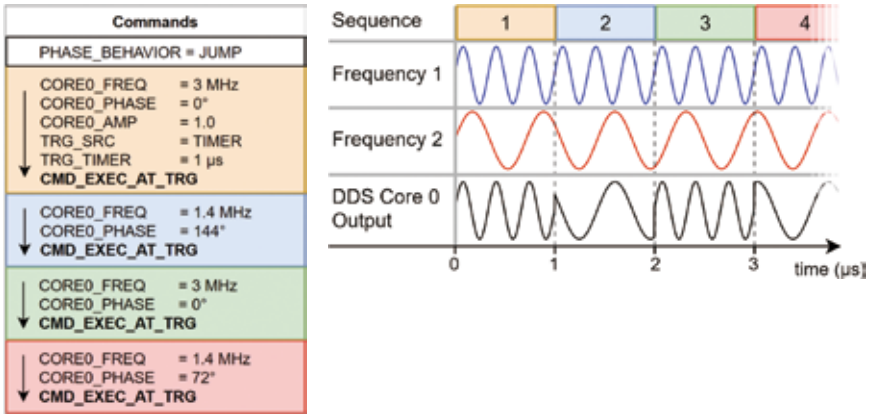


Figure 6：単一の出力で 2 つの異なる周波数が生成されます

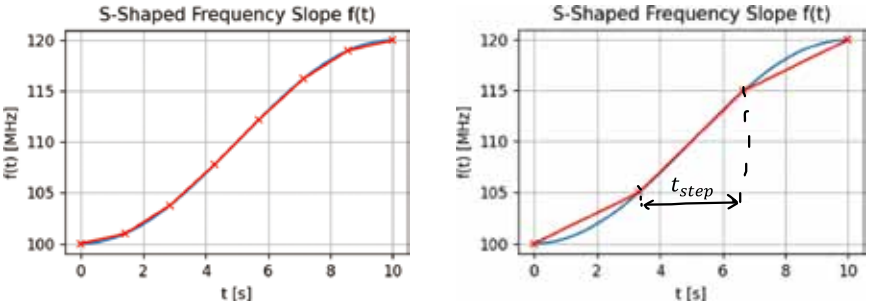
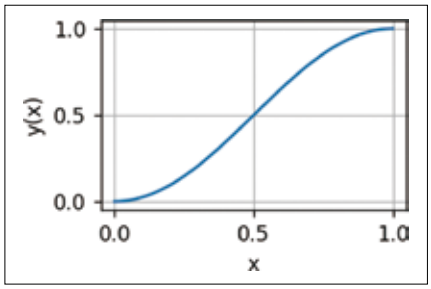


Figure 7：n=7 セグメントの近似波形と近似後の実際の波形

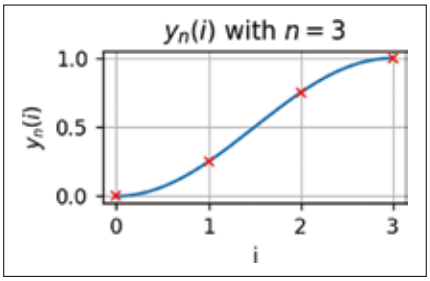
1. まず、正弦関数を使用して目的のS字型関数を定義する必要があります。x=0が傾斜の始まり、x=1が傾斜の終わりです。

$$y(x) = \frac{-\cos(x \cdot \pi) + 1}{2}$$



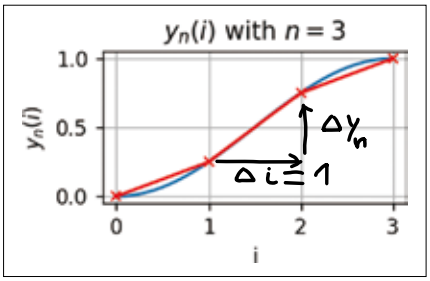
2. 次に、その範囲を必要なステップ数に調整し、完全な整数ステップに量子化します。これは、$x = i\text{-}n$ を i: 0, 1, 2, . . . n に置き換えることで行われます。

$$y_n(i) = y\left(\frac{i}{n}\right)$$



3. i = 0、1、2、… n-1 の各線形線分の単位なし勾配 $M(i)$ を計算します。
これらの値は、実行を高速化するために事前に計算してルックアップテーブルに保存できます。
前述のように、位相は通常連続していますが、DDSモジュールにはコヒーレント動作用の位相ジャンプモードも備わっており、位相が設定されるたびに位相アキュムレータ $\theta i(z) = 0$ がリセットされます。

$$M(i) = \frac{\Delta y}{\Delta y} = \frac{yn(i+1) - yn(i)}{1}$$



4. $M(i)$ を使って、$s(i, \Delta f, \Delta t)$ という単純な関数を構築できます。この関数は、$M(i)$ を特定の周波数差 Δf と時間差 Δt にスケーリングします。

$$s(i, \Delta f, \Delta t) = M(i) \cdot \frac{\Delta f}{\Delta t}$$
$$\Delta f = f_{end} - f_{start} = 120\text{MHz} - 100\text{MHz} = 20\text{MHz}$$

5. 初期周波数を定義し、開始時に5秒間保持します。

CORE0_FREQ = f_{start}
TRG_SRC = TIMER
TRG_TIMER = 5s
CMD = EXEC_AT_TRG

6. 区分線形Δiシーケンス間の時間t_{step}を設定します。

TRG_TIMER = t_{step}

7. i = 0、1、2、… n-1 の各ステップの傾きを設定します。

for (i = 0; i < n; i++)
{
CORE0_FREQ_SLOPE = f_{start}+$s(i,\Delta f,\Delta t)$
CMD = EXEC_AT_TRG
}

8. 最後に最終周波数を設定し、傾きを停止し内部タイマーを停止します。

CORE0_FREQ = f_{end}
CORE0_FREQ_SLOPE = 0
TRG_SRC = NONE
CMD = EXEC_AT_TRG

DDSコマンドを使用したXIOラインの制御

DDSは3つの異なるXIO出力モードをサポートしています。

- **Manual：**
  このモードでは、DDSコアの周波数と同様に、出力状態（highまたはlow）を制御できます。
- **Wait for trigger：**
  XIOラインは、DDSモジュールがトリガーを待機しているかどうかを示すステータス信号として機能します。EXECUTE_AT_TRIGを受信した後、DDSモジュールがトリガー待ちを開始すると、XIOラインはhighになります。プログラムされたトリガー信号（タイマーまたはカードトリガー）を受信すると、XIOラインはlowになります。
- **Execute：**
  XIOラインはステータス信号として機能し、トリガーまたは今すぐ実行コマンドが受信され、新しいパラメータが出力にロードされるたびにhighになります。

すべての信号は出力に合わせて調整されます。例については、Figure 15 を参照してください。

したがって、「Manual」モードに設定されたXIOラインをlowからhighに変更し、同じトリガーイベントで出力振幅を0から100%に変更する場合、両方の変更をほぼ同時に測定できます。振幅と手動XIOラインの変更と同時に、「Execute」モードに設定されたXIOラインが6.6ns の間highになり、その後lowになります。「Wait for trigger」ラインは、トリガーが到着すると同時にlowになり、次の「EXECUTE_AT_TRIG」コマンドがカードに送信されるとすぐにhighになります。

ソフトウェア駆動のフィードバックループ / 意思決定

AWGカード、ブラックボックスシステム、デジタイザーで構成される閉じたフィードバックループを想像してください。このループ内でAWGは周波数 f を使用してシステムを刺激し、システムは周波数を急速に上げるか、周波数を下げるようにシステムに信号を送ることで応答します。

この例では、新しい周波数値は一定の応答時間 $t_{response}$ 内にシステムに到達する必要があります。この応答時間はFigure 8に示されており、3つの異なる主要コンポーネントで構成されています。

$$t_{response} = t_{measure} + t_{processing} + t_{latency}$$

- $t_{measure}$ は、変化した入力信号を測定するのにかかる時間です。
- $t_{processing}$ は、必要な新しい出力パラメータ/データを計算する時間です。
- $t_{latency}$ は、ソフトウェアが新しいパラメータを設定してからカード出力に至るまでの遅延です。

トリガーイベントに必要な周波数が事前にわかっていれば、次のトリガーイベントの周波数値をプリロードして、1マイクロ秒以内に簡単に反応できます。これが「Trigger-to-Output」の時間です。ただし、このリアルタイムフィードバックアプリケーションでは、次にどの出力周波数が必要になるかは事前にわかりません。

つまり、新しい入力パラメータを受け取った後にどのように反応するかを決定し、時間内に反応を実行する必要があります。

以前のAWG FIFOモードでは、新しいデータを最初に計算してからAWGメモリに送信する必要がありました。このバッファリングのため、システムで大きな遅延が発生します（通常は数ミリ秒程度）。AWGシーケンスモードでは、事前に保存されたシーケンスを切り替えることでこの時間を短縮できます。ただし、切り替えプロセスによって時間の不確実性（数百ナノ秒程度）が生じる可能性があります。

DDSモードでは、必要な出力周波数のみを計算し個々のサンプルの長いシーケンスを計算する必要がないため、処理時間を短縮できます。さらに、転送するデータは最小限であるため、バッファリングが最小限で済み遅延時間も大幅に短縮できます。

要約すると、新しいDDSモードは2つの異なる実行コマンドをサポートしています。EXEC_NOWでは、新しい周波数値を可能な限り高速に出力にロードできます。

EXEC_AT_TRIGでは、新しい値が明確に定義されたタイミングで選択されたトリガー ソースに正確にロードされます。したがって、最速の応答時間を実現するには EXEC_NOW を使用できますが、確定的なタイミングと特定のレイテンシを実現するには、EXEC_AT_TRIGを使用する必要があります。これについては、次のセクションでさらに詳しく説明します。

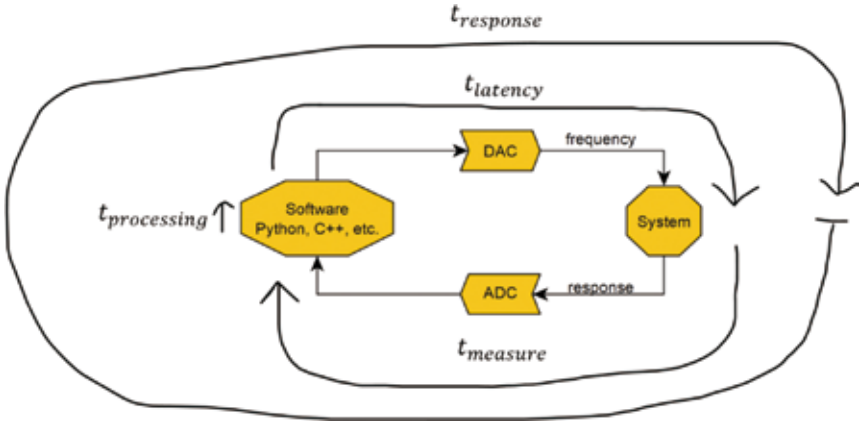


Figure 8：フィードバック ループとタイミングの例

■ EXEC_NOWで応答時間を測定する

ソフトウェアから物理的な出力までの遅延時間 $t_{latency}$ を測定するのは非常に困難です。キーを押してから文字が表示されるまでの時間を測定することを想像してください。指の押下によって定義される開始時間は、ミリ秒のオーダーで測定する場合、非常に不正確です。

ただし、完全な応答時間 $t_{response}$ は簡単に測定できます。つまり、処理システムがソフトウェアでポーリングして入力の変更を検出し、新しい周波数値を送信して別の信号を出力するのにかかる時間です。ここで、ADCの代わりにM4iカードにすでに搭載されている2つの補助デジタルIOラインを使用します。IOラインXIO-0がHighになると、周波数を可能な限り速く上げ、XIO-1がHighになると周波数を下げます。XIOラインはトリガー入力としては使用されず、ソフトウェアによってポーリングされるだけであることに注意してください。

このパラメータは、さまざまなシステムとプログラミング言語でテストされています。いくつかのコマンドの結果を、Table 1: レイテンシ測定結果に示します。ただし、すべての非リアルタイムオペレーティングシステムと同様に、これらのレイテンシ値は保証されていません。たとえば、ウイルススキャンによってシステムが停止しレイテンシが大幅に増加する可能性は常にあります。そのため、次のセクションで説明する「詳細なテスト設定」を使用して、独自のシステムでこの値をテストすることをお勧めします。

■ 詳細なテスト設定

IOラインXIO-0およびXIO-1の信号ソースとして、2つのプッシュボタンを使用します。反応するソフトウェアによって導入された変更の結果を明確に確認するために、別のIOライン XIO-2を出力として使用し、これをlow(0)からhigh(1)に切り替えます。また、信号周波数を変更するアナログ出力を使用します。

Condition	Typical achievable Roundtrip Times
C++ on Windows 10, Intel i7 13700k	20 µs
Python on Windows 10, Intel i7 13700k	70 µs

Table 1：レイテンシ測定結果

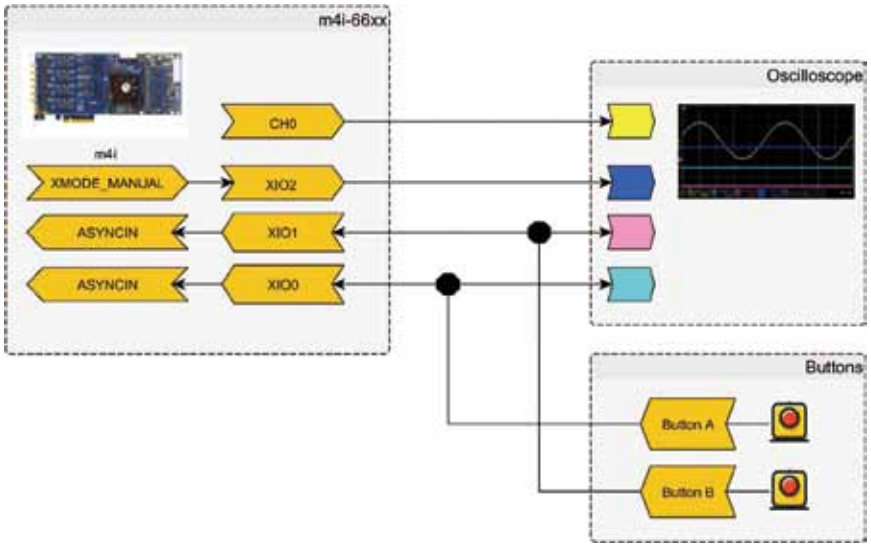


Figure 9：ソフトウェア設定から出力遅延までのテストセットアップ

```
// Send initial Values
X0_MODE = ASYNC_IN
X1_MODE = ASYNC_IN
X2_MODE = DDS_MANUAL

CORE0_AMP      = 1.0
CORE0_FREQ     = 0.5 MHz
MANUAL_OUTPUT = X2 -> 0
CMD            = EXEC_AT_TRG
CMD            = WRITE_TO_CARD
M2CMD_CARD     = FORCETRIGGER

// Poll XIO Lines until either one is High
while (1)
  {
  if (get_param(XIO0) == 1)
        {
        CORE0_FREQ       = 0.1 MHz
        CORE0_AMP        = 1.0
        MANUAL_OUTPUT    = X2 -> 1
        CMD              = EXEC_NOW
        CMD              = WRITE_TO_CARD
        break;
        }
  else if (get_param(XIO1) == 1)
        {
        CORE0_FREQ       = 1 MHz
        CORE0_AMP        = 1.0
        MANUAL_OUTPUT    = X2 -> 1
        CMD              = EXEC_NOW
        CMD              = WRITE_TO_CARD
        break;
        }
  }
```

レイテンシテストを出力するソフトウェアの擬似コード

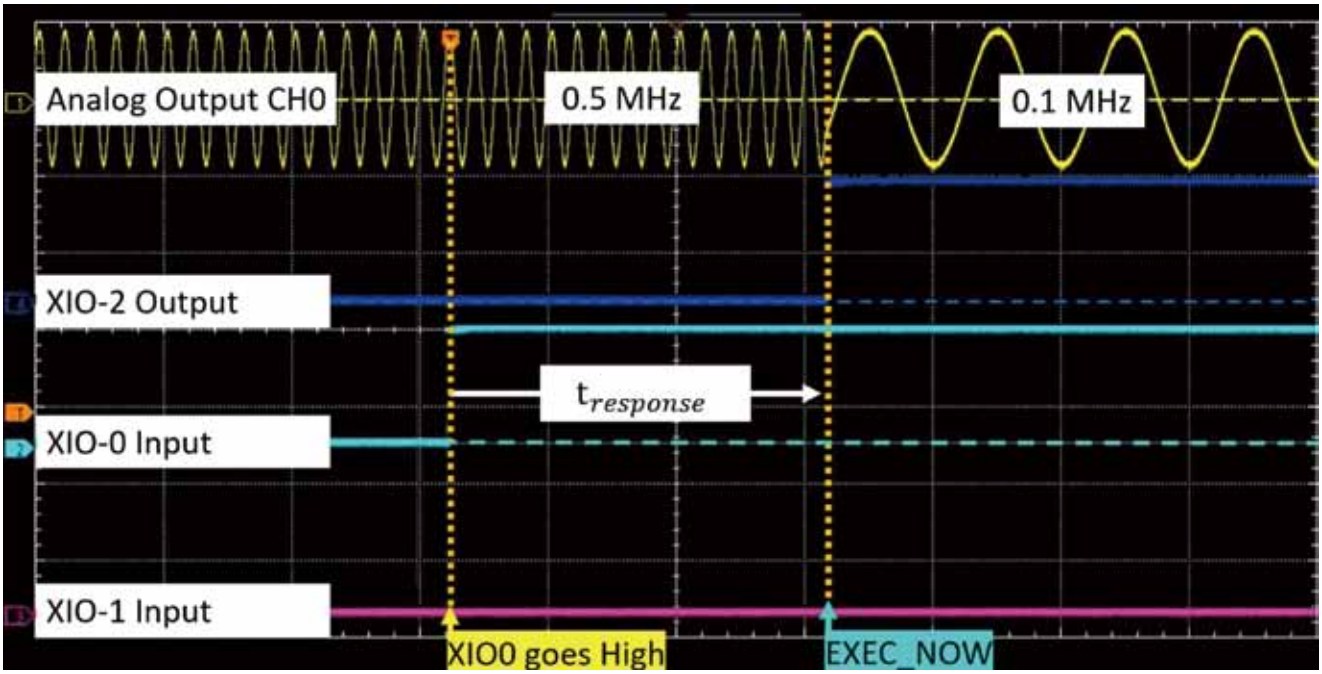


Figure 10：ソフトウェア設定から出力までの遅延テストの結果の出力波形画像。ボタン A が押されました

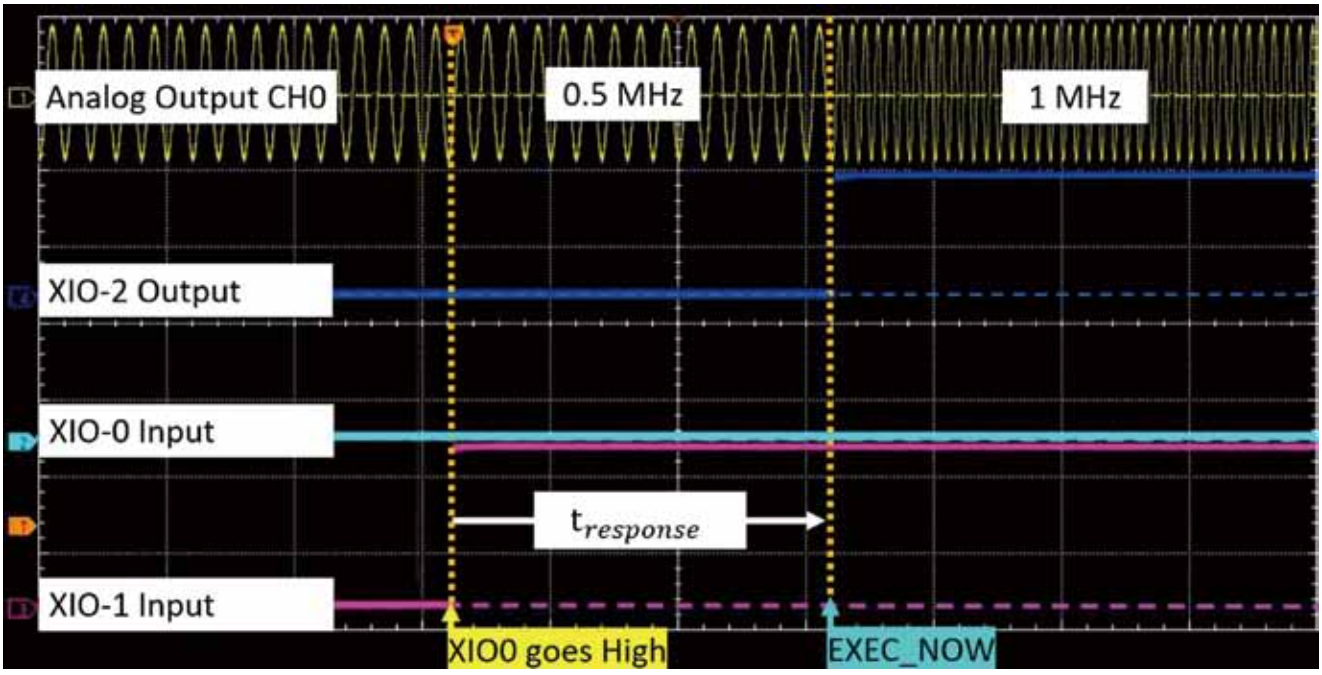


Figure 11：ソフトウェア設定から出力までの遅延テストの結果の出力波形画像。ボタン B が押されました

最後に、IO ラインXIO-0 またはXIO-1 が high になってから出力信号が変更されるまでの時間遅延が、私たちが関心のあるものです。この時間は、オシロスコープを使用して表示および計算できます。

■ EXEC_AT_TRIG による確定的レイテンシの実現

外部トリガーから出力周波数の動的変更までのレイテンシは、固定遅延を持つ2つのトリガーイベントを使用することで固定できます。最初のトリガーイベントは新しい入力パラメータの読み出しをトリガーするために使用され、2番目のトリガーは新しい周波数値を実行するために使用されます。

たとえば、外部トリガーイベントに対するソフトウェアの応答時間が最大20µs から80µs の間で変動することがわかっている場合は、2番目のトリガーイベントを最初のトリガーイベントの100µs後に設定します。

これで、新しい周波数値は常に最初のトリガーイベントの 100µs後に出力されます。2番目のトリガーイベントでは、再び外部トリガーまたは組み込みタイマーを使用できます。

■ テストセットアップ

M4i.66xx シリーズのカードはトリガー入力として XIO ラインをサポートしていないため、次の周波数を選択するには別の方法を使用する必要があります。そのため、AWG カードのトリガー入力 TRG0 に接続された 1 つのプッシュボタンを使用し、IO ライン XIO0 にスイッチを接続します。プッシュボタンによって生成されたトリガーイベントでスイッチが high の位置にある場合は、周波数 A 出力が必要になり、スイッチが low の位置にある場合は周波数 B 出力が必要になります。そのため、最初にスイッチを目的の XIO-0 レベルに設定してから、ボタンを押す必要があります。これらの信号をデジタルで制御する場合は、もちろん、トリガーと同時に XIO-0 ラインを切り替えることができます。

プログラミングに関しては、XIO ラインが変更されたかどうかをポーリングする代わりに、カードのトリガーエンジンを使用します。ここでは、TRG0 がハイになるとトリガーするように設定します。DDS には、トリガーが受信されると low になるステータス信号「WAITING_FOR_TRIG」があります。この信号が low になるまでポーリングし、XIO-0 が 0 または 1 のどちらをリードバックするかを確認します。これに応じて、次のトリガーイベントの値を設定します。

2 番目のトリガーイベントで確定的なレイテンシを実現するには、最初のトリガーイベントで組み込みのタイマートリガーを選択して 100µs に設定します。

IO ラインの値は、トリガーが押された時点とタイマートリガーが次のシーケンスを実行する時点の間のどこかで 1 回だけ評価されます。したがって、Figure 13 と Figure 14 の X マーカーは、「Don't Care」の信号領域を示します。

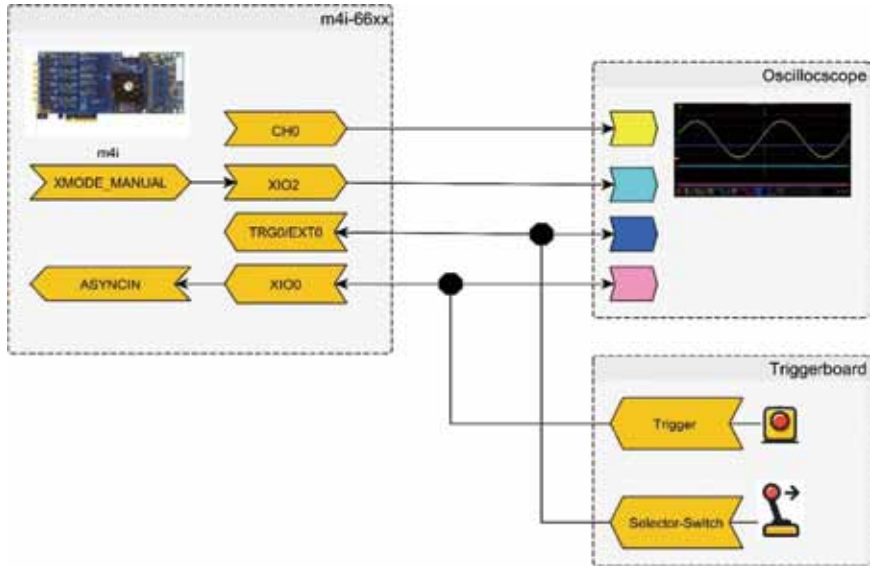


Figure 12：意思決定を伴う確定的遅延のテスト設定

```
frequency_initial        = 100 kHz
frequency_a              = 20 kHz
frequency_b              = 300 kHz

X0_MODE        = ASYNC_IN
X2_MODE        = DDS_MANUAL
// Setup the card trigger engine to trigger on EXT0
TRIG_ORMASK    = SPC_TMASK_EXT0

// Send initial Values
CORE0_AMP      = 1.0
CORE0_FREQ     = frequency_initial
MANUAL_OUTPUT = X2 -> 0

CMD            = EXEC_AT_TRG
CMD            = WRITE_TO_CARD
// Force Trigger initial settings
M2CMD_CARD     = FORCETRIGGER

/* This yellow marked EXEC_AT_TRG is triggered by the external
Trigger and starts the timer for the NEXT Blue EXEC_AT_TRG */
DDS_TRIG_SRC   = TIMER
TIMER           = 100µs
CMD            = EXEC_AT_TRG
CMD            = WRITE_TO_CARD

// Wait until we received the trigger
while (get_param(WAITING_FOR_TRIG) == 1)
    {
    // Wait
    }

/* Send new Data based on IO Line state.
The Data needs to be send before the timer runs out */
if (get_param(XIO0) == 0)
    {
    CORE0_FREQ       = frequency_a
    MANUAL_OUTPUT    = X2 -> 1
    CMD              = EXEC_AT_TRG
    CMD              = WRITE_TO_CARD
   }
else
   {
    CORE0_FREQ       = frequency_b
    MANUAL_OUTPUT    = X2 -> 1
    CMD              = EXEC_AT_TRG
    CMD              = WRITE_TO_CARD
    }
```

決定論的な遅延を伴う意思決定のための疑似コード

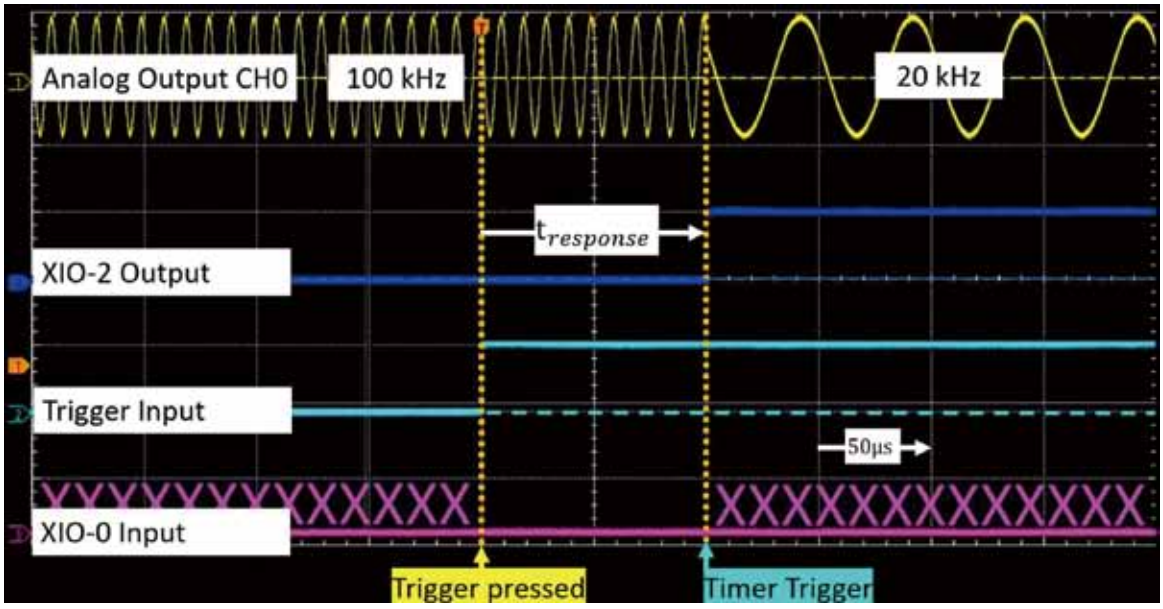


Figure 13 : 意思決定テストによる確定的レイテンシの出力波形画像。決定 A、XIO-0 は外部トリガーイベントで low になります。$t_{response} = 100\mu s$

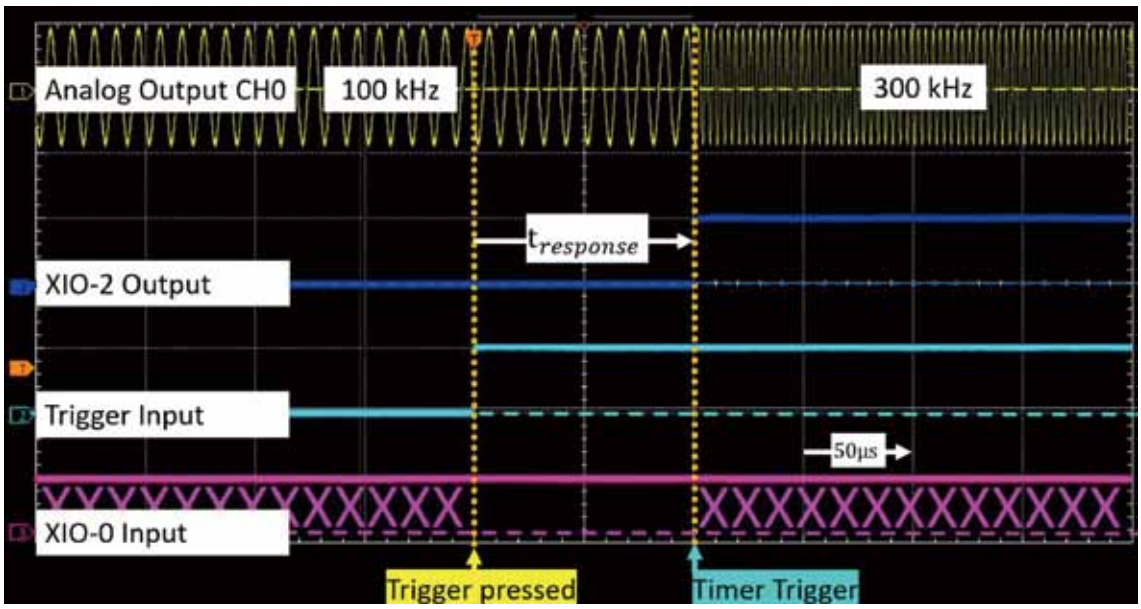


Figure 14 : 意思決定テストによる確定的レイテンシの出力波形画像。決定 B、XIO-0 は外部トリガー イベントで high になります。$t_{response} = 100\mu s$

継続的なストリーミング

101.3 MHzで独自のFMラジオ局を作成し、AWGカードをソフトウェア定義ラジオ(SDR)として使用するとします。前述のように、以前はサンプルごとにデータをストリーミングする必要があり、1.25GHzのサンプルレートで2GB/sを超えるデータストリームが発生しました。これには通常、専用のCUDA GPUアクセラレータカードが必要でした。

現在、DDSモードでは、新しい周波数値で構成される周波数変調信号のみを送信する必要があります。20kHz帯域幅のオーディオ信号の場合、44.1kHzのサンプリングレートが一般的です。この信号は、最小限の計算能力でDDSに継続的にストリーミングできます。

ストリームの信頼性を高めるために、大きなFIFOバッファを使用できます。Spectrum APIは、各コマンドを64ビット、つまり8バイトに収めます。これにより、カードの4GBオンボードメモリに最大5億のコマンドを保存できます。ただし、これによりストリームのレイテンシも増加します。 44.1kHzのコマンドレートでバッファがいっぱいになると、遅延は3時間強になります。ただし、バッファリングするコマンドの最大数はいつでも選択できます。

■ 最大ストリーミング速度

ストリーミング速度は、ユーザーアプリケーションとシステムに大きく依存します。M4i.66xxでのPython実装によるテストでは、約300kHzの信頼性の高い速度に達し、C++では10MHzを超える連続ストリーミングコマンドレートが正常にテストされました。

Execute-Nowを使用したシングルステッププログラミング

20秒間という長い期間にわたって100MHzの連続信号を生成し、その間に数ナノ秒ごとに変化する短い任意の信号を追加したいとします。残念ながら、タイマーの再アーム時間は 100ns程度と最小限です。ただし、より複雑な「1つの命令」/クロックサイクルアプローチを使用すれば、出力信号を6.4nsごとに変更できます。

これは、Execute-At-Trigコマンドの代わりに単一のExecute-Nowコマンドを使用することで可能になります。これらのコマンドは、コマンドキューの最後に到着するとすぐに実行されます。したがって、キューの状態に応じてタイミング動作が異なります。

- キューが空でカードがトリガーを待機していない状態でカードに送信された場合、それらは「ソフトウェアから出力までの遅延」で定義された可能な限り高速に実行されます。
- 2つのExecute-Nowコマンドがキューに続けて入れられた場合、それらは6.4ns間隔で実行されます。
- Execute-NowコマンドがExecute-At-Trigコマンドの後にキューに入れられた場合、それらはトリガー後の固定遅延時間 $t1$ で実行されます。

Example 6:

トリガー信号の後に正弦波を出力し、XIO出力2 を使用してデジタルデバイスを制御するとします。デバイスは、Figure 15に示すように、0‘ と 1’ の特定のシーケンスを期待します。右のリストは対応するコマンドリストです。

■ M4i.66xxカードのDDSモードとAWGモードの比較

AWGモードは、出力信号を事前に計算してメモリに保存できる場合に最適です。そうでない場合は、ループ、ストリーミング、またはシーケンスモードを使用する必要があり、セットアップが少し複雑になります。

閉ループフィードバックシステムがある場合は、高レイテンシでデータをストリーミングするか、シーケンスモードを使用する必要があります。

信号が正弦波で構成されている場合は、DDSコマンドを使用して記述するだけです。内部カードバッファに制限されず、バッファサイズを計算する必要もありません。バッファリングする必要があるデータの量は非常に少ないため、コマンドFIFOのレイテンシを非常に短く抑えることができます。実行時（シーケンス）にDDS設定を切り替えると、外部トリガーから出力までのレイテンシ時間が固定されます。変更を送信するだけでよいため、連続データ生成には多くの計算能力は必要ありません。

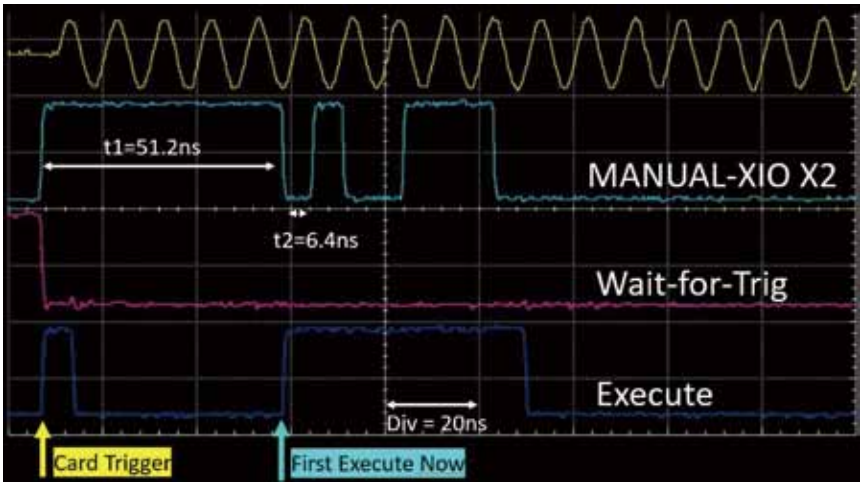


Figure 15：さまざまなXIO出力モードと単一の「Execute-Now」コマンドを使用した波形画像

```
CORE0_AMP          = 0.7
CORE0_FREQ         = 30 MHz
MANUAL_OUTPUT   = X2 -> 1
CMD                = EXEC_AT_TRG

MANUAL_OUTPUT   = X2 -> 0
CMD                = EXEC_NOW
MANUAL_OUTPUT   = X2 -> 1
CMD                = EXEC_NOW
MANUAL_OUTPUT   = X2 -> 0
CMD                = EXEC_NOW
MANUAL_OUTPUT   = X2 -> 0
CMD                = EXEC_NOW
MANUAL_OUTPUT   = X2 -> 1
CMD                = EXEC_NOW
MANUAL_OUTPUT   = X2 -> 1
CMD                = EXEC_NOW
MANUAL_OUTPUT   = X2 -> 1
CMD                = EXEC_NOW
MANUAL_OUTPUT    = X2 -> 0
```

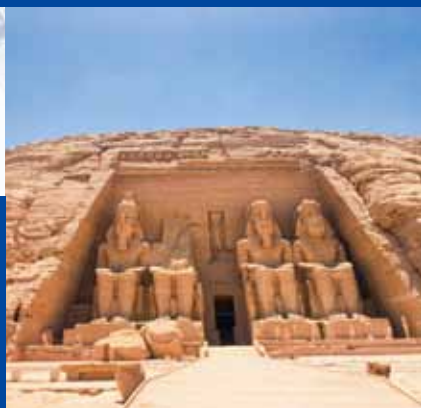
コマンドリスト

まとめ

DDSファームウェアオプションは、その高い柔軟性により幅広いアプリケーションをサポートします。レイテンシを削減できるため、従来のAWGモードよりも多くの使用用途が可能になります。さらに、プログラミングが容易になり、これまでAWGモードが使用されていた既存のアプリケーションが簡素化されます。

つまり、M4i.66xxシリーズ Spectrum製カードは、信号生成カードの無数のアプリケーションをサポートします。

リファレンスドキュメント：
Spectrum Instruments社
Product Note：Using the DDS Mode



AWGでダイレクトデジタル合成（DDS）を使用する利点

はじめに

任意波形発生器（AWG）は、電子テストに使用できる最も強力で柔軟性の高い信号源の 1 つです。AWGは、帯域幅と波形メモリの長さの範囲内で、ほぼ無限の数の波形を生成できます。AWGから信号を出力するためには、有用な波形を作成する必要があります。AWG用のアプリケーションソフトウェアを使用すると、方程式を使用して非常に正確な波形を作成したり、デジタイザまたはデジタルオシロスコープを使用してキャプチャした波形データを再生したりできます。テスト波形の作成、キャプチャ、変更、転送にかかるコストは非常に高額になる可能性があるためこのプロセスを簡素化するツールは非常に貴重です。

DDSオプション

Spectrum社は、16ビットの任意波形発生器（AWG）シリーズ向けに、ダイレクトデジタル合成（DDS）ファームウェアオプションをリリースしました。DDSは、単一の固定周波数リファレンスクロックから周期的な波形を生成する方法です。このDDSオプションは、複数の「DDSコア」を使用して各キャリアの周波数、振幅、位相が明確に定義されたマルチキャリア（マルチトーン）信号を生成します。Figure 16 は、最大1.25GS/sおよび400MHz の信号帯域幅を備えたM4i.66xxシリーズ AWGでのDDS信号生成のアーキテクチャを示しています。DDSオプションの信号ルーティングにより、使用可能なDDSコアと出力チャネル間のさまざまな接続が可能になり、最大限の柔軟性が得られます。M4i.66xxシリーズ向けに最初にリリースされたDDSオプションでは、合計23個のDDSコアが使用され、最大20個のDDSコアが1つのチャネルにルーティング可能になります。

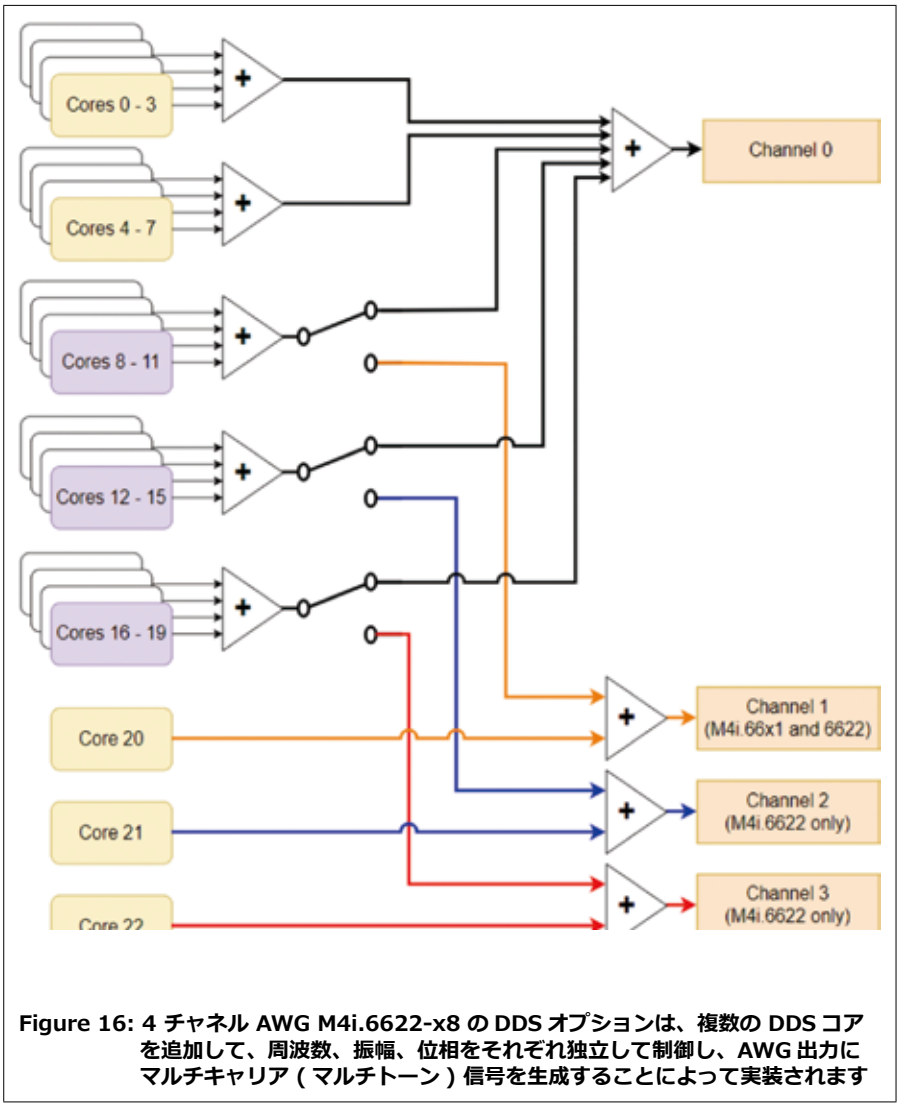


Figure 16: 4 チャネル AWG M4i.6622-x8 の DDS オプションは、複数の DDS コアを追加して、周波数、振幅、位相をそれぞれ独立して制御し、AWG 出力にマルチキャリア (マルチトーン) 信号を生成することによって実装されます

正弦波出力のプログラミング

DDSオプションは、波形作成において大きな利点があります。AWG波形は、ポイントごとに記述する必要があります。1メガサンプルの波形がある場合、その波形のすべての電圧値を計算して波形メモリに保存する必要があります。これに対し、DDSオプションは簡単なコマンドで記述されます。Figure 17は、100MHz正弦波を生成する方法の例を示しています。これらのシンプルなテキストベースのソースコマンドは、コンパイルされた後さらに簡潔な形式でDDSファームウェアに送信され、同等のAWG波形よりもはるかに少ないメモ

リで実行します。この例では、周波数と振幅が指定されています。コマンドは、Python、C/C++、MATLAB、LabVIEW、およびその他の多くのプログラミング環境で記述できます。

各キャリアの振幅、周波数、位相を設定でき、さらに周波数または振幅の線形変化を指定できます。たとえば、250MHzキャリアに対して5MHzのレーダーチャープを生成する場合のDDSコアコマンドは右のとおりです。(Code1参照)

周波数は247.5MHzから始まり、10µsかけて252.5MHzまで直線的に増加します。信号のFFTスペクトルをFigure 18に示します。

方程式を使用してAWGで同じ波形を計算するには、1.25GHzのサンプリングレートで12500サンプルの電圧値を計算する必要があります。

対照的に、DDSコアはわずか6つのコマンドを使用して同じ信号を生成します。さらに、これらのコマンドは、それぞれが64ビットつまり8バイトに収まる最適化されたドライバーを介してAWGに送信されます。これにより、AWGの4GBオンボードメモリに最大5億のコマンドを保存できます。

DDSコマンドは、ユーザーが指定したトリガーイベントと同期して実行されます。トリガーイベントには、AWGカードの内部トリガーロジックで生成されたものや、この場合のように内部タイマーを使用して生成されたものが含まれます。慎重に設計することで、DDS出力は、わずか6.4nsの分解能で任意の外部トリガーイベントまたはプログラム可能なタイマーと同期できます。

各コアは、独自の変調正弦波を生成できます。別のレーダーの例として、250MHzのキャリア、10kHzのパルス繰り返し周波数 (PRF)、10µsのパルス幅を持つレーダーパルス列を生成することが挙げられます。DDSコマンド例は右のとおりです。(Code2参照)

Figure 17：複数の DDS コアを 1 つのアナログハードウェアチャネルに合計します

```
// Chirp, 10µs duration, linear frequency sweep starting at 247.5 MHz to 252.5 MHz
CORE0_FREQ = 247.5 MHz
CORE0_FREQ_SLOPE = 5/10 MHz/µs
CORE0_AMP = 1.0 // 100% Amplitude
TRG_SRC = TIMER
TRG_TIMER = 10µs
CMD = EXEC_AT_TRG
```

Code1：250MHz キャリアに対して 5MHz のレーダーチャープを生成する場合のコマンド例

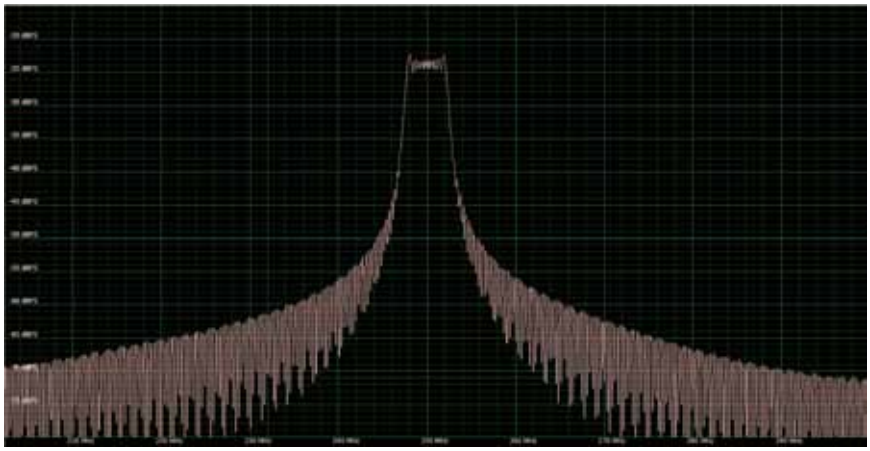


Figure 18：レーダーチャープの FFT スペクトルは、247.5MHz から 252.5MHz までの周波数の線形変化を示します

```
// Initial Settings
PHASE_BEHAVIOR = JUMP
CORE0_FREQ = 250 MHz
TRG_SRC = TIMER
for(i = 0; i < pulse_count; i++){

        // Switching on for 10 µs
        CORE0_PHASE = 0 // Start at Phase 0 each Cycle
        CORE0_AMP = 1.0 // 100% Amplitude
        TRG_TIMER = 10µs
        CMD = EXEC_AT_TRG

        // Switching of for 90µs to get 10kHz repetition Rate
        CORE0_AMP = 0
        TRG_TIMER = 90µs
}
```

Code2：250MHz のキャリア、10kHz のパルス繰り返し周波数 (PRF)、10µs のパルス幅を持つレーダーパルス列を生成するのコマンド例

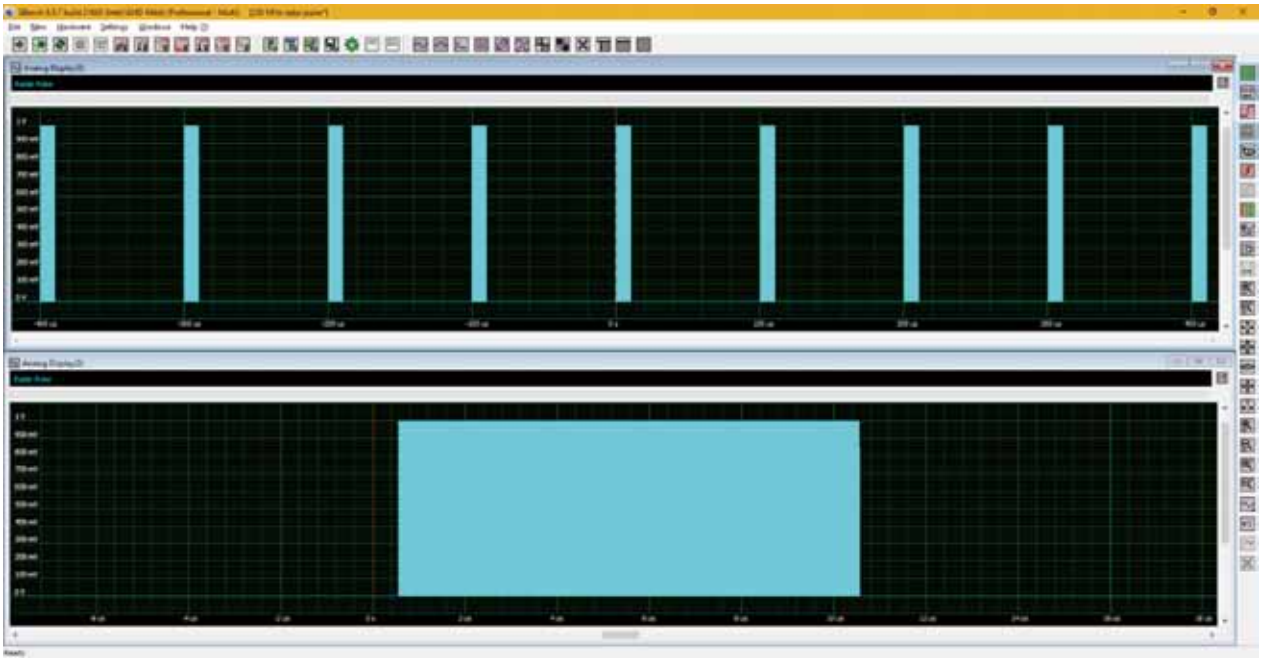


Figure 19：DDS オプションを使用して作成された、250MHz キャリア、10kHz PRF、および 10µs パルス幅のレーダーパルス。1 目盛りあたり 100µs（上）および 2µs（下）の水平スケールで表示されます

結果として得られるパルスストリームをFigure 19に示します。

これは、正弦波キャリアをゲートする機能を示しています。キャリア振幅をゼロに設定するとオフになります。振幅を設定する同じ手法を使用して、キャリアを振幅変調することができます。

デュアルトーン波形

DDSオプションでは、複数の加算キャリアを作成できます。マルチキャリア波形の最も一般的な使用法は、相互変調歪み（IMD）テストです。デュアルトーン波形（異なるキャリア周波数の正弦波）を加算してデバイスに適用します。デバイスの非線形性により、特定の周波数で混合信号が発生します。デバイスの出力スペクトルには、元の信号周波数、和周波数と差周波数、および和周波数と差周波数の倍数である相互変調生成が含まれます。IMDレベルは、相互変調生成の電力と目的の出力

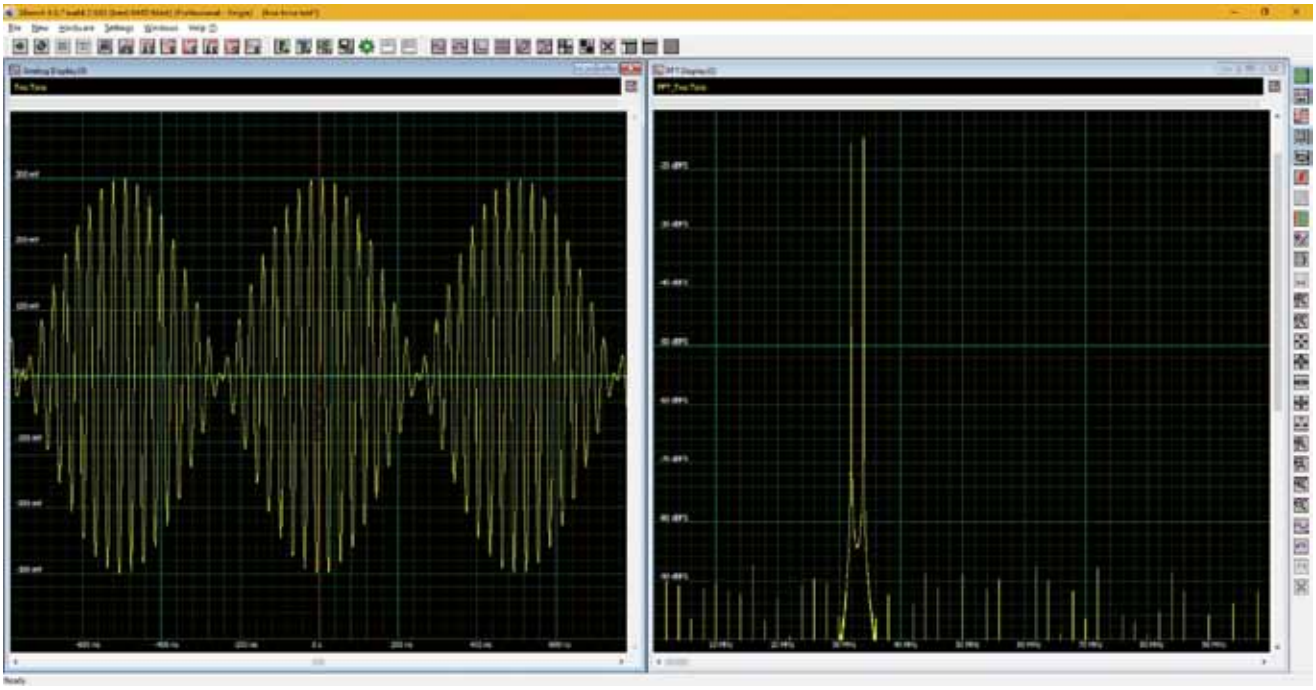


Figure 20: IMDテストでは、32MHz と 34MHz の 2つの正弦波が合計されます。デバイスの出力スペクトルには、元の 2つの周波数と、2MHz 間隔の高調波および相互変調積が表示されます

信号の変調生成が含まれます。IMDレベルは、相互変調生成の電力と目的の出力信号の電力の比で、デシベル（dB）で表されます。IMDレベルが低いほど、デバイスのパフォーマンスは向上します。Figure 20は、ツートーンテストの例を示しています。

使用される2つのトーンは32MHzと34MHzです。左側に示されている合計信号は、周期的に干渉し合い、互いに強め合い、結果としてビート音になります。スペクトルには、2つの元の周波数と、すべての混合および相互変調信号が表示されます。これらは2MHzの差周波数で間隔が空いています。

マルチトーン波形

マルチトーン正弦波は、アナログ-デジタルコンバータ（ADC）のパフォーマンスを迅速に評価するために使用できます。下の表は、15個の正弦波を加算したテスト波形の成分の周波数と振幅を示しています。スペクトラムアナライザで表示される結果の波形をFigure 21に示します。

Frequency (MHz)	Amplitude (%)	Relative Amplitude (dB)
34.0	50	0
33.5	25	-6
33.0	12.5	-12
32.5	6.25	-18
32.0	3.125	-24
31.5	1.562548	-30
31.0	0.781274	-36
30.5	0.390637	-42
30.0	0.195318	-48
29.5	0.097659	-54
29.0	0.04830	-60
28.5	0.024415	-66
28.0	0.012207	-72
27.5	0.006104	-78
27.0	0.003052	-84

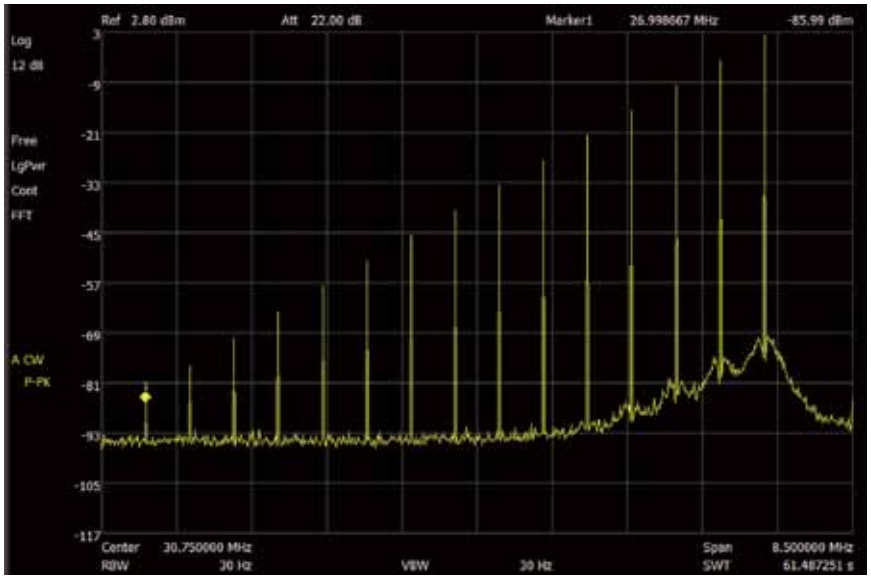


Figure 21：スペクトラムアナライザで表示される15トーンの波形

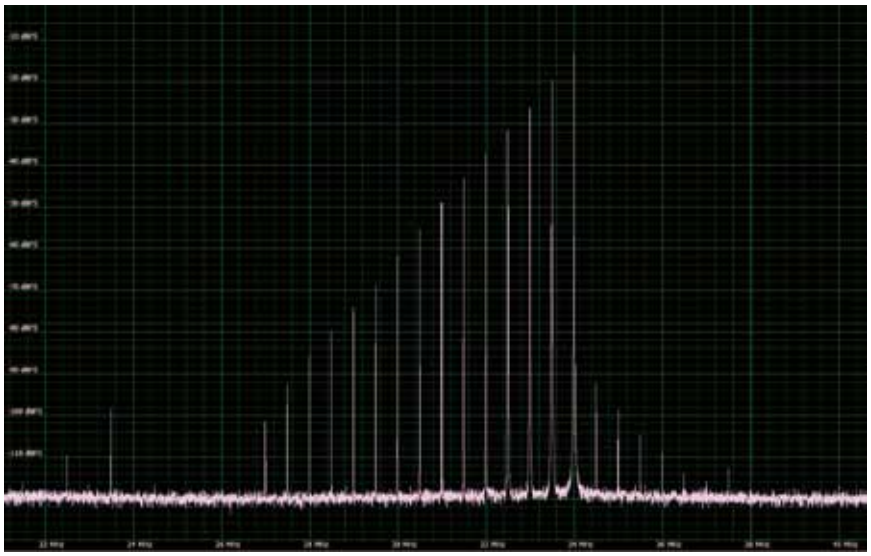


Figure 22：15トーンのテスト波に対するデジタイザーの応答

スペクトラムアナライザはトーン振幅の直線性を示します。

デジタイザに適用すると、Figure 22の結果の表示はデジタイザのダイナミックレンジと直線性を示します。

デジタイザの応答は、トーン振幅の読み取りに優れています。一部の相互変調成分は、-90dBを大きく下回る振幅で現れます。このシンプルな波形から、デジタイザの応答の品質の高さを把握できます。

マルチトーンマルチ出力アプリケーション

これまで説明したマルチトーン信号は合計され、1つのチャネルから出力されます。

また、複数のチャネルから単一の正弦波を出力することもできます。フェーズドアレイシステムの駆動などのアプリケーションでは、個々の正弦波の振幅と位相の制御を使用して、アンテナまたはトランスデューサーのアレイの波面を操作またはフォーカスします。

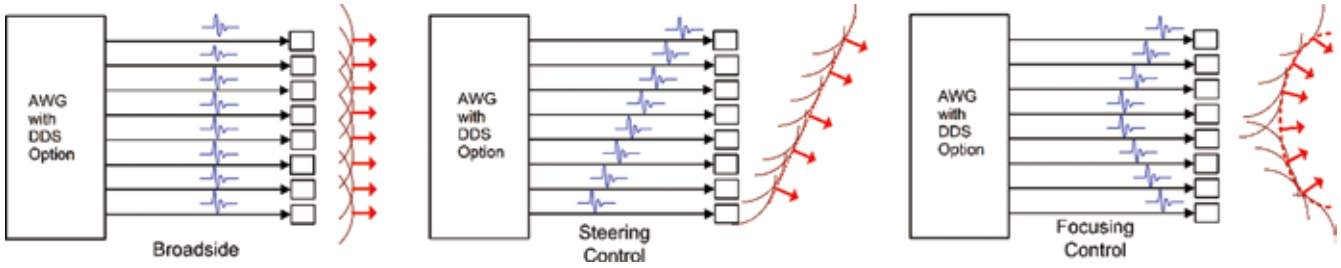


Figure 23 : アンテナまたはトランスデューサアレイを駆動する個々のソースの位相と振幅を制御することで、放射された波面を操縦または集束させることができる

Figure 23は、この機能を簡略化した図を示しています。

波形が同じ位相でエミッターに到達した場合、波面はフラットまたはブロードサイド形状になります。波形をエミッター全体で直線的に位相遅延させることにより、波面を操縦できます。最後に、波形の位相を調整して中心近くのエミッターが最初に到着し、外側のエミッターが後で駆動されるようにすることでアレイをフォーカスできます。

Figure 24は、ブロードサイドまたは無指向性アライメントの場合と、45度下方向に操縦された指向性波面の場合の8素子フェーズドアレイを駆動するための波形を示しています。

赤いカーソルは、両方のインスタンスの素子0波形の3番目のサイクルのピークを示します。無指向インスタンスのすべての波形は同じ位相を持ち、カーソルはすべての素子波形の 3番目のサイクルのピークに配置されます。指向波形素子では、連続する各素子の素子位相が徐々に遅れていることがわかります。

この例では8チャネルを使用しますが、これはSTAR-HUBオプションと同期された2つの 4チャネル M4i.6622-x8 AWGを使用して実現できます。最大8枚のPCIeカードを使用するシステム、合計32チャネルのシステムも同じ方法で構築できます。LXI/Ethernetベースの機器が望ましい場合は、最大24チャネルのDNシリーズジェネレータNETBOXユニットも利用できます。

直交変調

DDSオプションでは、位相変調信号の生成も可能です。直交変調は位相変調に基づいており、データ通信システムで使用されます。直交位相シフトキーイング（QPSK）は、送信シンボルごとに2ビットをエンコードする変調方式です。これは、同相（I）データストリームと直交（Q）データストリームと呼ばれる 2つのバイナリ位相シフトキーイング（BPSK）データストリームを作成することによって生成されます。I信号とQ信号は直交して合計され、QPSK信号が生成されます。これらの信号は、180度の差がある2つのバイナリ状態をエンコードします。DDSオプションでは、Figure 25 に示すようにこれらの波

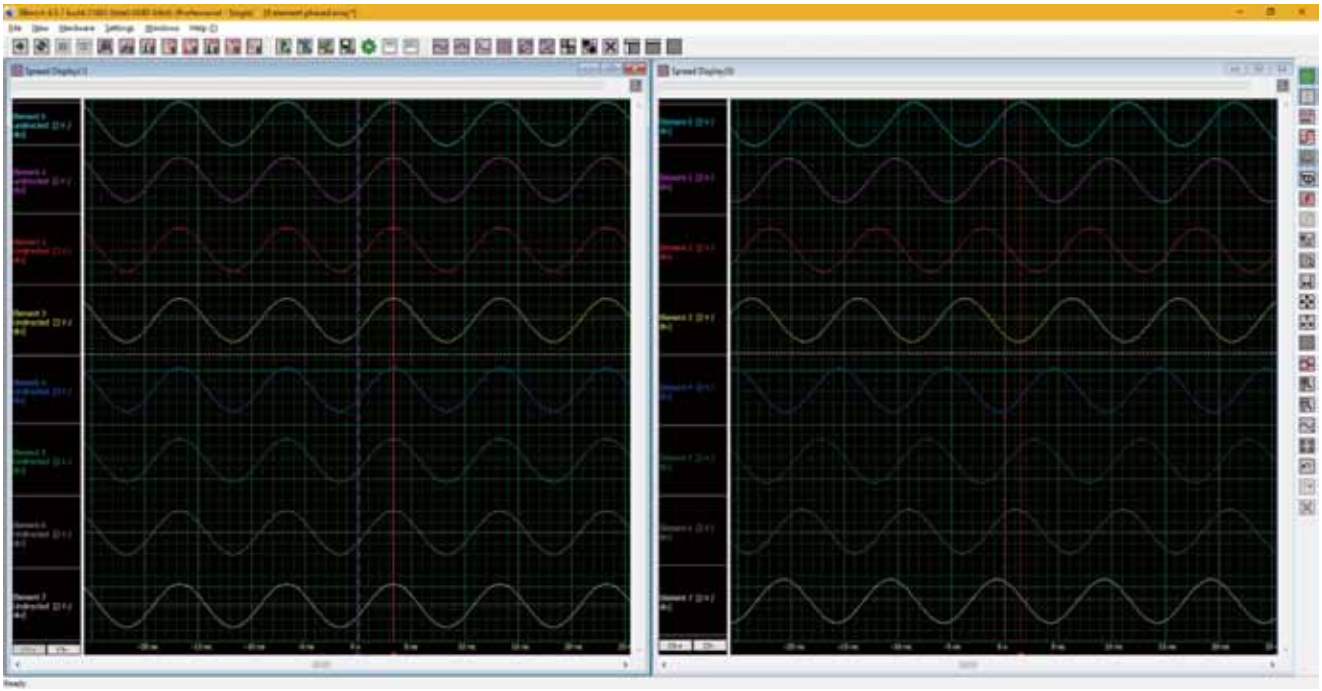


Figure 24:8素子のフェーズドアレイのゼロ位相シフトによる無指向（左）とプログレッシブ位相シフトによる指向性アライメント（右）の信号位相を比較します

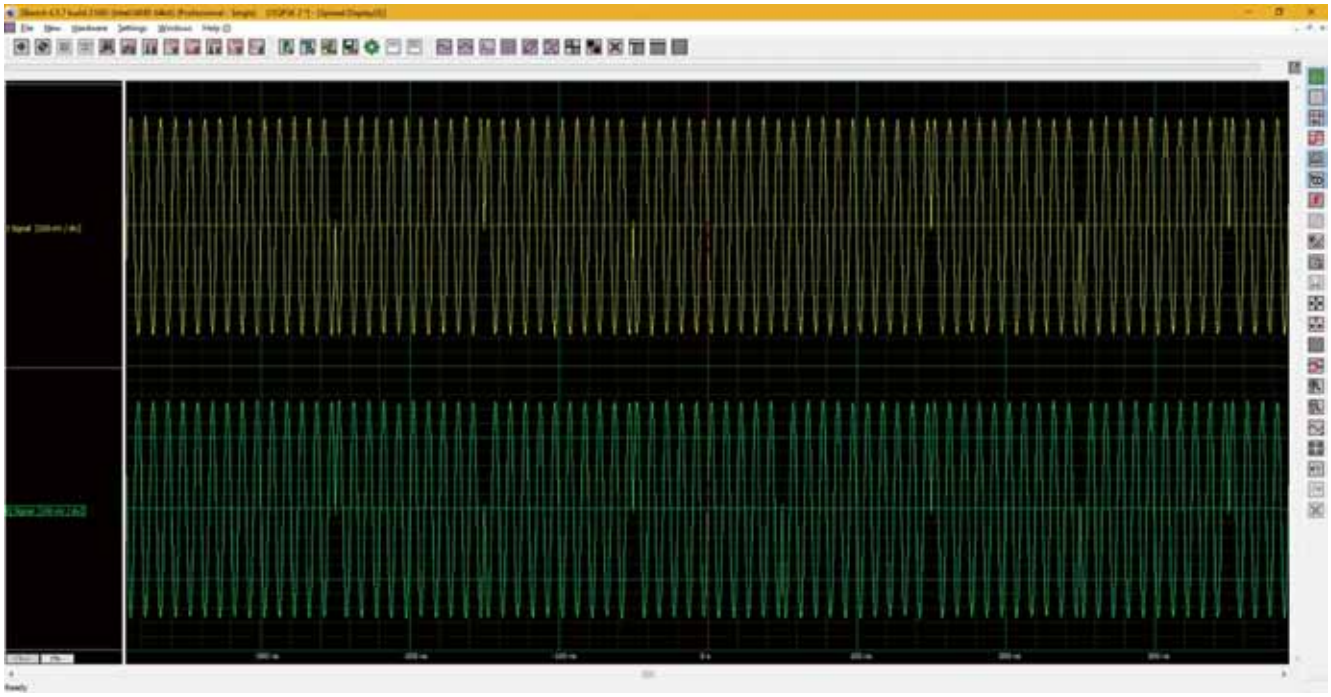


Figure 25：QPSK エンコード信号の I 成分と Q 成分が結合され、4 つのデータ状態を持つ信号が生成されます

形を簡単に生成できます。

2つのBPSK信号はそれぞれ2つのバイナリ状態のみを持ち、変調器で結合されて4状態の QPSK信号を形成します。この例では、バイナリデータの送信された1および0状態に対応するプラスおよびマイナス180度の位相シフトを使用して、100MHzのキャリアを5MHzのシンボルレートで位相変調します。

まとめ

AWG用のSpectrum製 DDSオプションは、AWG機能に、生成と制御が簡単な複数の正弦波出力を追加することで、AWG製品の汎用性を高めます。DDSオプションのシンプルなテキストベースのコマンドは、正弦波の振幅、周波数、位相を制御し、より応答性の高い制御のために、より大規模なサンプル値ベースのAWG波形を正弦波に置き換えます。DDSオプションは、通信、医療、画像処理、産業システムなど、高速周波数切り替え、細かい周波数、位相、振幅制御を使用するアプリケーションに適用できます。

```
// I Signal on Channel 0, CORE 0
// Q Signal on Channel 1, CORE 20


PHASE_BEHAVIOR = SHIFT
CORE0_FREQ     = 100 MHz
CORE0_AMP      = 1.0

CORE20_FREQ    = 100 MHz
CORE20_AMP     = 1.0

// 10 MHz Sampling Rate by using the Timer
TRG_SRC        = TIMER
TRG_TIMER      = 0.2µs

bool iq_data[64][2];    // 64 QSPK Symbols with 1 I and 1 Q Bit
                        // n'th I-Bit at Index [n][0], Q-Bit at Index [n][1]


// Fill IQ Data with 128 Bit of a Pseudorandom Binary Sequence
iq_data = generate_PRBS_data(128)

for (i = 0; i < 64; i++)
{
    CORE0_PHASE    = 0°  + iq_data[i][0] * 180°
    CORE20_PHASE   = 90° + iq_data[i][1] * 180°
    CMD            = EXECUTE_AT_TRIG
}
```

Code3：直交変調により I/Q 信号を生成する場合のコマンド例

M4i.6622-x8
4ch, 625MSPS AWG

リファレンスドキュメント：
Spectrum Instruments社
Application Note：Using DDS Mode in different Applications

SPECTRUM
INSTRUMENTATION

ADVANCED

２枚のデジタイザを使用して20GSPSを実現する方法

はじめに

この記事で実現する内容は以下のとおりです。

- ２枚の10GSPS 12bit デジタイザ(ADQ35) を時間インターリーブして20GSPSを実現
- DC ～ 2.5GHzの入力周波数範囲で4ps未満のタイミング誤差を実現
- 各デジタイザはPCIe経由で12.8GB/sでデータをGPUにストリーミングする

デモセットアップのブロック図をFigure 26、ADQ35を２枚実装したテストシステムの写真をFigure 27に示します。

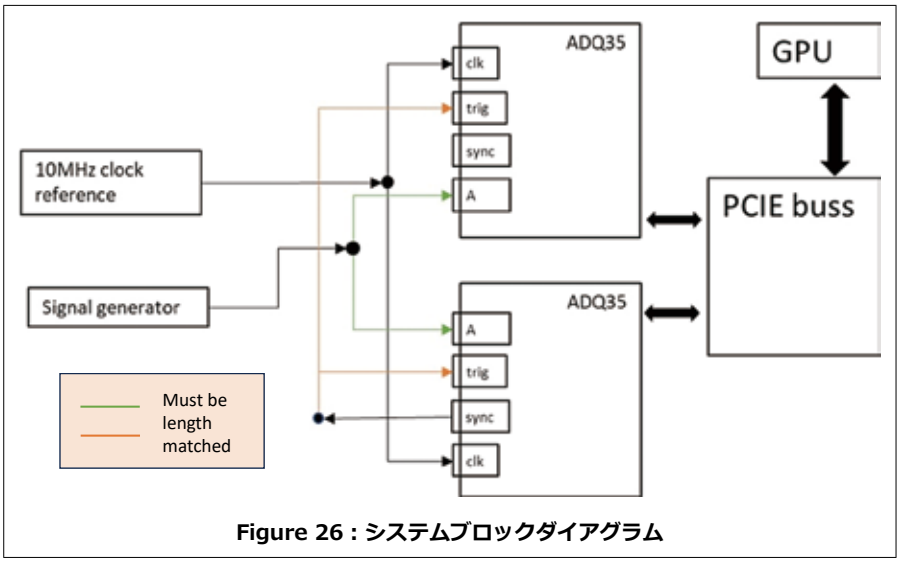


Figure 26：システムブロックダイアグラム

アプリケーション

２枚のデジタイザで、分割されたアナログ入力信号を時間インターリーブ方式でサンプリングします。これは、一方のデジタイザで、入力リファレンスクロックを遅延/スキューするオンボード機能を使用することで実現されます。

取得したデータの後処理はGPUで実行されます。信号処理のベクトル最適化されたCPU処理の実装は計算上十分かもしれませんが、比較的高い帯域幅を処理するため処理タスクにはGPUを選択します。

Figure 27：２枚のADQ35をインストールしたテストPC

データストリーミング

システムを選択する際に考慮すべき主な点は、PCIeとGPUメモリ帯域幅です。PCIe帯域幅は、CPUのPCIeレーンの数とマザーボード上のレーンのルーティング方法に分けられます。

ADQ35は、x16レーンのGen3 PCIe

インターフェイスを介して最大14GB/sでデータをストリーミングできます。

より新しいGPUには、x16レーンのGen4 PCIeインターフェイスが搭載されています。Gen4へのアップグレードにより帯域幅が2倍に増加します。

これを考慮すると、システムにはx16レーンのGen4 PCIeポートが1つとx16レーンのGen3 PCIeポートが2つ必要です。

ハードウェアの詳細

使用されるシステムには以下のハードウェアが含まれます:

- AMD EPYC 7282 CPU、128 個 のGen4 PCIeレーンを提供
- SUPERMICRO H12SSL-Iマザーボード、5つのx16レーン Gen4 PCIeポートを提供
- NVIDIA A2000 GPU、288GB/s VRAM帯域幅とGen4 x16 PCIeインターフェイスを提供

2枚のADQ35デジタイザへの信号入力

デジタイザ間のサンプリングクロックの位相差を制御できるように、分割された10MHzクロックリファレンスが各デジタイザのclkコネクタに接続されます。クロックの位相は各ADQ35に組み込まれた微調整遅延ラインによって調整されます。

単一トリガーコマンドを両方のボードに配布できるように、ソフトウェアコマンド SWTrig() で同期コネクタにパルス出力を生成します。一方のデジタイザの同期コネクタからの出力は分割され各デジタイザのトリガーコネクタに接続されます。次に、各デジタイザのトリガーポートは、トリガーイベントを10MHzクロックリファレンスに同期するように構成されます。これにより、両方のカードで同時にデータの取得を開始できます。

ケーブル接続をFigure 28に示します。

コネクタとその設定方法については、『ADQ3シリーズデジタイザ - ユーザーガイド』で説明されています。

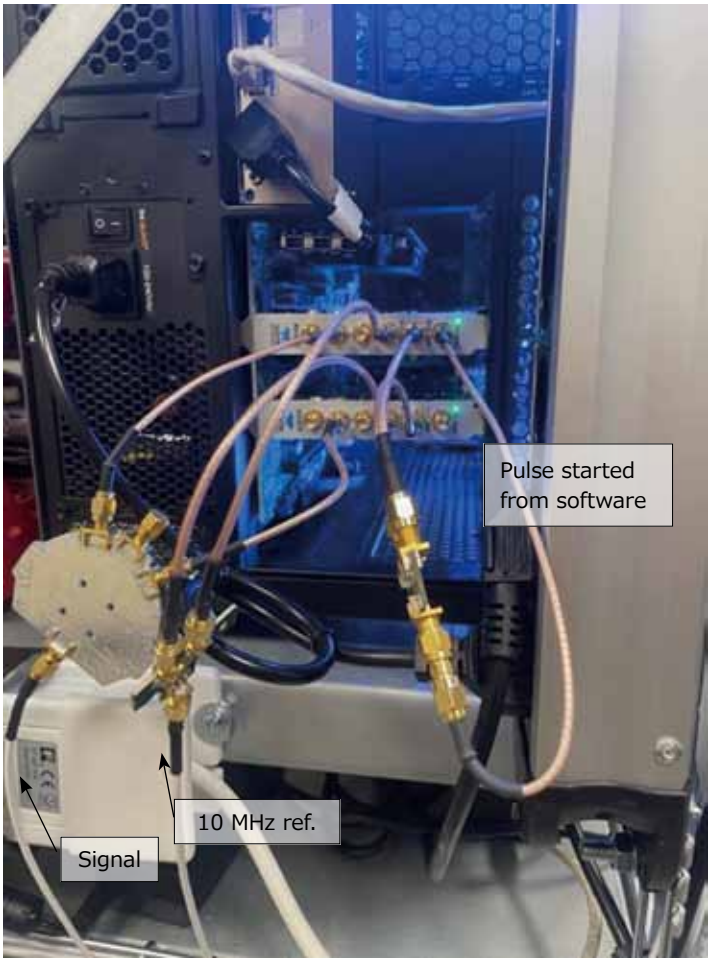


Figure 28：デジタイザへのケーブル接続

ソフトウェアの詳細

- オペレーティングシステムは Ubuntu 20.04.6 LTS
- デジタイザーファームウェアと ADQAPI のリリースパッケージ 2024.1
- NVIDIA ドライバー 550.54.14、CUDA 12.4

サンプルコード

サンプルソフトウェアである calibrate.py と interleave.py は、Teledyne SP Devices社リリースパッケージで提供される2つのPythonサンプルスクリプトに基づいています。これらは、examples/python/data_readout および examples/python/data_transfer_gpu_nvidia フォルダーにあります。

data_readoutスクリプトは最も基本的な例です。APIの簡略化されたdata_readout機能を使用してデジタイザからデータを取得します。詳細については、ExampleのREADMEファイルを参照してください。

スクリプトdata_transfer_gpu_nvidiaは、低レベルのAPI機能を使用してデジタイザから NVIDIA GPUにデータを移動します。詳細が記載された独自のREADMEもあります。

両方のスクリプトで必要な主な変更は次のとおりです。

- 2つのデジタイザをサポートするように拡張します。
- デジタイザを構成して10MHzリファレンスクロックを使用します。
- Clock_system.delay_adjustment_enableを設定します。
- トリガーポートを使用してイベントを作成しレコードの取得を開始します。これらのイベントを10MHzリファレンスクロックに同期します。

・(1枚のカードの場合) パルスジェネレーターを構成して、ソフトウェアトリガーで開始し同期コネクタでパルスを出力します。
・(1枚のカードの場合) clock_system.delay_adjustmentを構成します。

■ calibrate.py

2枚のカードが互いに180度位相がずれた信号をサンプリングするclock_system.delay_adjustmentの値を見つけるための1つの方法は、既知の正弦信号を提供することです。両方のデジタイザからサンプリングされたデータに正弦関数を当てはめます。これらの関数から必要な遅延を計算します。これは、標準サンプル data_readout.pyに基づくcalibrate.pyで実装されています。操作のフローチャートをFigure 29 に示します。

■ interleave.py

このスクリプトはデータ転送を使用して、2つのデジタイザからGPUのメモリに直接データを移動します。calibrate.pyと異なるのは、SWTrig()呼び出しがパルスジェネレータをすぐに起動しないことです。代わりに、同期コネクタに出力を作成するパルスジェネレータを起動する周期イベントソースを起動します。これは、トリガー信号がアクティブになったときに両方のカードが完全に構成され、準備完了になっている必要があるためです。

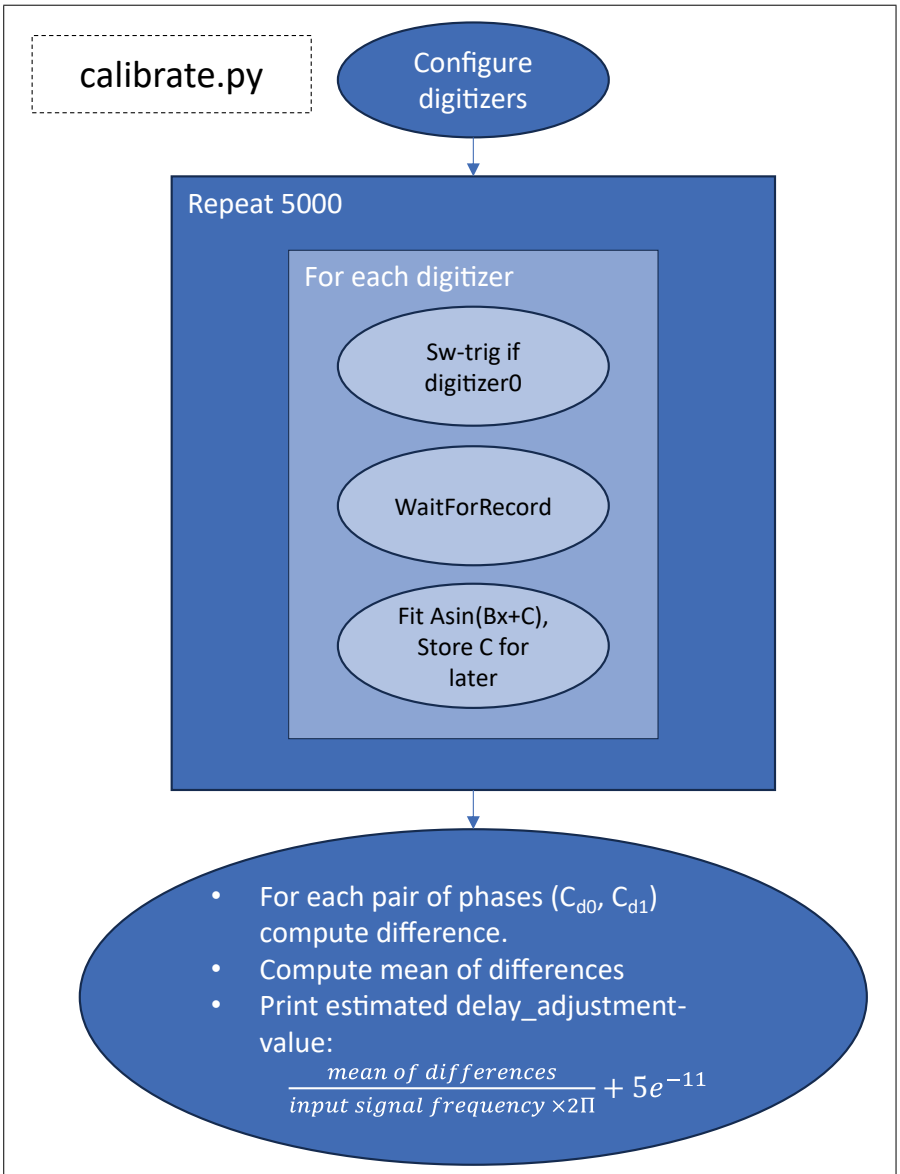


Figure 29：calibrate.py スクリプトのフローチャート

追加の測定機器

使用した追加機器:
・Stanford Research Systems CG635 (10MHzクロックリファレンス用)
・Agilent Technologies N5181A (信号ソース用)

結果

■ タイムドメイン

Figure 30 は delay_adjustment-calibrationの前後のサンプリングされた信号を示しています。

Figure 31は測定された時間誤差を示しています。つまり、予想されるサンプリング周期の半分 (50ps) からの偏差、異なる入力周波数で測定されたカード間の時間差です。
・2つのチャネル間の時間誤差が4ps未満で20GSPSに達します。
・GPUへのストリーミングと、最大25.6GB/sの速度でのcupyでのデータのインターリーブが測定されました。

■ 周波数ドメイン

2GHzでのテストトーンのFFTをFigure 32に示します。FFTには、2つのADQ35のインターリーブによるインターリーブスプリアスが含まれています。インターリーブスプリアスは、時間 (位相)、振幅、およびオフセットの不一致によって発生します。Figure 33には、オフセットと振幅のエラーが補正された同じ信号が含まれています。これは、時間誤差がインターリーブスプリアスに及ぼす影響です。

リファレンスドキュメント：
Teledyne SP Devices社
Application Note：Interleaving 2 pieces of ADQ35 to get 20 GSPS and stream data to GPU

TELEDYNE SP DEVICES
Everywhereyoulook™

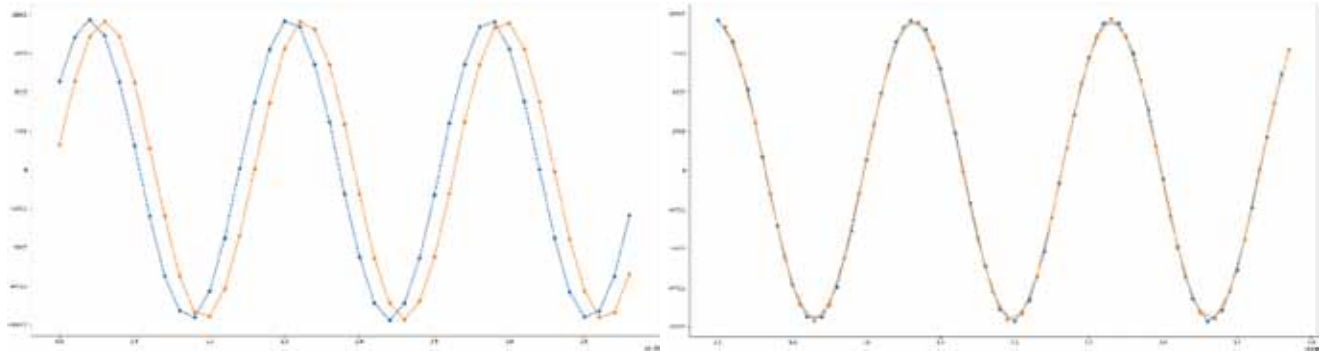


Figure 30：オレンジの波形はデジタイザ 0 からのサンプルポイント、青はデジタイザ 1 からのサンプルポイントです。Y 軸は信号振幅、X 軸は秒単位の時間軸です。左のグラフは delay_adjustment のキャリブレーション前です。右はキャリブレーション後です

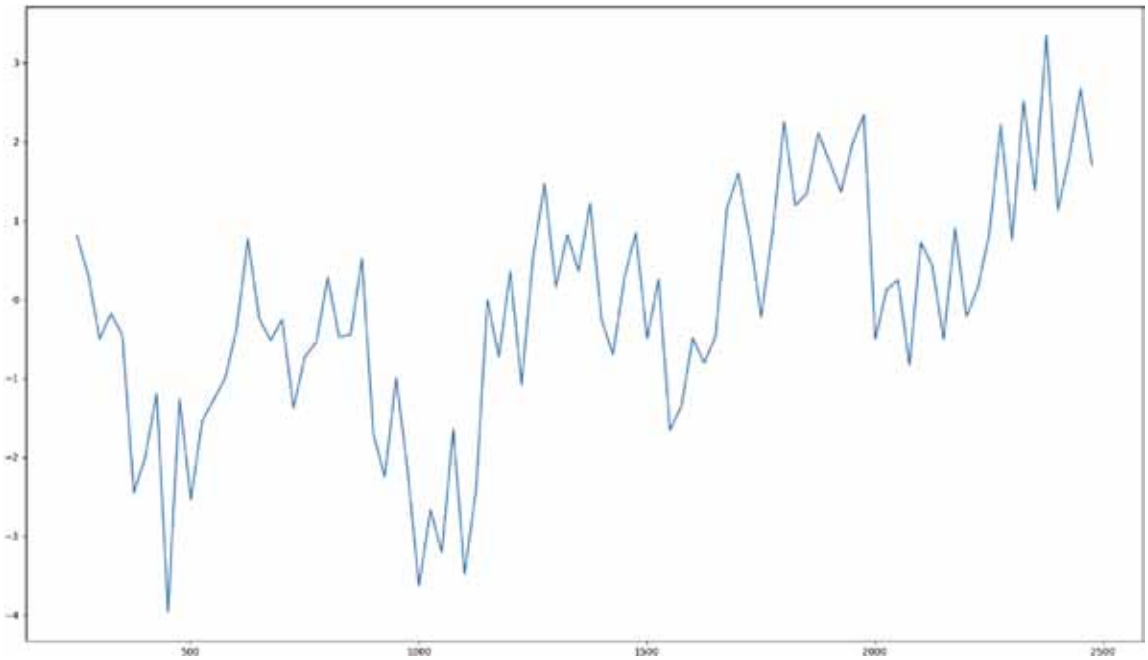


Figure 31：X 軸は MHz 単位のテストトーン周波数、Y 軸は ps 単位の測定時間誤差です

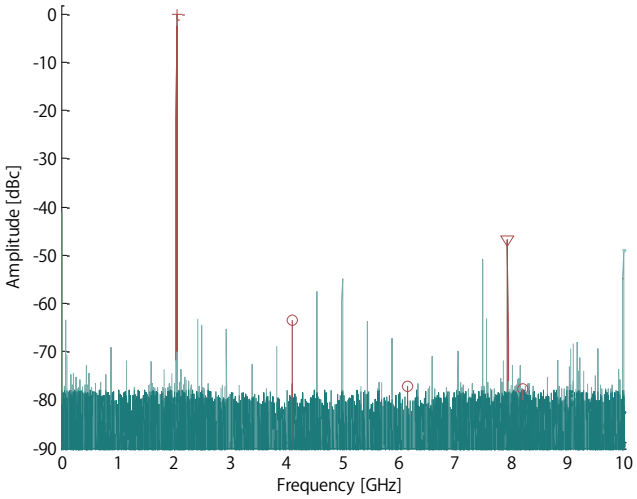


Figure 32：10GSPS でサンプリングされた 2GHz 信号の FFT

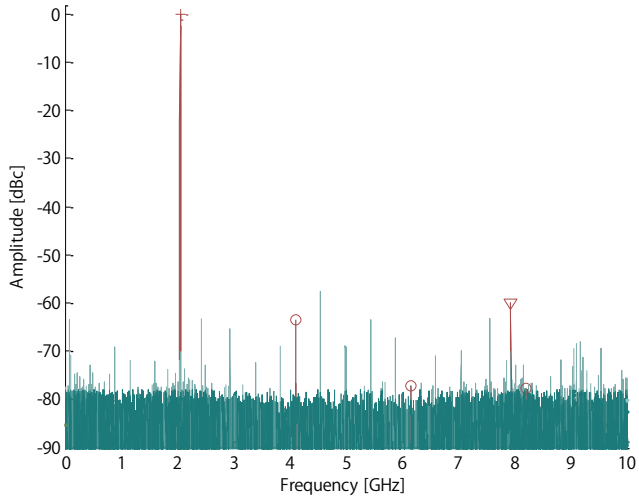


Figure 33：振幅誤差補正後の FFT

APPLICATION

分散型光ファイバーセンシング（DFOS）

はじめに

光ファイバーは、データ伝送ネットワークにおける古いツイストペア銅線接続に代わる、私たちの世界をつなげる最新の通信網として一般的に認識されています。しかし、光ファイバーには独自の物理的特性があることから、さまざまな分散型センシングアプリケーションに適しており、実際にその用途が想定されています。光ファイバーの経済性と、長い距離にわたって局所的な測定を行えるという本来の能力が組み合わさったアプリケーションが、その恩恵を受けています。計測方法は、主に3つのセンシング手法に分けられます。

- 分散温度センシング（DTS）
- 分散音響センシング（DAS）
- 分散歪みセンシング（DSS）

光ファイバーケーブルは、集中した光（レーザー、VCSEL、またはLED光源から供給）を適度な減衰で長距離伝送する導波管として機能します。信号を伝送する場合、ファイバーは物理的に小型・軽量であることの利点があります。さらに、電磁干渉の影響を受けません。

分散型光ファイバーセンシング（DFOS）の基礎

科学的には、製造中にファイバーに不純物が生じると、ファイバーを通過する光に低レベルの散乱が生じることが分かっています。散乱には3つの種類（Figure 34）があり、その物理的特徴と検出に必要な方法によって次のように説明されます。

光子の散乱はランダムな確率プロセスです。散乱は弾性的（レイリー散乱など）であり、散乱された光子は同じ波長を維持することを意味します。非弾性ブリルアン散乱およびラマン散乱では、散乱された光子は原子レベルでのエネルギーレベル遷移と連動して、かなりの波長シフトを経験します。波長の変化は、波長が減少する場合はストークスまたは反ストークスとして示されます。

分散型光ファイバーセンシング（Figure 35）の基本操作では、コヒーレント光の繰り返しパルスで光ファイバーを照射する必要があります。使用する機器はインタロゲータ（Interrogator）と呼ばれます。各パルスはファイバーに沿って進むにつれて伝送遅延が発生します。ファイバー全体の個々の位置で、大きく減衰した後方散乱信号（Backscatter）が発生し、その飛行時間によって決まる既知の時間（伝搬遅延）で送出元に戻ります。適切な光結合デジタル化システムで戻り経路信号をサンプリングすることで、温度、機械的歪み、さらに音響エネルギーなどの局所的な物理的特性を検出でき、さまざまな経済的計測アプリケーションを提供します。

デジタイザを選択する際に考慮すべき事項

DFOSアプリケーションの最も重要な側面は、次の点を理解することです：

1. 検出される特定のタイプに関係なく、後方散乱（Backscatter）は、低レベル信号で構成されているため検出するのに高いダイナミックレンジが必要です（ADC分解能：10ビット以上）。
2. 光ファイバー計測に使用される短い光パルス（通常5ns ～ 20μs）は、それらを効果的にキャプチャするために高速デジタイザが不可欠であることを意味します。したがって、数ギガサンプル（GSPS）のサンプルレートが必要になる場合があります。
3. 適切な測定パルス幅の選択は、ファイバーの長さを考慮しパルスの飛行時間を評価することによって決定されます。
4. DFOSシステム設計者は、選択したデジタイザが提供する高パフォーマンスを補完するため、適切な光フロントエンドパルス生成および検出回路の設計にも注意を払う必要があります。

リファレンスドキュメント：
Teledyne SP Devices社
Application Note：Interleaving 2 pieces of ADQ35 to get 20 GSPS and stream data to GPU

TELEDYNE SP DEVICES
Everywhereyoulook™

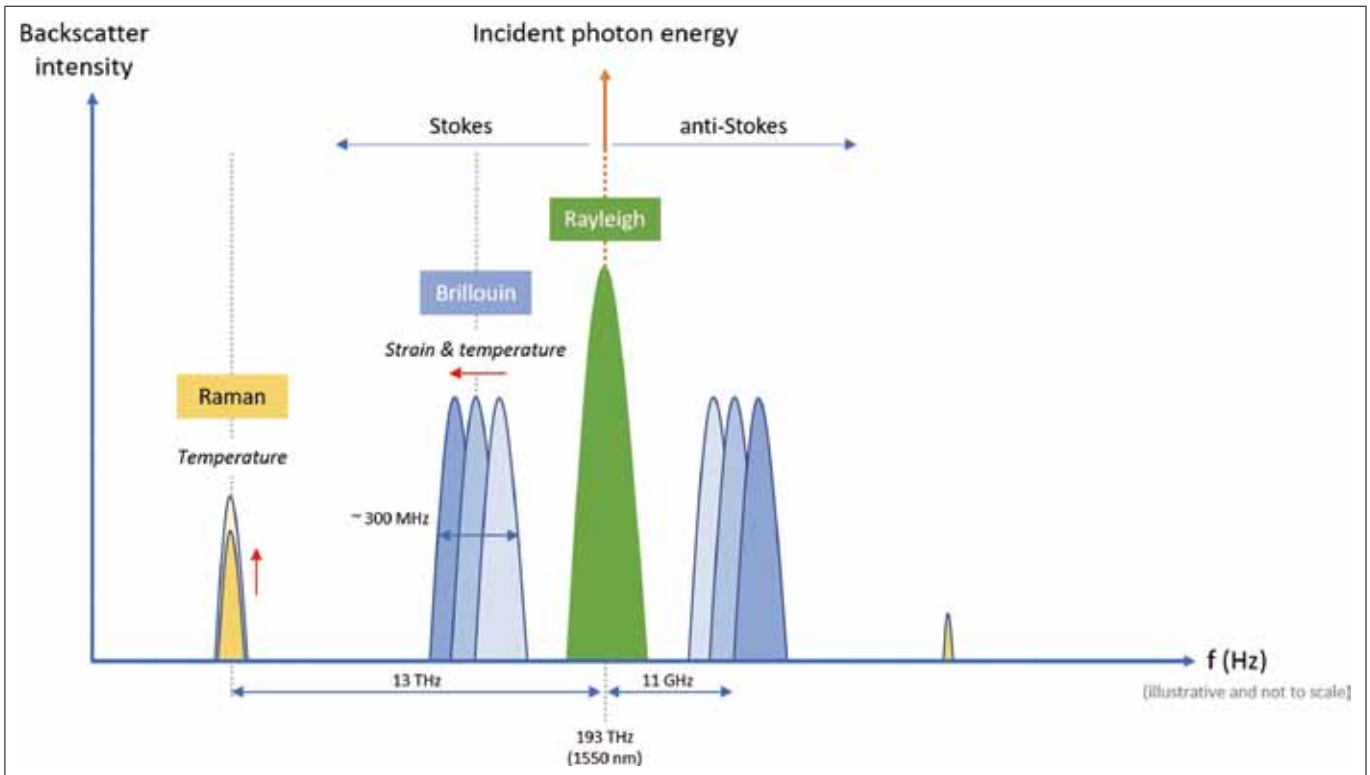


Figure 34：相対的な周波数 / 強度特性を示す３種類の光子散乱

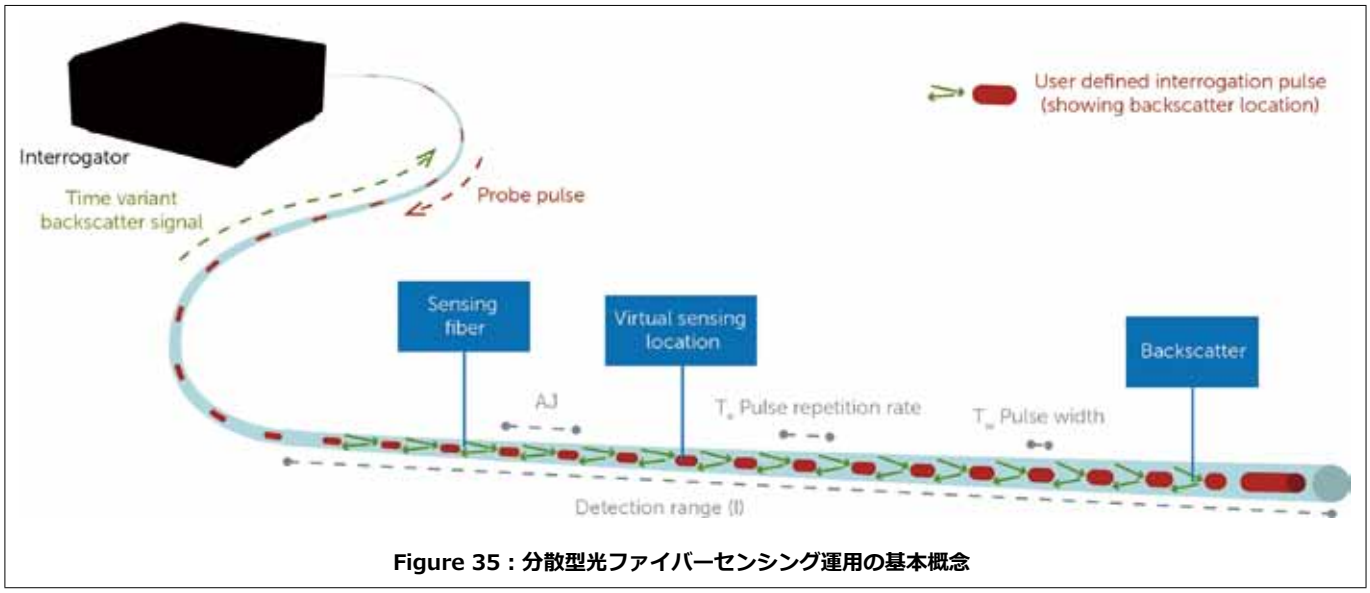


Figure 35：分散型光ファイバーセンシング運用の基本概念

新製品ピックアップ

ここでは今年の新製品をピックアップして紹介します。

- SPECTRUM社製　M5i.6357-x16　10GSPS高速D/Aボード
- Alpha Data社製　ADM-XRC-9R1-B　RFSoC搭載FPGAボード
- Teledyne SP Devices社製　ADQ35-WB　10GSPS広帯域A/Dボード
- Dagscribe社製　DDR7400　400Gbps イーサネットデータレコーダ
- Dagscribe社製　RDR4050　MIL規格対応 耐環境イーサネットレコーダ

SPECTRUM社製　M5i.6357-x16 10GSPS高速D/Aボード

M5i.6357-x16 仕様	
サンプリングレート	10GSPS, 5GSPS
出力チャンネル数	1CH, 2CH
分解能	16bit
帯域幅	2.5GHz
フォームファクタ	PCI Express Gen3, x16 レーン
温度範囲	0 ～ +50℃
OS	Windows, Linux

M5i.6357-x16 は、10GS/s の D/A コンバータを搭載した任意波形ジェネレータ（AWG）ボードです。 サンプリングレートは1chで10GS/s、2chで5GS/sをサポートし、16bitの分解能を備えているため速度と分解能の両方で最高のパフォーマンスを提供します。また、オプションのStar-Hubを使用することで最大16chを同期することができます。超音波、レーダ、LiDAR、無線通信、レーザ、加速器、非破壊検査などのアプリケーションに適合します。Windows又はLinux対応。

Alpha Data社製　ADM-XRC-9R1-B RFSoC搭載FPGAボード

ADM-XRC-9R1-B 仕様	
A/D コンバータ	5GSPS, 14bit, 8ch
D/A コンバータ	10GSPS, 14bit, 8ch
FPGA	Zynq UltraScale+: XCZU47DR-2 または XCZU48DR-2
ARM コア	4x ARM Cortex-A53 MPCore, 2x ARM Cortex-R5 MPCore
フォームファクタ	XMC
温度範囲	-40 ～ +70℃
OS	Windows, Linux

ADM-XRC-9R1-B　は、AMD Zync UltraScale+ RFSoC（5GHz AD & 10GHz DA内蔵）を搭載した XMCタイプの FPGA ボードです。8chの5GSPS ADコンバータ及び8chの10GSPS DA コンバータ をRFSoCチップに内蔵しています。FPGA はユーザプログラム可能で、ユーザロジックを実装することが可能です。耐環境仕様としてコンダクションクールをサポートしています。ソフトウェアドライバは Windows, Linuxが用意されており、レーダ・画像処理・ソフトウェア無線機・5Gアプリケーションに最適です。

Teledyne SP Devices社製　ADQ35-WB 10GSPS広帯域A/Dボード

ADQ35-WB 仕様	
サンプリングレート	10GSPS, 5GSPS
チャンネル数	1DH, 2CH
分解能	12bit
帯域幅	2.5GHz
メモリ	8GB SDRAM
フォームファクタ	PCI Express Gen3, x16 レーン
温度範囲	0 ～ +55℃
OS	Windows, Linux

ADQ35-WBは、10GHz 12bit 1ch（又は5GHz 12bit 2ch）のサンプリングレートで最大9GHzのアナログ入力帯域幅を備えた広帯域の高速A/Dボードです。Kintex UltraScale FPGA を搭載しており、カスタムのリアルタイムデジタル信号処理（DSP）に十分なリソースを提供します。最大14 Gbyte/s の速度でピアツーピアストリーミングをサポートし、GPU、CPU またはSSDへの効率的なデータ転送を可能にします。LIDAR、ビーム位置モニター、高エネルギー物理科学の用途に適しています。

Daqscribe社製　DDR7400 400Gbps イーサネットデータレコーダ

DDR7400は、400GbEをサポートしたイーサネットインタフェースのデータレコーダです。記録容量は最大360TBを搭載する事ができます。入出力インタフェースはリンクオプションでSFP+(10GbE)からQSFPDD (400GbE)の光ポートを選択することができます。ストレージはNVMeタイプのSSDを採用しており、30TB ～ 360TBまで選択することができます。オプションでFPGAデータ処理を追加することができます。100% イーサネットキャプチャ、記録/再生をサポート。19インチラックマウントタイプでOSはLinuxを搭載しています。レーダや広帯域無線信号の記録・再生アプリケーションに最適です。

DDR7400 仕様	
インタフェース	400Gb, 200Gb, 100Gb, 40Gb, 25Gb, 10Gb Ethernet
ストレージ容量	360TB
タイプ	1U ラックマウント
温度範囲	0 ～ +35℃
OS	Linux

Daqscribe社製　RDR4050 MIL規格対応 耐環境イーサネットレコーダ

RDR4050は、MIL規格に対応した耐環境仕様のイーサネットデータレコーダです。記録容量は最大120TBを搭載する事ができます。入出力インタフェースはリンクオプションでSFP+(10GbE)からQSFP+(40GbE)の光ポートを選択することができます。ストレージはNVMeタイプのSSDを採用しており、7TB ～ 120TBまで選択することができます。

データオフロードオプションにより高速データ転送が可能です。温度範囲は-40 ～ +60℃をサポートしており、MIL-STD-810に準拠しています。航空機や回転翼に搭載してご利用いただくことができます。 レーダや広帯域無線信号の記録・再生アプリケーションに最適です。

RDR4050 仕様	
インタフェース	50Gb, 25Gb, 10Gb Ethernet
ストレージ容量	120TB
タイプ	耐環境ボックス
温度範囲	-40 ～ +60℃
OS	Linux

地球温暖化を考える３

ドナルド・トランプ氏がアメリカの第４７代大統領になってから、パリ協定から離脱してしまいました。
世界の二酸化炭素排出量の約13%を排出しているアメリカがパリ協定から離脱したことで「2050年カーボンニュートラル」の実現は困難になるでしょう。
少なくともトランプ氏が大統領である任期中は13%が減少方向になることは無さそうですね。
このアメリカの決定を受けて中国も離脱しないか心配です。
地球温暖化は現在も進行していますが、少しでもその上昇を抑える為に私達は日々の生活で心がける必要があります。つい忘れてしまいがちですが、もう一度日常を見直してみましょう。

- 冷房を１℃高く、暖房を１℃低く設定する
- 長時間停車時はエンジンを切る
- 炊飯器の保温を止める
- 風呂の残り湯を洗濯に使う
- 家族が同じ部屋で過ごす
- TVの視聴時間を１時間減らす

少し不便になるかもしれませんが、昔の人は不便な生活でも知恵を使って楽しく暮らしていたのです。

展示会のご案内

DSEI Japan
日時：2025年5月21～23日
場所：幕張メッセ
皆様のご来場をお待ちしています。

受託開発

弊社ではソフトウェア / ハードウェア /FPGA の受託開発も承っております。お困りの事がございましたらお気軽にご相談ください。
✉ sales@mish.co.jp

おわりに

テックジャーナルでは、これからも出来る限りお客様に有効となる情報を提供していきたいと思います。今後ともどうぞよろしくお願いいたします。

『ミッシュ・テックジャーナル』次回発行をお楽しみに！

NOVO SPACE

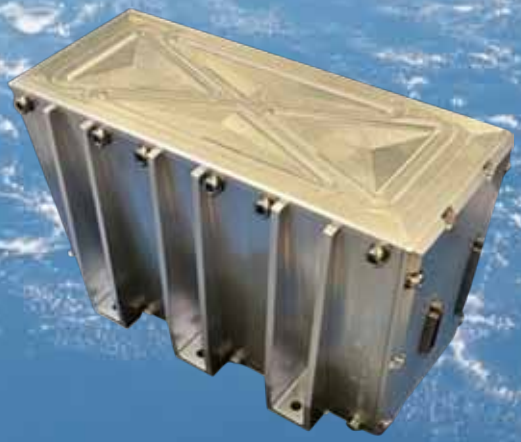
EASY
◆ Plug & Playで複数のH/Wを組み合わせることで開発期間を短縮します

POWERFUL
◆ 最先端の部品と高速インタフェースで高性能化を実現します

RELIABLE
◆ 共有バスを使用していないため障害が他の基板に波及しません

COMPACT
◆ 独自の特許技術で体積を最大35%節約します

OPEN STANDARD
◆ 標準規格(SpaceVPX)を採用しています



株式会社ミッシュインターナショナル
〒190-0004 東京都立川市柏町 4-56-1 TEL：042-538-7650
e-mail：sales@mish.co.jp URL：https://www.mish.co.jp