

はじめに

アルテラの FFT MegaCore® ファンクションは、内部でブロック浮動小数点 (BFP) 演算を使用して計算を実行します。BFP アーキテクチャは、固定小数点とフル浮動小数点アーキテクチャ間のトレードオフです。

浮動小数点演算を使用する FFT 演算器と異なり、ブロック浮動小数点 FFT には、データの指数部入力はありません。内部では、共通のスケーリング係数を持った整数としてデータを表現しています。FFT の各ステージの後、最大出力値が検出され、精度を向上させるために中間結果がスケーリングされます。指数 (exponent) は、スケーリングの実行に使用された右シフトまたは左シフト量を記録します。よって、コア出力に以下のスケーリングを行うことにより、最終的な FFT 演算結果が得られます。

$$\text{output} * 2^{-\text{exponent}}$$

例えば、 $\text{exponent} = -3$ の場合、出力 output を 3 ビット左シフトした値 $\text{output} * 2^3$ が、最終的な演算結果となります。

ブロック浮動 小数点

FFT コアの基数 2 または基数 4 エンジンを通過するたびに、加算および乗算演算を実行すると、データ・ビット幅が増加します。つまり、FFT 演算によるトータル・データ・ビット幅はバス数に比例して増加することになります。FFT/IFFT 演算のバス数は、変換ポイント数の対数によって決まります。3 ページの表 1 に、各変換ポイント数 N に対する指数出力の範囲を示します。

固定小数点アーキテクチャ FFT には、広いダイナミック・レンジを表す大きなビット幅の増加に対応するために、大規模な乗算器およびメモリ・ブロックが必要です。浮動小数点演算は、非常に高精度な演算結果が得られますが、その能力は浮動小数点乗算器や浮動小数点加算器など、デザインの複雑さを犠牲にして得られたものです。BFP 演算は、浮動小数点演算と固定小数点演算の利点を兼ね備えています。BFP 演算は、ハードウェア実装で同じビット数を持つ浮動小数点演算および固定小数点演算よりも優れた信号対ノイズ比 (SNR) およびダイナミック・レンジを提供します。

ブロック浮動小数点アーキテクチャ FFT では、各パスの基数 2 または基数 4 の計算は同じハードウェアを共有しており、データはメモリから読み込まれ、コア・エンジンを通過して、メモリに書き戻されます。加算および乗算演算からのキャリ・アウト・ビットがある場合は、次のパスに入る前に、各データ・サンプルは右シフトされます（これを“スケーリング”と呼びます）。シフトされるビット数は、データ・サンプルと前段で検出された最大データ・サンプルとの間のビット増加の差に基づきます。最大ビット増加は指数レジスタに記録されます。これにより、各データ・サンプルはすべて同じ指数値およびデータ・ビット幅を持ち、次のコア・エンジンに入ります。ビット増加に対応するためにより大きなエンジンを必要とせず、同じコア・エンジンを再利用できます。出力 SNR は、発生する右シフトのビット数および基数コア計算のステージに応じて異なります。つまり、信号対ノイズ比は入力データに依存し、SNR を計算するには入力信号が分かっている必要があります。

可能な指数値 の計算

FFT/IFFT の長さに応じて、基数エンジンを通過するパス数が計算でき、それによって指数の範囲も計算できます。可能な指数の範囲は、以下の式によって算出されます。

$P = \text{ceil}\{\log_4 N\}$ 、ここで N は変換長

$R = 0$ 、 $\log_2 N$ が偶数のとき、それ以下は $R=1$

シングル・エンジンの場合の出力範囲 = $(-3P+R, P+R-4)$

クワッド・エンジンの場合の出力範囲 = $(-3P+R+1, P+R-7)$

これらの式は、表 1 に示す値に変換されます。

シングル出力エンジン				クワッド出力エンジン			
N	P	MAX (2)	MIN (2)	N	P	MAX (2)	MIN (2)
64	3	-9	-1	64	3	-8	-4
128	4	-11	1	128	4	-10	-2
256	4	-12	0	256	4	-11	-3
512	5	-14	2	512	5	-13	-1
1024	5	-15	1	1024	5	-14	-2
2048	6	-17	3	2048	6	-16	0
4096	6	-18	2	4096	6	-17	-1
8192	7	-20	4	8192	7	-19	1
16384	7	-21	3	16384	7	-20	0

表 1 の注:

- (1) この表は、内部で発生する可能性のあるビット・シフト量である指数の範囲を示しています。IFFT の場合、出力を外部で N で除算する必要があります。このステップの後でさらに演算処理を実行する場合は、精度の低下を防ぐために、最後に N による除算を実行する方法が良いです。
- (2) MAX および MIN の値は、データがシフトされるべき回数を示します。負の値は左シフト、正の値は右シフトを示します。



IFFT 動作での N による除算の詳細は、「FFT MegaCore ファンクション・ユーザーガイド」の「Specifications」の章の式 2 を参照してください。

スケーリングの実装

スケーリング・アルゴリズムは、次のように実装してください。

1. フル・スケール・ダイナミック・レンジ・ストレージ・レジスタの長さを決定します。長さを得るには、データがシフトされる回数にデータ幅を加算します (表 1 に示す MAX 値)。例えば、16 ビット・データ、256 ポイント・クワッド出力 FFT/IFFT の場合、MAX = -11 および MIN = -3 です。この MAX 値は 11 ビットの左シフトを示します。したがって、フル・スケールのデータ幅は $16 + 11 = 27$ ビットです。
2. 指数出力に基づき、データ出力をビット幅に拡張したレジスタ内の適切な位置にマップします。上記の例では、FFT/IFFT からの 16 ビット出力データ [15..0] を、指数が -11 の場合は [26..11]、指数が -10 の場合は [25..10]、-9 の場合は [24..9] というようにマップします。

3. フル・スケール・レジスタ内で符号拡張を行います。

符号拡張付き出力データ (指数 -11 ~ -9) のスケーリングを示す、Verilog コードのサンプルを以下の例に示します。

```

case (exp)
  6'b110101 : //-11 Set data equal to MSBs
    begin
      full_range_real_out[26:0] <= {real_in[15:0],11'b0};
      full_range_imag_out[26:0] <= {imag_in[15:0],11'b0};
    end
  6'b110110 : //-10 Equals left shift by 10 with sign extension
    begin
      full_range_real_out[26] <= {real_in[15]};
      full_range_real_out[25:0] <= {real_in[15:0],10'b0};
      full_range_imag_out[26] <= {imag_in[15]};
      full_range_imag_out[25:0] <= {imag_in[15:0],10'b0};
    end
  6'b110111 : //-9 Equals left shift by 9 with sign extension
    begin
      full_range_real_out[26:25] <= {real_in[15],real_in[15]};
      full_range_real_out[24:0] <= {real_in[15:0],9'b0};
      full_range_imag_out[26:25] <= {imag_in[15],imag_in[15]};
      full_range_imag_out[24:0] <= {imag_in[15:0],9'b0};
    end
  .
  .
  .
endcase

```

この例では、出力はフル・スケールの 27 ビット・ワードです。後段の処理部に転送するデータのビット位置およびビット数は、ユーザーが決定する必要があります。ビット位置の選択により、入力サンプル・レベルに対する絶対ゲインが決まります。

図 1 は、入力信号が (5000H、0、0.. 最初のサンプルのみ 0×5000 で他はすべて 0) に対する 256 ポイント・クワッド出力 FFT のスケーリングについて説明しています。この入力に対する出力は、周波数 0 において 280H、exponent : -5 となります。この図は、フル・スケール・ストレージ・レジスタ [26..0] に対するスケーリングのすべての有効な指数値を示しています。exponent は -5 なので、該当するカラムのレジスタ値を参照します。このデータを図の一番右の 2 列に抜き出してあります。一番右の列では、スケーリング (0005000H) 後のゲイン補正されたデータを示します。このデータは期待される入力データと一致します。以降の処理で 16 ビットデータ幅を維持する場合、5000H になる下位 16 ビットを選択することができます。しかし、異なるビット範囲、例えば上位 16 ビットを選択した場合、結果は 000AH になります。したがって、ビットの選択は処理チェーンの相対ゲインに影響を与えます。

この例は、27 ビットのフル・スケール分解能と 16 ビットの出力分解能を有しているため、入力信号に対するユニティ・ゲインを維持するために下位 16 ビットを選択します。すべてのケースにおいて、下位 16 ビットを選択することが唯一のソリューション、または正しい選択肢であるとは限りません。この選択はどの信号レベルが重要であるかによって決めます。実験的に適切な範囲を選択する方法の 1 つは、期待されるシステム・データをテスト・ケースとし、シミュレーションすることです。シミュレーションの出力から、出力レジスタとして使用するビットの範囲を知ることができます。フル・スケール・データを使用しない場合（または MSB のみを使用する場合は、切り捨てる上位ビットに対し、飽和処理を行う必要があります。

図 1. 入力データ・サンプルのスケーリング = 5000H

Bit	Input	Output Data	Exponent									Looking at Exponent = -5			
			-11	-10	-9	-8	-7	-6	-5	-4	-3	Taking All Bits	Sign Extend / Pad		
	5000 H	280 H													
26			0												0
25			0	0											0
24			0	0	0										0
23			0	0	0	0									0
22			0	0	0	0	0								0
21			0	0	0	0	0	0							0
20			1	0	0	0	0	0	0					0	0
19			0	1	0	0	0	0	0	0				0	0
18			1	0	1	0	0	0	0	0	0			0	0
17			0	1	0	1	0	0	0	0	0			0	0
16			0	0	1	0	1	0	0	0	0			0	0
15	0	0	0	0	0	1	0	1	0	0	0			0	0
14	1	0	0	0	0	0	0	1	0	1	0	0		1	1
13	0	0	0	0	0	0	0	0	1	0	1	0		0	0
12	1	0	0	0	0	0	0	0	0	1	0	1		1	1
11	0	0	0	0	0	0	0	0	0	0	1	0		0	0
10	0	0		0	0	0	0	0	0	0	0	1		0	0
9	0	1			0	0	0	0	0	0	0	0		0	0
8	0	0				0	0	0	0	0	0	0		0	0
7	0	1					0	0	0	0	0	0		0	0
6	0	0						0	0	0	0	0		0	0
5	0	0							0	0	0	0		0	0
4	0	0								0	0	0		0	0
3	0	0									0	0		0	0
2	0	0										0		0	0
1	0	0												0	0
0	0	0												0	0

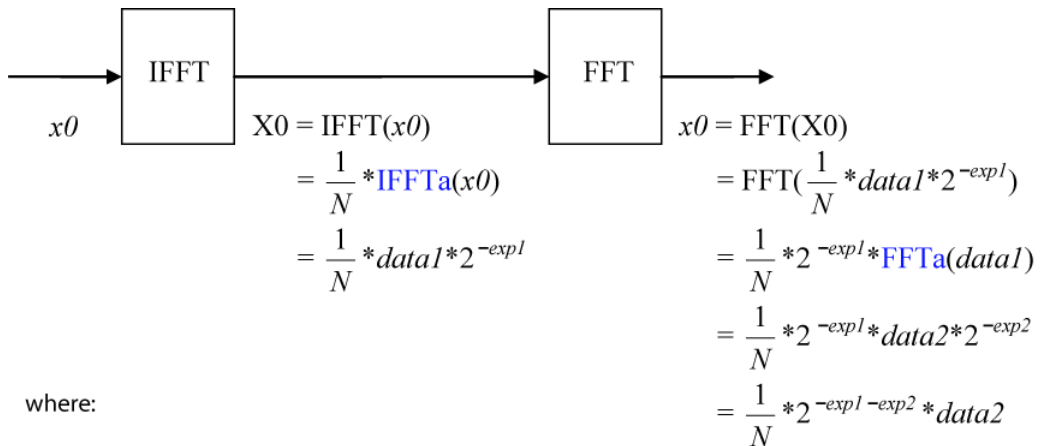
IFFT+FFT ペアでのユニティ・ゲインの達成

浮動小数点演算など、精度が十分高い場合には、IFFT と FFT をカスケード接続した場合、理論的にはユニティ・ゲイン（ゲイン 1）となります。ただし、BFP 演算では、ユニティ・ゲインを達成するために、IFFT/FFT ブロックの exponent 値に特に注意する必要があります。この項では、BFP 演算を使用してアルテラの IFFT/FFT MegaCore ペアからユニティ・ゲイン出力を得るために必要な手順を説明します。

BFP 演算部は exponent 入力を持たないため、直後に FFT ブロックに出力を供給している場合、IFFT ブロックからの exponent 値を追跡し、補正した後、最後に N で除算して元の信号を復元します。

図 2 に、IFFT および FFT の動作とユニティ・ゲインを達成するための式を導出します。

図 2. IFFT/FFT ペアのユニティ・ゲインの達成



where:

- $x0$ = Input data to IFFT
- $X0$ = Output data from IFFT
- N = Number of points
- $data1$ = Altera IFFT MegaCore output data & Altera FFT MegaCore input data
- $data2$ = Altera FFT MegaCore output data
- $exp1$ = Altera IFFT MegaCore output exponent
- $exp2$ = Altera FFT MegaCore output exponent
- IFFTa = Altera IFFT
- FFTa = Altera FFT

切り捨てを伴う X0 のスケーリング操作では、データ精度が失われ、x0 ではユニティ・ゲインは達成されません。X0 が最終結果となる場合にのみ X0 でスケーリング操作を実行しなければなりません。

ユニティ・ゲインを維持する方法の 1 つは、*exp1* 値を FFT ブロックの出力に渡すことです。また、 $data1 * 2^{-exp1}$ の完全精度を保持し、この値を FFT ブロックの入力として使用する方法もあります。2 番目の方法の欠点は、IFFT 動作からのビット幅の増加により、FFT ブロックの入力ビット幅が非常に大きくなることです。



詳しくは、www.altera.co.jp の「FFT/IFFT Unity Gain」のデザイン例を参照してください。



101 Innovation Drive
San Jose, CA 95134
(408) 544-7000
www.altera.com
Applications Hotline:
(800) 800-EPLD
Literature Services:
literature@altera.com

Copyright © 2005 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

