WP487 (v1.0) 2017 年 6 月 27 日

8 ビット ドット積の高速化

著者 : Yao Fu、Ephrem Wu、Ashish Sirasao

UltraScale™ および UltraScale+™ デバイスの DSP アーキテクチャでは、INT8 ベクター ドット積をスケーラブルに求めることができるため、ニューラル ネットワークにおけるたたみ込みと行列乗算のスループットを向上させます。このホワイト ペーパーで説明する手法により、DSP リソースの従来 (ネイティブ) の使用法に比べて 1.75 ～ 2 倍のスループットを実現できます。

概要

ザイリンクスの DSP48E2 スライスでは、2 つの並列 INT8 乗算/加算演算をパックすることで、3 つのベクター間で 2 つのドット積を計算できます。これは、人工知能を高速化するための、たたみ込みと行列乗算に一般的な計算パターンです。各 DSP スライスはニューラル ネットワークに特有の並列処理を利用し、平均で 1.75 倍 (LUT 未使用) ～ 2 倍 (LUT リソース使用) の乗算/加算 (Y = A*B+C) または乗算/加算/累算 (Y += A*B+C) 演算を実行して、INT8 ベクターのドット積を得ることができます。正規化線形関数 (ReLU) は一般的な活性化関数であるため、ザイリンクスの DSP48E2 スライスは符号なし 8 ビット (UINT8) データ オペランドをサポートして、INT8 に比べてデータ精度を 1 ビット高め、UINT8 データ ベクターと INT8 重みベクターとのドット積の処理速度を 1.78 ～ 2 倍向上させます。

ニューラル ネット ワークにおける数値精度の低減

ニューラル ネットワークにおける行列乗算とたたみ込みは、メモリと計算を多用する演算です。ニューラル ネットワークのデータと重みは 32 ビットの単精度浮動小数点形式 (fp32) でモデル化されていますが、研究者は 8 ビットの固定小数点整数 (INT8) をうまく利用して、高い精度を維持しながらメモリの格納域と帯域幅を減らしています [参照 1] [参照 2]。INT8 フォーマットはメモリの帯域幅と格納域を減らすだけでなく、ザイリンクスの DSP48E2 スライスを使用したドット積のスループットも向上させます [参照 3]。

ドット積のスループットが向上すると、行列乗算とたたみ込み両方の処理速度が向上します。行列乗算は一連のドット積に分解できるため、ドット積演算が高速になることのメリットが得られます。たたみ込みは行列乗算の特殊なケースです。たとえば、ニューラル ネットワークにおける 2 次元のたたみ込み (実際には 2 次元の相互相関の和を表す省略表現) は、入力データ ベクターとニューラル ネットワークの重みベクターとのドット積です。図 1 は、画像認識のたたみ込みニューラル ネットワーク (CNN) 例の第 1 段階を示しています。1 つの RGB 入力画像が 3 つの入力特徴マップ (IFM) にわかれ、一連のフィルター重みを用いて 4 つの出力特徴マップ (OFM)[1] を生成します。IFM、OFM、フィルター重みの関係を明確に示すため、IFM を縦、OFM を横に並べ、各 IFM-OFM ペアの交点にそれぞれ専用の 3x3 フィルター重みを並べています。図 1 は、12 のたたみ込みフィルターが、1 つの RGB 画像 (3 つの IFM) から 4 つの異なるビュー (4 つの OFM) をどのように生成するかを示しています。IFM エレメントはすべての OFM 列に横方向に送られるのに対し、出力値は下から上へ縦方向に合算されます。各行のフィルター重みは 1 つの IFM にのみ適用されます。同様に、各列のフィルター重みは 1 つの OFM にのみ影響します。各 OFM 列に沿った OFM の各エレメントは、2 つの 27 次元ベクターのドット積です。これは、3 つの IFM それぞれのデータに対する 9 つのエレメントにより合計 27 のエレメントが生成され、OFM 列に沿った 3 つの 3x3 フィルターにより 27 のフィルター重みが生成されるためです。

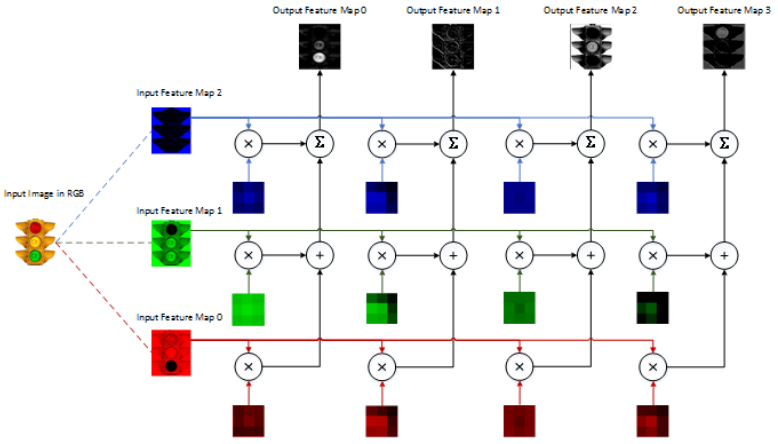

図 1: 画像認識における 2D たたみ込み

1. スペースに限りがあるため、4 つの OFM のみを示しています。実装される CNN では通常、最初のたたみ込み層で 64 以上の OFM が生成されます。

画像認識における 2D たたみ込み

CNN における conv2d() アルゴリズムは 2D たたみ込み (より正確には相互相関) の和であり、D_1 D_2 個のフィルターを用いて D_2 個の OFM を D_1 個の IFM から生成します。具体的に、Y を $B \times H_2 \times W_2 \times D_2$ 出力テンソル、X を $B \times H_1 \times W_1 \times D_1$ 入力テンソル、W を $F_y \times F_x \times D_1 \times D_2$ フィルター重みテンソルとすると、Y = conv2d(X,W,S) が成り立ち、B 個のジョブ群の中のジョブ g について、次のようになります。

$Y[g,u,v,d_2]$ は、OFM $d_2 \in [1,D_2]$ の行 u および列 v にあるエレメントです。

$X[g,i,j,d_1]$ は、IFM $d_1 \in [1,D_1]$ の行 i および列 j にあるエレメントです。

$W[y,x,d_1,d_2]$ は、行 y および列 x にあるフィルター エレメントで、IFM d_1 に適用されて OFM d_2 を生成します。

$S = (S_y,S_x)$ です。ここの S_y は垂直ストライド、S_x は水平ストライドを示します。

次に示す Conv2d() の例で、conv2d() は、すべての OFM エレメントの 2D たたみ込みの和を計算する入れ子のループです。B 個のジョブ群の中の各ジョブ g について、conv2d() 内で呼び出される関数 conv() は、ある特定の重みセット $W[:,:,d_1,d_2]$ を入力テンソル X の 1 つのスライス、つまり $X[g,:,:,d_1]$ に適用します (コロン記号 (「:」) は、すべての有効値を意味する)。入力テンソル X の $H_1 \times W_1$ 範囲外にある IFM エレメントは、すべてゼロです。各出力スライス $Y[g,:,:,d_2]$ は、conv() を D_1 回 (入力スライスごとに 1 回) 呼び出し (「Conv2d() から呼び出される Conv()」を参照)、結果を合算することで得られます。

たたみ込みニューラル ネットワークにおける Conv2d()

$$\text{conv2d}(X,W,S)\{$$

for g in 1 to B // each job

for $d_2 \in [1,D_2]$ // each OFM

$$Y[g,:,:,d_2] = \sum_{d_1=1}^{D_1} \text{conv}(X[g,:,:,d_1], W[:,:,d_1,d_2], [S_y,S_x])$$

return Y

Conv2d() から呼び出される Conv()

$$\text{conv}(X[g,:,:,d_1], H[:,:,d_1,d_2], [S_y,S_x]) \{$$

for c in 1 to W_2

for r in 1 to H_2

$$Z[r,c] = \sum_{u=-F_y/2}^{F_y/2} \sum_{v=-F_x/2}^{F_x/2} X[g,S_x r+u,S_y c+v,d_1] H[u+F_y/2,v+F_x/2,d_1,d_2];$$

return Z;

}

低精度ニューラル ネットワークでの DSP48E2

並列処理

たとえば INT18 では 1 つのドット積しか処理されませんが、2 つのドット積が共通の入力ベクターを共有する場合、INT8 オペランドを用いることで各 DSP48E2 スライスで 2 つのドット積が並列処理されます (図 2 参照)。行列乗算とたたみ込みでは並列処理が多いため、ニューラル ネットワークで 3 つのベクターのみを使用して 2 つのドット積を生成することは合理的です。

1. 行列 - 行列乗算 WX の場合、
   a. X の各列ベクターが W の 2 つの行ベクターを使用して 2 つのドット積を生成する (図 3 の a 参照)、または
   b. W の各行ベクターが X の 2 つの列ベクターを使用して 2 つのドット積を生成します (図 3 のb 参照)。
2. 行列 - ベクター乗算 Wx の場合、列ベクター x が W の 2 つの行ベクターを使用して 2 つのドット積を生成します。これは 1.a の特殊なケースで、行列 X に列が 1 つしかない場合です。
3. たたみ込みの場合、
   a. 1 つのフィルター重みベクターが IFM 全体で 2 つの区画で動作し、同一の OFM 内に 2 つのエレメントを生成する (図 4 の a 参照)、または
   b. 1 つの IFM 区画が 2 つのたたみ込みフィルターを通り、2 つの異なる OFM 内に 2 つのエレメントを生成します (図 4 の b 参照)。

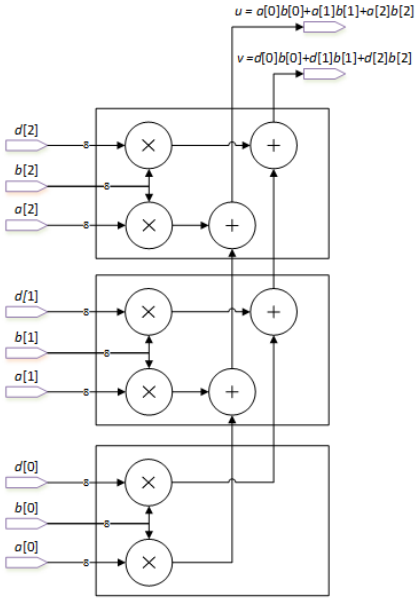

図 2: 共通の入力ベクターを共有する 2 つの並列 INT8 ドット積の概念図

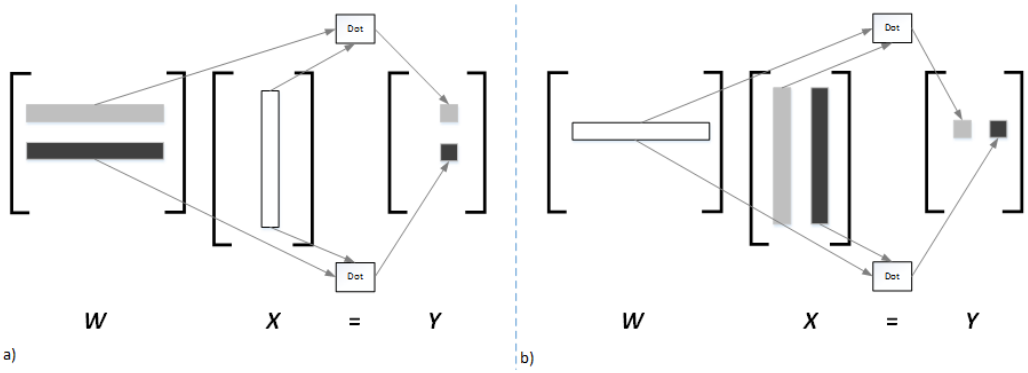

図 3: 行列 - 行列乗算と行列 - ベクター乗算

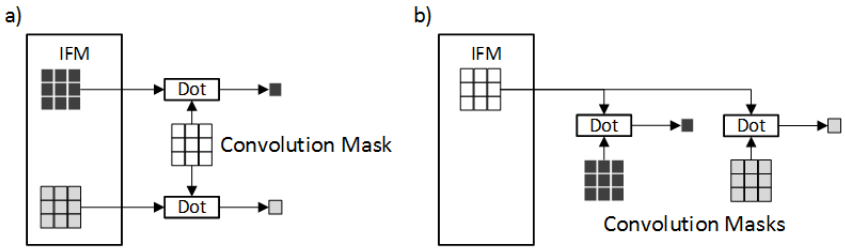

図 4: DSP48E2 での INT8 たたみ込みの並列処理

INT8 用の DSP48E2 モード

DSP48E2 スライスは、27 ビットの前置加算器、27x18 乗算器、48 ビットの後置加算器を備えた 2 の補数演算ユニットとして機能でき (図 5 参照)、次のような結果が得られます。

PCOUT = P = (A + D)B + PCIN

出力ポート P と PCOUT の違いは、P が FPGA インターコネクトに接続されるのに対し、PCOUT は同じ DSP48E2 カラムにある DSP48E2 スライスの別インスタンスの PCIN ポートにハードワイヤされる点です。PCIN-PCOUT 接続は、DSP48E2 カスケード接続を確立します。隣接する DSP48E2 スライスどうしは接しているため、動作クロック周波数を最大化できます。

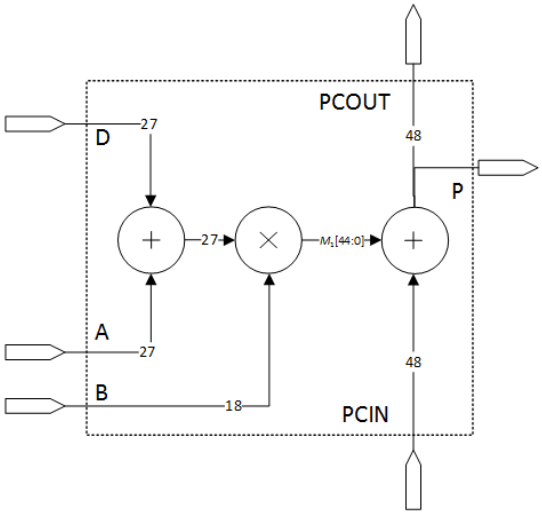

図 5: 8 ビット ドット積の高速化に使用される DSP48E2 モード

並列入力とパックされた出力のフォーマット

DSP48E2 スライスは、3 つの INT8 オペランド (a、d、b) を受け入れて ab と db を並列演算できます。オペランド a と d は 2 つのベクターのエレメントで、共通オペランド b (第 3 のベクターからのエレメント) で乗算されます。DSP48E2 スライスをこの演算に使用するには、a、d、b をそれぞれポート A、D、B に割り当てます (図 6 参照)。

- オペランド d は符号拡張され、ポート D の 27 ビット ワードを形成します。
- オペランド a は、算術的に 18 ビットだけ左シフトされ、27 ビットに符号拡張され、ポート A に割り当てられます。この演算により、オペランド a が DSP48E2 の前置加算器のポート A 内のできるだけ左端に効果的にシフトされると共に 1 ビットの余裕ができるため、オーバーフローを起こさずに、シフトされたオペランドを d に加算して 27 ビット ワードを生成できます(1)。a の左シフト量 18 は、B のポート幅と偶然同じになっています。
- 共有オペランド b は符号拡張され、ポート B に割り当てられます。

DSP48E2 スライス内の乗算器は、これらの入力割り当てを用いて $M = (a2^{18} + d)b$ を計算します (図 7 参照)。次に、3 つの N 次元ベクター (a、d、b) 間で 2 つのドット積をパックして積和にできます。

$$\sum_{i=0}^{N-1}(a_i 2^G + d_i)b_i$$

ここで、G = 18 です。

次の上位ドット積をパックされた結果から復元可能にするには、

$$\boldsymbol{a}\cdot\boldsymbol{b} = \sum_{i=0}^{N-1} a_i b_i$$

下位ドット積を下位の G = 18 ビット内に留める必要があります。これにより、N≤7 項になります。(2)

$$\boldsymbol{d}\cdot\boldsymbol{b} = \sum_{i=0}^{N-1} d_i b_i$$

この結果、G = 18 とすると、3 つの 7 次元ベクター間で 2 つのドット積を計算でき、下位ドット積が上位ワードに流れ込むことはありません。

1. DSP48E2 の前置加算器は 2 つの 27 ビット オペランドを受け入れて、28 ビットではなく 27 ビットの結果を生成します。

2. m ビットの 2 の補数は $[-2^{m-1}, 2^{m-1}-1]$ の範囲内にあるため、2 つの m ビットの 2 の補数は $\left[2^{m-1}-2^{2(m-1)}, 2^{2(m-1)}\right]$ の範囲内になります。したがって、このような N の積和は $\left[N(2^{m-1}-2^{2(m-1)}), N(2^{2(m-1)})\right]$ の範囲内になります。この和を $[-2^{q-1}, 2^{q-1}-1]$ の範囲内にある q ビットの 2 の補数として表すには、$-2^{q-1} \le N(2^{m-1}-2^{2(m-1)})$ および $2^{q-1}-1 \ge N(2^{2(m-1)})$ でなければなりません。後者の制約は前者の制約を伴うため、後者の不等式を並べ替えると $N \le (2^{q-1}-1)/2^{2(m-1)}$、つまり $N = \left\lfloor (2^{q-1}-1)/2^{2(m-1)} \right\rfloor$ になります。前述の例では、$m = 8$ および $q = 18$ のため、$N = 7$ となります。

DSP48E2 スライスの出力ポート P と PCOUT には、2 つのドット積を特殊な方法で表す、2G = 36 ビットのパック ワードが含まれます (図 8 参照)。各ベクターのエレメント数が 7 を超えないため、P[17:0] は常にドット積 **d·b** を 2 の補数として表しますが、P[35:18] は(1)、**d·b** が負でない場合は **a·b** を表し、それ以外の場合は **a·b** – 1 を表します。言い換えれば、**a·b** = P[35:18]+P[17] になります。P の残りのビット、つまり P[47:36] は符号拡張ビットであり、P[35] と同じです。

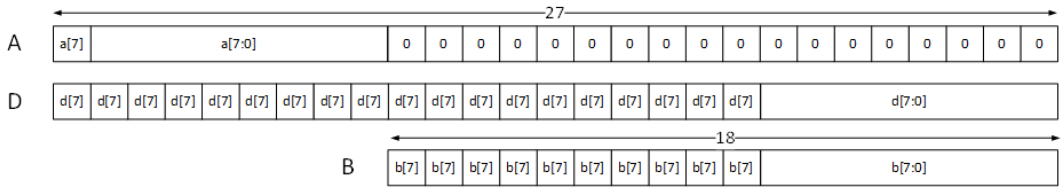

図 6: 入力フォーマット

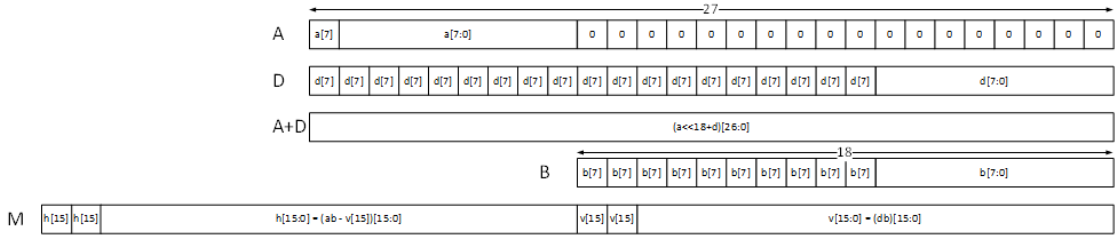

図 7: パック ワード内の 2 つの積

図 8: 7D までのベクターの 2 つのドット積の出力フォーマット

各 DSP48E2 スライスから P[35:18] を介入調整せずに、1 つまたは複数の DSP48E2 スライス(2) からパック ワードを 7 つまで合算できます (図 9 参照)。図 9 の上部にあるインクリメンターを LUT で実装した場合、ドット積のスループット速度向上は 100% になります。

1. 18 ビットの $P[35:18]$ には、$[N(2^{m-1} - 2^{2(m-1)}) - 1, N(2^{2(m-1)}) - 1]$ の範囲内 ($N = 7$ および $m = 8$) の $\boldsymbol{a} \cdot \boldsymbol{b} - 1$ を格納できます (7 ページの脚注 1 参照)。したがって、上位ワード $\boldsymbol{a} \cdot \boldsymbol{b} - 1$ は、18 ビットの 2 の補数で表現可能な $[-113793, 114687]$ の範囲内になります。

2. 後置加算器がアキュムレータとして構成されている場合、これらの 7 項を 7 つの DSP48E2 スライスで計算する必要はなく、1 つの DSP48E2 スライスのみで計算できます。実際に、1 ～ 6 つの DSP48E2 スライスの一部を乗算/加算モード、それ以外を積和モードにして、これらの 7 項を計算できます。詳細は、[参照 3] を参照してください。

調整の対象は、最終結果が含まれるパック ワードのみであることに注意が必要です。(1) P[17:0] では、下位ドット積は変更されていません。上位ドット積は、下位ドット積が負でない場合は変更されません。それ以外の場合は、P[35:18] の値を 1 だけインクリメントして、上位ドット積を復元する必要があります。

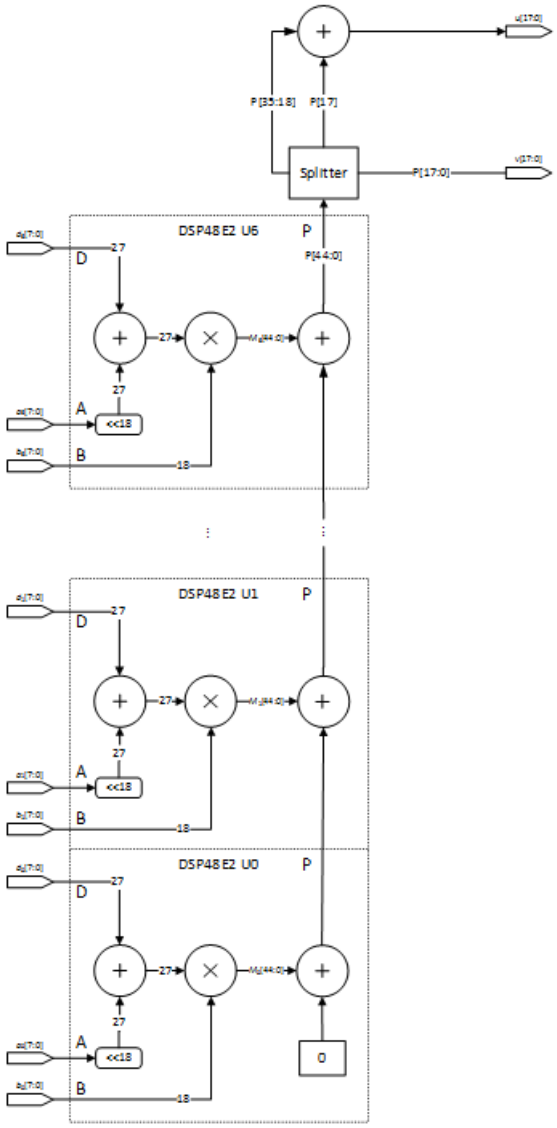

図 9: 上位ドット積の復元

1. たとえば、積和モードの DSP48E2 スライス 1 つで 7 項を 7 クロック サイクルで計算する場合は、7 番目のサイクルのパック ワードのみを調整します。

例

表 1 は、7 組のドット積 (1 行に 1 組) を 1 次元ベクター (i = 0) から 7 次元ベクター (i = 6) まで例示したものです。パック ワードを互いに合算して正しい結果を出す方法を説明するため、これら 7 組のドット積は、P_i 列に示すように互いを基に計算されています。この列には、次のような累積和が示されています。

$$P_i = \sum_{j=0}^{i} (a_j b_j 2^G + d_j b_j)$$

これは、次のような漸化式で表すこともできます。

$$P_0 = (a_0 2^G + d_0) b_0 \text{ および } P_i = P_{i-1} + (a_i 2^G + d_i) b_i, i > 0$$

DSP48E2 スライスの場合、a_i の左シフト量は G = 18 です。なお、パック ワードに対象のドット積が含まれていて、下位ドット積 $P_i[17:0]$ が負の場合 (つまり $P_i[17] = 1$ のとき) に限り、$P_i[35:18]$ 列に示す値を 1 だけインクリメントして上位ドット積にする必要があります。たとえば、表 1 の最終行の上位ワードは $P_6[35:18] = 24$、下位ドット積は次のようになります。

$$P_6[17:0] = \sum_{j=0}^{6} d_j b_j = -1$$

下位ドット積が負であるため、正しい上位ドット積 25 を得るには $P_6[35:18]$ の値を 1 だけインクリメントする必要があります。これで、2 つのドット積の値が、予測される結果を示す次の 2 列の値と一致します。

$$\sum_{j=0}^{i} a_j b_j \text{ および } \sum_{j=0}^{i} d_j b_j$$

すべての j<i に対して、$P_j[17] = 1$ のたびに $P_j[35:18]$ の値を 1 だけインクリメントして P_i を計算する必要はありません。上位 18 ビットの値を 1 だけインクリメントする必要があるのは、最後のパック ワードで、下位ドット積が負の場合のみです。

DSP48E2 の場合、入力 a_i の左シフト量は G = 18 です。4 項のベクター ドット積を計算するには、i = 3 の行を見ます。下位ドット積は、最後から 2 番目の列にあります (つまり $P_3[17:0] = -1$)。下位ドット積が負であるため、上位ドット積は $P_3[35:18] + 1 = 2$ にする必要があります。6 項のドット積を計算する場合、下位ドット積は $P_5[17:0] = 1$ です。これは負でないため、上位ドット積は調整なしで $P_5[35:18] = 18$ となります。表 1 を参照してください。

表 1: INT8 の計算例

i	a_i	d_i	b_i	$a_i b_i$	$d_i b_i$	$P_i = \sum_{j=0}^{i} (a_j 2^G + d_j) b_j$	$P_i[35:18]$	$\sum_{j=0}^{i} a_j b_j$	$P_i[17:0]$	$\sum_{j=0}^{i} d_j b_j$
0	1	-4	-2	-2	8	-524280	-2	-2	8	8
1	2	8	-3	-6	-24	-2097168	-9	-8	-16	-16
2	3	17	2	6	34	-524270	-2	-2	18	18
3	4	-19	1	4	-19	524287	1	2	-1	-1
4	5	-1	2	10	-2	3145725	11	12	-3	-3
5	6	4	1	6	4	4718593	18	18	1	1
6	7	-2	1	7	-2	65553599	24	25	-1	-1

7 次元ベクターを超える場合

図 6 に示す入力フォーマットでは、下位ドット積が上位ワードに入ることなく、7 項までのドット積を合算できます。8 項以上を合算するには、下位ドット積と上位ドット積をさらに分けて、より大きいパック ワードを形成する必要があります。この分割は、7 番目の項を合算する DSP48E2 スライスの出力で、LUT を使用せずに FPGA ファブリックの配線可能です。大きいパック ワードの合算は、最後の大きいパック ワードが得られるまで、上位ワードを調整せずに実行する必要があります。DSP48E2 スライスは 48 ビットの 2 入力加算器としても使用できるため、下位ワードと上位ワードを 18 ビットから 24 ビットに拡張できます (図 10 参照)。これにより、複数の 7 DSP48E2 カスケード接続からの結果を合算できます。繰り返しますが、下位ドット積の符号に基づいて上位ワードを調整する必要があるのは、最後の大きいパック ワードのみです。クロック レートを最大にするには、図 10 に示す 48 ビットの加算器と 24 ビットのインクリメンターも DSP48E2 スライスにします。この技法を用いなければ 8 項しか計算されませんが、この場合は 8 つの DSP48E2 スライスごとに 14 項が計算されます。したがって、スループットの向上率は 14/8 – 1 = 75% となります。

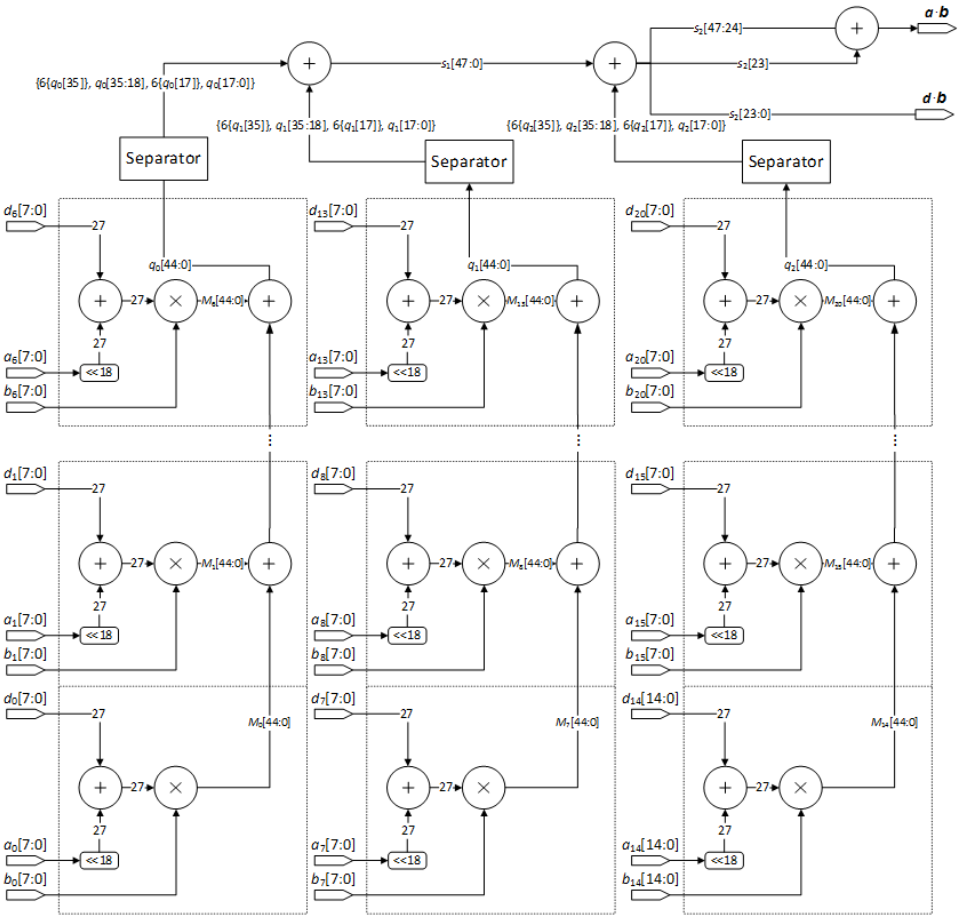

図 10: 高次元ベクターのドット積でのパック ワードの分割

符号なし /符 号付き INT8

正規化線形関数 (ReLU) は f(x) = max (0,x) を実装し、活性化関数として、機械学習のトレーニング速度向上に用いられています [参照 4] [参照 5] (図 11 参照)。INT8 値として表される正規化数値には、必ずゼロの最上位ビットが含まれます。したがって、データを符号なしの 8 ビット整数 (UINT8) で表すことが望まれます。これは、このフォーマットではメモリの格納域または帯域幅が同じ INT8 に比べて精度が 1 ビット増すためです。DSP48E2 は 2 つのドット積 **a**·**b** および **d**·**b** をサポートしますが、それには共通ベクター **b** が INT8 エレメントで構成され、2 つのベクター **a** および **d** に UINT8 エレメントが入ることが条件となります。たたみ込みの場合、重みには符号が付くため、重みベクターはベクター **b** になります。正規化データはベクター **a** および **d** になります。6 ページの脚注 2 と同じダイナミック レンジ解析を使用すると、次に示すように、下位ドット積が上位ドット積に流れ込むことなく合算できる項数が 8 になり、結果的にスループットが 78% 向上します。

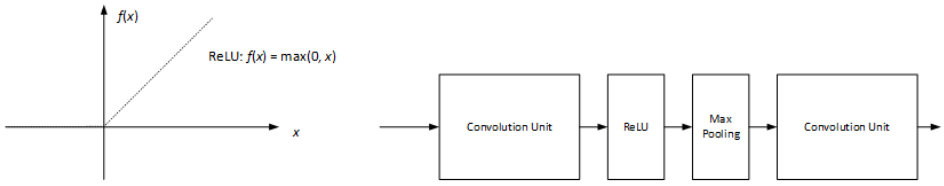

図 11: 正規化線形関数 (ReLU)

a と **d** はどちらも符号なしのため、DSP48E2 前置加算器を使用せずに、これら 2 つのベクターを 1 つの 27 ビット ベクターにパックできます。エレメント $a_i \in$ **a** とエレメント $d_i \in$ **d** のギャップを最大化するため、a_i は DSP48E2 ポート A によって左揃えされ、d_i は右揃えされます。言い換えれば、演算 a_i << G+d_i (「<<」は左シフト演算子) により 27 ビット ワードが形成されます。a_i のシフト量は G = 19 ビットになり、INT8 の場合より 1 ビット多くなります。これで a_i の最上位ビットは DSP48E2 ポート A の符号ビットになるため、a_i[7] = 1 であれば必ず、DSP48E2 はポート A の値を負として処理します。UINT8 値 u が INT8 値 x として解釈される場合は、次のようになります。

$$x = \begin{cases} u, & u < 128 \\ u - 256, & u \geq 128 \end{cases}$$

したがって、ポート A の値は次のようになります。

$$A_i = \begin{cases} a_i 2^G + d_i, & a_i < 128 \\ a_i 2^G + d_i - 256 \cdot 2^G, & a_i \geq 128 \end{cases}$$

これを次のように簡略化できます。

$$A_i = a_i 2^G + d_i - a_i[7] \cdot 256 \cdot 2^G$$

b_i で乗算すると、出力パック ワードは次のようになります。

$$M_i = (a_i 2^G + d_i - a_i[7] \cdot 256 \cdot 2^G) b_i$$

合算の前に、バイアス $-256 \cdot 2^G a_i[7] b_i$ を M_i から削除する必要があります。

DSP48E2 スライスの後置加算器には幅の広いバイアス ポート C があるため、値 $256 \cdot 2^G a_i[7] b_i$ を積 M_i に加算して

バイアスを削除し、$P_{i+1} = P_i + M_i + 256 \cdot 2^G a_i[7] b_i = a_i b_i 2^G + d_i b_i$ を形成できます。

これで、最大 N 次元において 2 つのドット積を P_N として表すことができます。このパック ワードのフォーマットは INT8 の場合と同じです。つまり、**d**·**b** が G = 19 ビットを超えないような N 次元ベクターにおいて、**d**·**b** < 0 の場合は P_N[37:19] = **a**·**b** – 1、そうでない場合は P_N[37:19] = **a**·**b** となります。

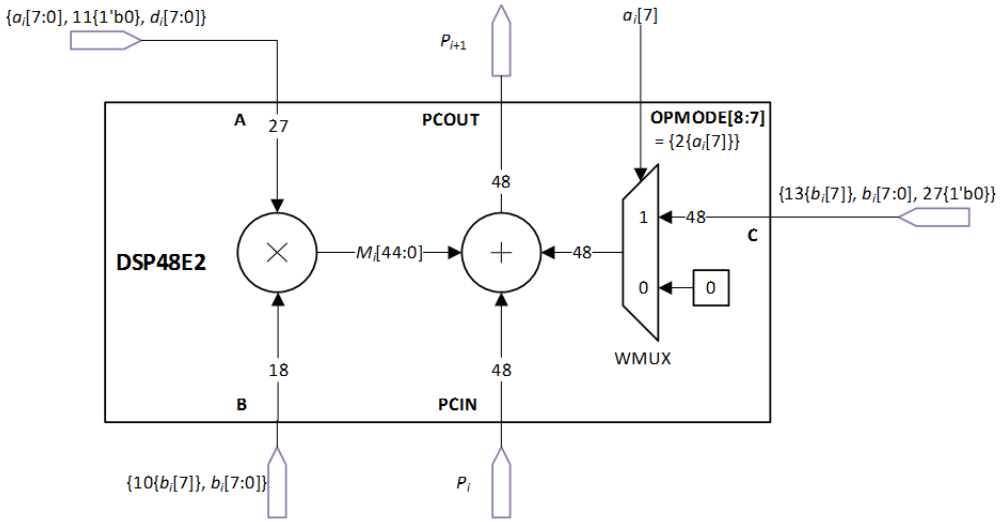

図 12: INT8 エレメントで乗算された 2 つの UINT8 エレメント

ポート A は、符号なし 8 ビット オペランド a_i および d_i を最大分割量で受け取ります。ポート B は符号付きオペランド b_i を受け取ります。a_i の左シフト量は G = 19 です。

ただし、$a_i > 128$ の場合は必ず、DSP48E2 スライスで A の入力が負として処理されるため、実質的に $-256 \cdot 2^G b_i$ が M_i に加算されます。

このバイアスを削除するため、DSP48E2 スライスの WMUX を通して値 $256 \cdot 2^G a_i[7] b_i$ がパック ワード M_i に再度加算されます。$a_i[7] = 1$ の場合に限り、WMUX はポート C を介して後置加算器に $256 \cdot 2^G b_i = 2^{27} b_i$ を送信します。2 つの入力ピン OPMODE[8:7] は、WMUX の選択ポートを制御します。これら 2 つのビットがどちらも 1 の場合はポート C の値が選択され、そうでない場合は 0 が選択されます。

m ビットの符号付き 2 の補数と m ビットの符号なし数値の間における最大 $N = \lfloor (2^{q-1} - 1)/(2^{2m-1} - 1) \rfloor$ 個の積の和は、q ビットの 2 の補数で表すことができます。(1) ここで、m = 8、q = G = 19 です。したがって、最大で N = 8 個の積を 19 ビットの 2 の補数に累算できます。図 12 と同じ技法を用いれば、8 次元ベクターを超えることができます。この場合は、9 つすべての DSP48E2 スライスが 8×2 = 16 の乗算/加算演算を実行しますが、パッキングなしの場合は、この演算が 9 回しか実行されません。したがって、スループットの速度向上率は 16/9≈78% になります。

1. m ビットの符号なし数値の範囲は $[0, 2^m - 1]$ であり、符号付き 2 の補数の範囲は $[-2^{m-1}, 2^{m-1} - 1]$ です。

   したがって、これら 2 つの数値の積の範囲は $[-2^{m-1}(2^m - 1), (2^m - 1)(2^{m-1} - 1)]$ であり、

   $2m$ ビットの 2 の補数で表すことができます。

   なぜなら、この範囲は $[-2^{2m-1}, 2^{2m-1} - 1]$ であり、$m \geq 1$ であれば必ず積の範囲より広くなるためです。

   このような N の積和は $[-N2^{m-1}(2^m - 1), N(2^m - 1)(2^{m-1} - 1)]$ となり、q ビットの 2 の補数で表すことができ、

   この範囲は、$N \leq (2^{q-1} - 1)/(2^{2m-1} - 1)$ であれば必ず $[-2^{q-1}, 2^{q-1} - 1]$ となります。

   したがって、N の最大値は $\lfloor (2^{q-1} - 1)/(2^{2m-1} - 1) \rfloor$ となります。

まとめ

人工知能における大規模な並列処理は、高い計算密度を必要とします。ザイリンクスの DSP48E2 スライスでは、2 つの並列 8 ビット乗算/加算演算をパックすることで、たたみ込みと行列乗算の処理速度を 1.75 ～ 2 倍向上できます。DSP48E2 スライスの 27x18 乗算器は、1 つの共通オペランドを共有する 2 つの並列 8 ビット乗算をサポートします。具体的には、2 つの INT8 オペランドが 27 ビット ポートにパックされた後、18 ビット ポートにある 3 番目の INT8 共有オペランドで並列乗算されます。ReLU 活性化関数を使用するアプリケーションの場合、DSP48E2 スライスは 2 つの UINT8 データ オペランドを 1 つの INT8 重みで並列乗算して、符号なしデータに有効ビットを追加します。どちらの場合でも、DSP48E2 スライスは 2 つの並列積を特殊な完全精度の双対積フォーマットで出力します。このフォーマットで表される 2 つの並列積は、1 つの 2 の補数であるかのように (INT8xINT8 の場合は 7 項まで、UINT8xINT8 の場合は 8 項まで) 合算でき、完全精度が維持されることが数学的に証明されています。これより高い次元のドット積の場合は、FPGA ファブリックの配線で 2 つの並列積が分割され、LUT または追加の DSP48E2 スライスによって合算されます。この特殊な双対積フォーマットは、合算全体を通して調整なしで使用されます。ドット積の最終結果のみ、変換が必要な場合があります。下位積が負の場合に限り、上位積の値を 1 だけインクリメントする必要があります。これにより、追加の合算を LUT が実行する場合は、速度が 2 倍向上します。INT8xINT8 の場合は速度向上が 1.75 倍と若干下がりますが、ファブリックの配線をほとんど用いない通常の DSP アレイ実装によって、クロック周波数は高くなります。同様に、UINT8xINT8 の場合の速度向上は 1.78 倍です。

参考資料

注記: 日本語版のバージョンは、英語版より古い場合があります。

1. Dettmers, T.、『8-Bit Approximations for Parallelism in Deep Learning』ICLR 2016
2. Gysel, P.、Motamedi, M.、Ghiasi, S.、『Hardware-oriented Approximation of Convolutional Neural Networks』arXiv preprint arXiv:1604.03168, 2016
3. 『UltraScale アーキテクチャ DSP スライス ユーザー ガイド』(UG579: 英語版、日本語版)
4. Glorot, X.、Bordes, A.、Bengio, Y.、『Deep Sparse Rectifier Neural Networks』JMLR, 15, 315-323, 2011
5. Krizhevsky, A.、Sutskever, I.、Hinton, G.、『Imagenet Classification with Deep Convolutional Neural Networks』NIPS, 1097-1105, 2012

改訂履歴

次の表に、この文書の改訂履歴を示します。

日付	バージョン	内容
2017 年 6 月 27 日	1.0	初版

免責事項

本通知に基づいて貴殿または貴社 (本通知の被通知者が個人の場合には「貴殿」、法人その他の団体の場合には「貴社」。以下同じ) に開示される情報 (以下「本情報」といいます) は、ザイリンクスの製品を選択および使用することのためにのみ提供されます。適用される法律が許容する最大限の範囲で、(1) 本情報は「現状有姿」、およびすべて受領者の責任で (with all faults) という状態で提供され、ザイリンクスは、本通知をもって、明示、黙示、法定を問わず (商品性、非侵害、特定目的適合性の保証を含みますがこれらに限られません)、すべての保証および条件を負わない (否認する) ものとします。また、(2) ザイリンクスは、本情報 (貴殿または貴社による本情報の使用を含む) に関係し、起因し、関連する、いかなる種類・性質の損失または損害についても、責任を負わない (契約上、不法行為上 (過失の場合を含む)、その他のいかなる責任の法理によるかを問わない) ものとし、当該損失または損害には、直接、間接、特別、付随的、結果的な損失または損害 (第三者が起こした行為の結果被った、データ、利益、業務上の信用の損失、その他あらゆる種類の損失や損害を含みます) が含まれるものとし、それは、たとえ当該損害や損失が合理的に予見可能であったり、ザイリンクスがそれらの可能性について助言を受けていた場合であったとしても同様です。ザイリンクスは、本情報に含まれるいかなる誤りも訂正する義務を負わず、本情報または製品仕様のアップデートを貴殿または貴社に知らせる義務も負いません。事前の書面による同意のない限り、貴殿または貴社は本情報を再生産、変更、頒布、または公に展示してはなりません。一定の製品は、ザイリンクスの限定的保証の諸条件に従うこととなるので、https://japan.xilinx.com/legal.htm#tos で見られるザイリンクスの販売条件を参照してください。IP コアは、ザイリンクスが貴殿または貴社に付与したライセンスに含まれる保証と補助的条件に従うことになります。ザイリンクスの製品は、フェイルセーフとして、または、フェイルセーフの動作を要求するアプリケーションに使用するために、設計されたり意図されたりしていません。そのような重大なアプリケーションにザイリンクスの製品を使用する場合のリスクと責任は、貴殿または貴社が単独で負うものです。http://japan.xilinx.com/ legal.htm#tos で見られるザイリンクスの販売条件を参照してください。

自動車用のアプリケーションの免責条項

オートモーティブ製品 (製品番号に「XA」が含まれる) は、ISO 26262 自動車用機能安全規格に従った安全コンセプトまたは余剰性の機能 (「セーフティ設計」) がない限り、エアバッグの展開における使用または車両の制御に影響するアプリケーション (「セーフティ アプリケーション」) における使用は保証されていません。顧客は、製品を組み込むすべてのシステムについて、その使用前または提供前に安全を目的として十分なテストを行うものとします。セーフティ設計なしにセーフティ アプリケーションで製品を使用するリスクはすべて顧客が負い、製品の責任の制限を規定する適用法令および規則にのみ従うものとします。

この資料に関するフィードバックおよびリンクなどの問題につきましては、jpn_trans_feedback@xilinx.com まで、または各ページの右下にある [フィードバック送信] ボタンをクリックすると表示されるフォームからお知らせください。いただきましたご意見を参考に早急に対応させていただきます。なお、このメール アドレスへのお問い合わせは受け付けておりません。あらかじめご了承ください。