C-010

SoC FPGA を用いた MoveNet の高速化
High-speed MoveNet using SoC FPGAs

鎌倉 生昇‡　中西 知嘉子‡
Takanori Kamakura　Chikako Nakanishi

1. はじめに

近年，エッジデバイス上で AI 推論処理を行う「エッジ AI」が注目されている．エッジデバイスは，サーバ用 PC などと比較するとデバイス性能が低い．よって，複雑かつ膨大な量の処理を行う高精度な AI をリアルタイムに動作させることは難しい．

本研究では SoC FPGA を用い，AI 処理内の高負荷で繰り返し回数の多い処理のみを FPGA 回路で行うことで AI 処理の高速化を行った．SoC FPGA は CPU と FPGA が 1 つのチップ上に統合された製品であり，互いに高速なバスで接続されていることから協調動作を行いやすいという特徴を持つ．この特徴に着目し，AI 処理の大半の処理を CPU 上で動作させ，高負荷で繰り返し回数の多い特定の処理を回路化し，FPGA 上で動作させることで，姿勢推定モデル「MoveNet」の高速化を図った[1].

本発表では，更なる高速を目指す際の課題である CPU-FPGA 間での不要なデータコピーによるデータ共有時間を削減する．そのため，Add 層の回路化，CPU と FPGA の両者がアクセス可能なバッファ領域（以降，共有メモリと呼ぶ）への CPU での層の処理時の直接書き込むように変更し動作させた結果をまとめる．

2. 使用デバイス・モデル・ツール

2.1 Ultra96-V2[2]

Ultra96-V2 は，AVNET 社より販売されている Arm コア内蔵の SoC FPGA ボードである．SoC FPGA とは，CPU と FPGA が同一チップ上に存在するデバイスである．OS は Ultra96-V2 向けに公開されている Debian11 を使用した．

2.2 MoveNet[3]

MoveNet とは，2021 年に Google から発表された姿勢推定モデルである．MoveNet には，速度を重視した Lightning と精度を重視した Thunder の 2 つのモデルが存在する．本研究では，速度を重視するため，Lightning を使用した．

2.3 開発ツール

2.3.1 回路生成ツール

回路は，C++言語で記述したコードに高位合成を使用して RTL 記述言語に変換することで作成した．また，高位合成を行うツールとして Vitis HLS 2020.2[4]を，回路生成を行うツールとして Vivado Design Suite 2020.2[5]を使用した．

2.3.2 Ceras[6]

Ceras は，弊研究室で独自に開発された学習済み AI モデルを C++言語の標準ライブラリのみで実行可能なライブラリである．CPU と FPGA の協調動作を前提に設計されており，高位合成前の回路用 C++コードを接続することで，回路生成前に論理の確認が可能である．

また，Ceras のモデルファイルには，回路を効率的に動作させるための情報（メモリブロック番号）が付加されている．

2.4 使用回路

使用回路は，Conv2D 層，Depthwise Conv2D 層，活性化関数処理を行うものである．先行研究[7]によって開発された Conv2D 層向け汎用回路をベースに，MoveNet で使用することを想定し作成した．この回路では，共有メモリをブロックに分割することにより，データを管理する．メモリブロックの指定は Ceras によって，モデルファイル作成時に行われる．推論処理実行時には，CPU が転送するデータと，どのメモリブロックに回路からの出力を書き込むかを回路側に指示する．具体例を図 2.1 に示す．今後使用されないデータは上書きすることで共有メモリを効率的に使用する．しかし，共有メモリを用いて管理するデータは回路で実行する層のデータのみである．

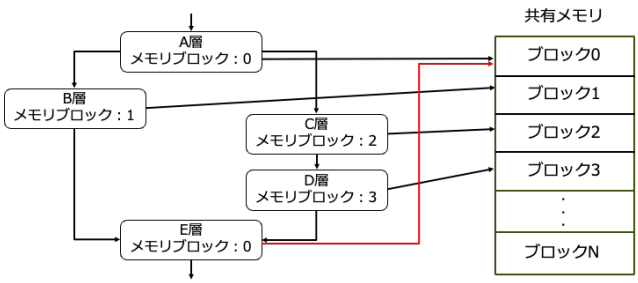

図 2.1　データの管理方法

3. 推論時間と内訳の調査

2.4 節の使用回路を用いて，Conv2D 層，Depthwise Conv2D 層を回路で動作させた際の推論時間とその内訳の調査を行なった．表 3.1 に推論時間とその内訳を示す．MoveNet の全 90 層の内，74 層を回路で，16 層を CPU で処理している．

表 3.1　推論時間とその内訳

推論時間(ms)	回路実行	271.792
	CPU処理	61.924
	データ共有	24.028
	合計	357.744

表 3.1 より，推論時間全体において，CPU での処理に 61.924ms，CPU と共有メモリ間のデータコピーに 24.028ms かかっていることが確認できる．CPU で層を処理する場合，回路から出力された共有メモリ上のデータを CPU 上の配列にコピーする．そして，層の処理を CPU 上で行い，出力を再び共有メモリ上にコピーする．これにより，CPU で

処理する層が存在すると CPU と共有メモリ間のデータコピーが発生し，推論処理に時間がかかる．そこで，CPU での処理時間削減のために Add 層の回路化を，CPU と共有メモリ間のデータコピーの削減するために，CPU で処理する層の出力結果の共有メモリへの直接書き込みを行うように変更を行う．

4. 提案手法

4.1 Add 層の回路化

Add 層は，２層の演算結果を加算する層である．Add 層の回路化を行い，回路で処理することで，CPU での処理時間の削減を図る．しかし，回路は 2 つの入力データが共有メモリ上にある場合のみ，層を回路で処理する仕様である．そのため，入力が CPU で処理する層の場合，CPU で処理した出力結果を共有メモリへコピーする必要がある．図 4.1 に共有メモリへのデータコピーが発生する具体的な層構成を示す．

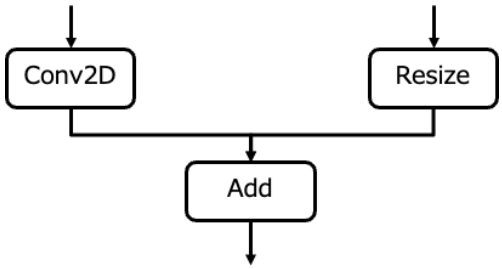

図 4.1　Add 層の入力例

図 4.1 より，Resize 層は CPU で処理するため，Resize 層の出力結果を共有メモリへコピーする．

4.2 CPU で処理する層の共有メモリへの直接書き込み

Resize 層は 90 層のうち，3 層のみである．層数が少ないため，回路化は行わず Resize 層は CPU で処理する．ただし前節で述べたように，CPU と共有メモリ間のデータ共有時間を削減するために，CPU と共有メモリ間のデータコピーの削減を行う．そのため，CPU で層を処理するとき，共有メモリのブロックにメモリアドレスでアクセスし処理する（以降，データの再利用と呼ぶ）．共有メモリ上のデータの再利用方法を図 4.2 に示す．

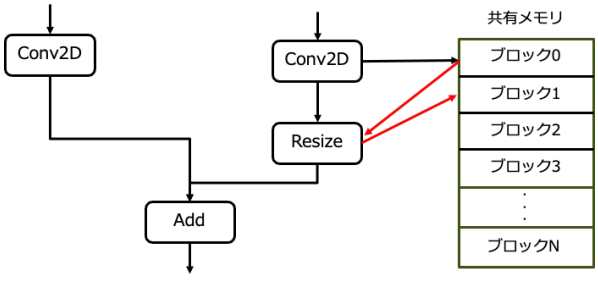

図 4.2　データ再利用方法

図 4.2 のように，直前の Conv2D 層の演算結果が格納されている共有メモリのブロックにメモリアドレスでアクセスし，処理を行い，Resize 層に割り振られているメモリブロックに結果を書き込む．これにより，CPU 上の配列への不要なデータコピーを削減する．

5. 結果

提案手法を適用して MoveNet を動作させた際の推論時間とその内訳を表 5.1 に示す．

表 5.1　提案手法適用後の推論時間とその内訳

推論時間(ms)	回路実行	274.026
	CPU処理	15.643
	データ共有	9.466
	合計	299.135

表 3.1 と表 5.1 より，推論時間全体では提案手法により，357.744ms から 299.135ms となり，約 1.2 倍の高速化を実現した．また，CPU での処理時間では 46.281ms，約 4 倍の高速化を実現した．さらに，CPU と共有メモリ間のデータ共有時間では 14.562ms，2.5 倍の高速化を実現した．

6. 考察

5 章より，CPU での処理時間が大幅に削減された．その要因として，Add 層の回路化だけでなく，CPU での配列の初期化の影響が考えられる．本研究で，CPU と共有メモリ間のデータコピーを削減したことで，配列の初期化が不要になり，CPU の処理時間を大幅に削減できたと考えられる．

7. まとめ

本発表では，Add 層の回路化と，CPU で処理する層の出力結果を共有メモリに直接書き込むことにより，推論時間の短縮を図った．結果として，手法適用前と比較して約 1.2 倍の高速化を実現した．

今後の展望として，現在推論を 32 ビット浮動小数点で行なっているが，Conv2D 層などの一部の層を 8 ビット量子化したモデルでの推論処理を考えている．しかし，8 ビット量子化を行うと，推論精度が低下する可能性があるため，姿勢推定モデルの精度評価を行う必要があると考えている．

参考文献

[1] 鎌倉生昇, 中西知嘉子, “姿勢推定モデル「MoveNet」のエッジデバイスにおける動作高速化手法の検討”, 情報処理学会全国大会(2025).
[2] Ultra96-V2, https://japan.xilinx.com/products/boards-and-kits/1-vad4rl.html
[3] MoveNet, https://blog.tensorflow.org/2021/05/next-generation-pose-detection-with-movenet-and-tensorflowjs.html
[4] Vitis HLS, https://japan.xilinx.com/support/documentation-navigation/design-hubs/2020-2/dh0090-vitis-hls-hub.html
[5] Vivado Design Suite, https://www.xilinx.com/content/dam/xilinx/support/documents/sw_manuals_j/xilinx2020_2/ug910-vivado-getting-started.pdf
[6] 西岡駿, 中西知嘉子, “機械学習ライブラリの C 言語化の実現”, 電子情報通信学会ソサイエティ大会(2021)
[7] 田嶋夏己, 中西知嘉子, “SoC FPGA を用いたエッジ AI の推論処理の高速化手法の検討”, 電子情報通信学会総合大会(2024)

† 大阪工業大学 情報科学研究科 情報科学専攻
Graduate School of Information Science and Technology
Osaka Institute of Technology
‡ 大阪工業大学 情報科学部 情報知能学科
Department of Information and Computer Science
Osaka Institute of Technology