C-012

エッジ AI のアクセラレータによるソフト部サポート方法の検討
Investigation of operational method for support of software part by hardware part in edge AI

吉田 裕紀‡　　中西 知嘉子‡
Hiroki Yoshida　Chikako Nakanishi

1. はじめに

近年, AI 技術は進化し続けている. その中でもエッジデバイス上で AI を動作させる「エッジ AI」が注目を集めている. しかし, 一般的なエッジデバイスは性能が低く, 高速な AI 処理は難しい.

そこで, 本研究では SoC FPGA を用い, CPU によるソフトウェア処理と, FPGA による回路を使用したハードウェア処理の両面から AI 処理の高速化を行った. SoC FPGA は CPU と FPGA が 1 つのチップ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が行いやすいという特徴を持つ. この協調動作が行いやすいという点に着目し, 大半の処理を CPU 上で動作させつつ, 高負荷な特定の箇所のみを回路化し FPGA 上で動作させることで, 物体検出 AI の高速化を図った[1].

回路を用いた高速化の結果を確認すると, 推論時間に占める割合として CPU 処理に約 23%, 回路動作に約 42%, CPU-FPGA 間のデータ共有に約 35%となっており, 更なる高速化のためには, 特に CPU-FPGA 間でのデータ共有に要している時間は無視できないと考えた.

本発表では CPU-FPGA 間のデータ共有方法に焦点を当て, アクセラレータによるソフト部サポートの方法について検討していく.

2. 開発環境・使用モデル

2.1 開発環境

エッジデバイスに Avnet 社より販売されている Ultra96-V2[2]を用いた. OS は Ultra96-V2 向けの Debian を使用した.

2.2 YOLOX

YOLOX[3]は物体検出 AI の YOLO シリーズより, 2021 年に発表されたモデルである. YOLO シリーズは, 他の物体検出モデルと比較し, 推論速度が高速なことが特徴として挙げられる. YOLOX には計算量に応じた複数のモデルが存在するが, 本研究では比較的軽量で高速に動作する YOLOX-s モデルを使用した.

3. 実装手法

3.1 回路部

3.1.1 高位合成による回路生成

回路は, C++言語で記述したコードを高位合成を用いて RTL 言語へ変換することで作成した. 高位合成には Vitis HLS 2020.2[4], 回路生成には Vivado Design Suite 2020.2[5]を使用した. Ultra96-V2 には Debian のデバイスツリー・オーバーレイを用いることで回路を実装する.

3.1.2 使用回路

使用回路は, Conv2D 層及び活性化関数処理を行うものである. 先行研究[6]によって開発された Conv2D 層向け汎用回路をベースに作成した. 実装されている活性化関数は Hard-Swish 関数, Hard-Sigmoid 関数, Linear 関数であり, 活性化関数の種類の選択や, 活性化関数処理を行うか否かを CPU による制御で決定できるよう設計されている. また, 回路には保持できるデータの上限が存在するため, 仕様として各上限サイズが決められている. 表 1 に回路の仕様を示す.

表 1 ベース回路の仕様

入力データサイズ上限	66
カーネルの上限数	64
1x1チャネル数上限	576
3x3チャネル数上限	64

3.1.3 データ共有方法

回路で処理を行うには, 演算に必要なデータを CPU 側と回路側で共有する必要がある. CPU 側メモリと共有メモリとの間では, データは互いにコピーすることで共有され, 共有メモリから回路へは DMA 転送によって回路へと転送される. コピーするデータはカーネルデータ, バイアスデータ, 入力データの 3 種類となっている. 推論開始後変化のないカーネルデータ及びバイアスデータはモデル読み込み時に, あらかじめ共有メモリにコピーを行う. 入力データは演算時に CPU から共有メモリへコピーを行う. また, 演算終了後も DMA 転送により回路から共有メモリへと演算結果を取り出し, 共有メモリから CPU 側メモリへとコピーを行う.

3.2 ソフトウェア部

3.2.1 データ整形

入力データの縦×横サイズや, チャネル数, カーネル数が表 1 の各上限を上回る場合, それぞれ CPU 上でデータの分割処理を行う. 分割されたデータは順に回路で演算され, 分割された全てのデータの演算が終了した後, CPU 上で演算結果を合成する. チャネル数が上限を上回った場合のみ, 活性化関数処理は CPU 上で行い, その他の分割では回路内で活性化関数処理を行う.

3.2.2 Ceras

Ceras[7]は同研究室にて開発された, 学習済み ONNX モデルの推論を C++言語で実行するライブラリである. C++言語の標準モジュールのみで構成されており, 環境構築が不要な点や, 層内部の処理単位にまで着目できる点, 開発環境を C++言語で統一できる点から使用した.

本研究では, Ceras を用いる関係で YOLOX-s を ONNX モデルへと変換し使用している.

4 手法の検討

4.1 データコピー時間

前章で述べた環境で YOLOX-s を実行した際の推論時間は約 9.83s となっており，回路化されている Conv2D 層と活性化関数処理は約 7.61s であった．そして，回路化部に要した時間の約 46%にあたる約 3.47s が CPU-FPGA 間でデータコピーを行うために要した時間であった．これは，推論時間全体と比較しても約 35%もの時間を占めている．図 2 に推論時間におけるデータコピー時間の割合を示す．

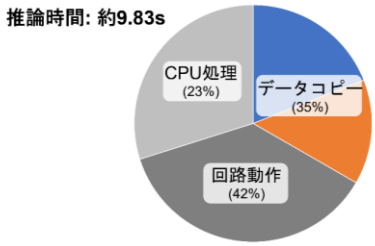

図 2 推論時間中のデータ共有時間の割合

4.2 共有メモリの活用

データ共有時間の削減を行うにあたり，データの共有方法に注目した．流れのイメージを図 3 に示す．

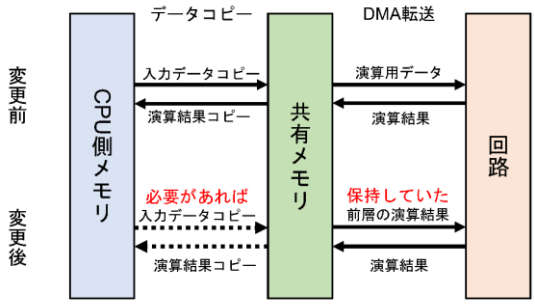

図 3 データコピー削減イメージ

回路を用いて演算を行う際，演算に用いるデータの入力や，演算結果の取り出しは必須であり，演算結果を CPU 側のメモリに格納するまでの一連の過程が必要となる．しかし，回路の演算結果をそのまま用いて再び回路で演算を行うことが可能な場合には，共有メモリ内に演算結果を保持し，保持した演算結果を入力データとして使用することで，CPU 側メモリを介すことなく次の演算処理に移ることができる．

この手法はデータコピーの回数自体の削減を行うことができるため，CPU 側メモリから共有メモリへの入力データコピー時間，共有メモリから CPU 側メモリへの演算結果コピー時間の両方を削減することができる．

本原稿執筆時点では，CPU による入力データの分割処理が行われておらず，回路によるデータ出力順に変化のない層のみでの適用となっている．そのため，合計 83 層ある回路動作対象層のうち 8 層のみの適用となっている．

4.3 回路仕様の変更

使用回路の使用リソースに余裕があることがわかった．そこで，共有メモリ活用の他，回路に設定している各上限サイズの見直しを行い，データの分割回数を削減することで回路の動作回数を減らす．

上限サイズの変更は，使用モデルの入力データサイズを考慮し行った．入力サイズには縦×横が 20, 40, 80, 160, 320 の 5 種類があったため，サイズ上限を 66 から 82 へと増加させた．また，カーネル数上限は 152 から 128 へと減少させた．カーネル分割処理は，分割した回数分，バイアスデータ及び入力データを含むデータ転送が発生し，処理時間が大幅に増加してしまう．そこで，カーネル数を割り切れる 128 を上限に採用することで，分割回数を変化させず，それぞれ BRAM 10%, FF 15% LUT 18%の回路リソースを削減できた．

5 検証方法

検証は，Ultra96-V2 上で回路を用いて YOLOX-s を動作させ，データコピーに要した時間について，検討手法適用前と，共有メモリの活用および上限サイズの変更を行った手法適用後で比較を行う．

6 結果

手法適用前後でそれぞれデータコピーに要した時間を表 2 に示す．

表 2 データコピー時間

	手法適用前	手法適用後
CPUから共有メモリ	1.87s	1.71s
共有メモリからCPU	1.60s	1.49s
合計	3.47s	3.20s

表 2 より，手法適用前と適用後では合計 0.27s 削減されている．また，共有メモリへの書き込み時間が 0.16s，書き出し時間が 0.11s 減少した．

7 まとめ

本発表では，回路動作時に発生するデータ共有において，特にデータコピー時間の削減手法を検討した．そして，共有メモリの活用方法を中心に，データコピー回数の削減について考え，共有メモリの活用が，入力データのコピーに対し有効であることを確認した．今後の展望として，共有メモリ活用の適用範囲拡大を行っていく．

参考文献

[1] 吉田裕紀, 中西知嘉子, “物体検出モデル「YOLOX」のエッジデバイス上での動作高速化手法の検討”, 信学技報, vol. 123, no. 71, RECONF2023-4, pp.17-22, (2023)
[2] Ultra96-V2, https://japan.xilinx.com/products/boards-and-kits/1-vad4rl.html
[3] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, Jian Sun, “YOLOX: Exceeding YOLO Series in 2021”, arXiv preprint arXiv:2107.08430 (2021)
[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] 大戸彰馬, 中西知嘉子, “推論処理における畳み込み処理の回路化の検討”, 電子情報通信学会総合大会(2022).
[7] 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).

† 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