C-011

FPGA を使用した物体検出モデル「YOLOv7」の高速化手法の検討

Investigation of speed-up method of object detection model "YOLOv7" using FPGA

舩橋 駿介‡　　中西 知嘉子‡
Shunsuke Funahashi　Chikako Nakanishi

1. はじめに

近年，エッジ AI が注目を集めている．エッジ AI とは携帯電話や車載機など比較的安価でかつ小さい機器に搭載された AI である．推論に通信を必要としないため，通信による遅延時間が無く，セキュリティが高く，使用環境を選ばないなど様々な利点がある．しかし，エッジ AI を搭載する機器は性能が低く，精度と処理時間を両立することは難しい．そこで我々は，CPU と FPGA が同一チップ上に存在する Soc FPGA を採用し，ソフトウェア(CPU)と回路(FPGA)を協調動作させることでエッジ AI を高速化させる手法を検討している．物体検出モデルの 1 つである YOLOv7[1]を対象に，先行研究[2]で開発された畳み込み処理回路をベース回路とし，ベース回路を YOLOv7-tiny に特化させ，高速化を試みた[3]．その結果データ転送に改善の余地があることが分かった．そこで，本発表では更なる高速化のための手法を提案する．

2. 使用機器・使用ネットワーク

2.1 使用機器・使用ツール

本節では本研究で使用した機器やツールについて述べる．使用機器は Avnet 社からリリースされた Ultra96-V2[4]である．使用ツールは C++言語を用いて高位合成[5]を行う Vitis HLS 2020.2 と回路設計に用いる Vivado Design Suite 2020.2 である．

2.2 使用ネットワーク

高速化の対象とする深層学習モデルは YOLOv7 の中で最も演算量が少ない YOLOv7-tiny である．回路と CPU の協調動作を行うため，用いる高位合成ツールの Vitis HLS に合わせ，C++言語で推論処理を行うことができる Ceras[6]というライブラリを使用する．

YOLOv7-tiny を Ultra96-V2 の CPU のみで実行した際の処理時間を表 2.2.1 に示す．

表 2.2.1　CPU のみの実行時間

名称	処理時間(ms)	割合
推論全体	180134	100%
Conv2D 層	177157	98.35%
LeakyReLU 層	658.57	0.37%
MaxPooling 層	1505.63	0.84%
Concatenate 層	413.6	0.23%
Resize 層	51.57	0.03%

表 2.2.1 より，Conv2D 層の処理に 1 番時間がかかっていることが分かった．そこで，先行研究である畳み込み処理の回路をベース回路とし，YOLOv7-tiny に特化した回路を作成した．YOLOv7-tiny に特化させた回路を用いて CPU と協調動作することで実行したときの処理時間を表 2.2.2 に示す．

表 2.2.2　特化回路の実行時間

名称	処理時間(ms)	割合
推論全体	7845.11	100%
Conv2D 層 +LeakyReLU 層	5328.36	67.92%
MaxPooling 層	1486.8	18.95%
Concatenate 層	412.35	5.26%
Resize 層	51.37	0.65%

表 2.2.2 より，Conv2D 層の処理時間が最も長かった．そこで，Conv2D 層の処理時間の内訳を表 2.2.3 に示す．表 2.2.3 の書き込み，読み出しは CPU から共有メモリへの入力データの書き込みと読み出しを表している．

表 2.2.3　Conv2D 層の実行時間

名称	処理時間(ms)	割合
Conv2D 層 +LeakyReLU 層	5328.36	100%
書き込み	1214.19	22.89%
回路動作	2945.17	55.27%
読み出し	905.37	16.99%

表 2.2.3 より，共有メモリへの書き込みと読み込みに計 2 秒以上かかっていることが分かった．これは Conv2D 層の処理を回路で行うたびに入力データを CPU から共有メモリへ書き込み，計算結果を共有メモリから CPU へ読み出しているためである．そこで，共有メモリへの書き込み，読み出しの回数を減らすことで実行時間の削減を目指す．

3. 設計

3.1 回路のデータ保持

回路が一度に保持することができるデータ量には限度がある．そのため，Conv2D 層の入力データが大きい場合，一度に保持することはできない．もし，入力データのサイズが保持できるサイズを超えてしまった場合は CPU 側で入力データを分割し，回路に転送する必要がある．そこで特化回路が保持することができるデータサイズを，YOLOv7-tiny の入力データの大きさを考慮し 82 から 162 へ変更した．YOLOv7-tiny の，入力データのサイズは 160 以上であれば全て 160 で割り切ることができ，パディング処理に対しても対応できるため，データの分割転送回数を最小限に抑えることができる．

3.2 共有メモリの活用

本研究において、共有メモリは CPU 上にある入力データ等を一時的に保持し，回路へ DMA 転送するために使用される．2.2 節で述べた通り，共有メモリ内のデータは各層で推論が行われるたびに入力データとして書き込まれ，回路から転送されたデータを計算結果として CPU へ読み出される．そのため，データの書き込みと読み出しに時間がかかってしまう．そこで Conv2D 層が連続している際に共有メモリ内部に存在する計算結果を入力データとして回路に転送する．共有メモリ活用のイメージを図 3.2.1 に示す．

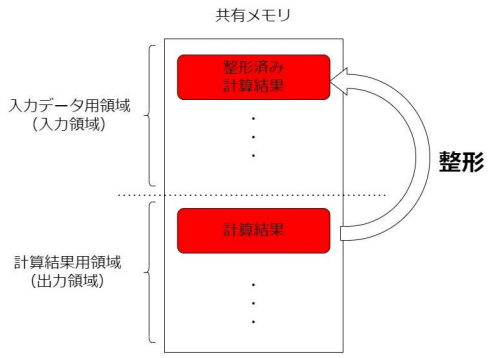

図 3.2.1　共有メモリ内のデータ利用

図 3.2.1 に示したように，共有メモリを回路への入力を保持する領域と回路からの出力を保持する領域に分ける．3.1 節で述べたデータの分割が発生せず，処理をする Conv2D 層の次の層が 1 つの Conv2D 層のみであった場合，出力領域から直接入力領域に計算結果を書き込むことで書き込み，読み出し回数を減らすことができる．一度入力領域に書き込むのはパディング処理を CPU 上で行っており，データの整形が容易になるためである．ただし，パディング処理がない Conv2D 層の入力データに対してはデータの整形を行う必要が無いため出力領域から直接，計算結果を入力データとして回路に DMA 転送する．データ分割が発生した際にこの手法を適用していないのは，入力サイズ分割の場合，共有メモリ内のデータの並び方が変わるためである．チャネル分割の場合，回路内で LeakyReLU 層の処理を行えず，共有メモリ内の計算結果が入力データに適した値とならないため共有メモリを活用できない．また，Conv2D 層の次の層が Conv2D 層以外である場合にこの手法を適用できないのは，他の層の処理を CPU 上で行っているためである．

4. 評価と結果

3 章で述べた高速化手法を適用した結果を表 4.1 に示す．

表 4.1　作成回路の実行時間

名称	CPU(ms)	高速化前(ms)	高速化後(ms)
推論全体	180134	7845.11	7750.79
Conv2D 層 +LeakyReLU 層	177815.57	5328.36	5231.30
書き込み	0	1214.19	1167.42
回路動作	0	2945.17	2942.32
読み出し	0	905.37	860.30

表 4.1 より，全体の推論時間は CPU と比較して約 23.2 倍，Conv2D 層は約 33.9 高速化できた．対して，書き込み，読み出しの時間はいずれもあまり高速化することができなかった．

5. 今後の課題

4 章より，共有メモリへの書き込み，読み出しの高速化はあまり効果が無かった．これは共有メモリの活用を行う際，3.2 節で述べた共有メモリを活用するための条件に当てはまっている Conv2D 層が 1 つのみであったためである．YOLOV7-tiny は E-LAN という 4 つの Conv2D 層の計算結果を Concatenate 層で結合する構造を持つ．半数以上の Conv2D 層がこの構造の一部であるため，共有メモリを活用するための条件に当てはまる Conv2D 層が少なかった．そこで，共有メモリを活用するための条件を増やし，複数の計算結果を出力領域に保持できるように改良する．そして，共有メモリで保持している複数の計算結果を Concatenate 層で結合し，全体の共有メモリへの書き込み，読み出し時間の削減を目指す．また，共有メモリの活用を行う際に共有メモリの出力領域から入力領域への書き込みが発生しているため，書き込み時間をあまり高速化することができなかった．そこで，パディング処理を行う Conv2D 層の計算結果に対しても出力領域から回路へ直接 DMA 転送を行う手法を考える予定である．

6. 結論

本研究では YOLOv7-tiny の特化回路に対し，CPU から共有メモリに対するデータの書き込み，読み出しの時間削減を行った．結果として CPU のみの実行時間に対し，23.2 倍高速化することができた．しかし，データの書き込み，読み出しの時間をあまり削減することはできなかった．

今後の展望として 5 章で述べたデータの書き込み，読み出し時間の更なる削減手法の検討を行う予定である．

参考文献

[1] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao, Institute of Information Science, Academia Sinica, “Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors”, (2022)
[2] 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).
[3] 舩橋駿介,中西知嘉子,” 物体検出モデル「YOLOv7」のエッジ端末での実用化の検討”,信学技報, vol. 123, no. 71, RECONF2023-5, pp. 23-28,(2023)
[4] https://japan.xilinx.com/products/boards-and-kits/1-vad4rl.html
[5] Vivado Design Suite ユーザーガイド 高位合成, 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] https://pytorch.org/docs/stable/generated/torch.nn.LeakyReLU.html

†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