# MMS 留学生レポート（完了報告）
# コンパクト化 AI の FPGA 実装技術修得

第 4 事業部エンベデッドハードウエア技術部 LSI 技術課　岩河 秀知

## 1. まえがき

当社国内留学制度により，2018～2020 年度の 3 年間，三菱電機（株）情報技術総合研究所へ出向し，研究開発を行った。本稿では，出向期間中の研究内容，及び，修得したコンパクト化AI のFPGA 実装技術について述べる。

## 2. 出向の目的

AI 推論は，高い認識精度を実現するために膨大な積和演算を実行する必要があり，リアルタイム性等の観点から，FPGA アクセラレーターにて推論処理を行う要求が増えている。今回の研究開発では，AI の FPGA 実装関連技術を当社に技術蓄積することを目的とし，表 1 に示す主要な FPGA 向けの AI 実装技術修得を行った。その結果，当初計画していた推論分野の技術に加え，その他応用技術も修得した。

表 1. AI 実装技術一覧

（×：未収得，○：修得済み）

| 分野 | 技術項目 | 当社技術修得可否 | |
|---|---|---|---|
| | | 計画 | 実績 |
| 推論 | アーキテクチャの FPGA 実装 | ○ | ○ |
| | 推論処理の高速化 | ○ | ○ |
| | 推論回路の小規模化 | ○ | ○ |
| | コンパクト化<br>（枝刈り・量子化） | ○ | ○ |
| その他 | その他応用技術<br>（ソフトウエア IP 化・自動生成環境の構築など） | － | ○ |
| 学習 | アーキテクチャの FPGA 実装など | － | × |

## 3. 活動内容

### 3.1 開発概要

筆者は，出向先で AI 推論回路の FPGA 実装容易化技術の開発を行った。これは，AI 推論処理を高速かつ，小規模で容易に FPGA 実装し，搭載可能な FPGA を選定することを目的としている。

FPGA とは Field Programmable Gate Array の略称であり，HDL などのプログラム言語を用いて回路をプログラムすることができるという特徴がある。

### 3.2 FPGA 向け AI 推論回路開発の現状

AI 推論回路の FPGA 実装手法として，RTL/高位設計を用いた開発がある。しかし，本設計手法は FPGA デバイスの選定，AI 推論機能の回路実装方法，実装アーキテクチャの検討などの高い専門知識が必要とされ，設計難易度が高い。この設計難易度を軽減する手法として FPGA 実装容易化検討が進められている。

### 3.3 FPGA 実装容易化の課題

FPGA 実装容易化を実現するには以下の課題がある。

(1) 高速かつ小規模な AI 推論処理を行うアーキテクチャを確立すること。
(2) 搭載可能な FPGA を選定するための処理性能，回路規模，メモリ量の見積手法を確立すること。
(3) 専門知識不要で高速かつ小規模な AI 推論回路を設計可能とすること。
(4) 要求性能，ネットワーク規模が異なるアプリケーションでも，短期間で AI 推論回路を設計すること。（出向先での開発実績では，画像認識アプリで仕様確定～RTL 作成まで約 40 人日）。

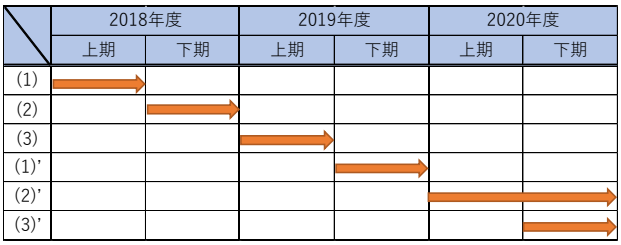
### 3.4 活動実績

FPGA 実装容易化の課題を解決するために，以下の活動を行った。

(1) FPGA 向けパイプライン型（詳細は後述）AI 推論回路のアーキテクチャ設計・検討（課題①）
(2) (1) の性能見積・FPGA 実装（課題②）
(3) (1) のソフトウエア IP の設計（課題③），C/C++ソースコード自動生成環境の構築（課題④）

(1)’～(3)’は FPGA 向け時分割型（全畳み込み層共通の演算器を用いる構成）AI 推論回路で (1)～(3) と同様の研究開発を行った。活動実績を表 2 に示す。

表 2. 活動実績




本稿では，学会発表[1][2]を行ったパイプライン型 AI 推論回路について示す。

# 4. 課題と対策

## 4.1 アーキテクチャ設計(課題①)

AI 推論機能の FPGA 実装手法は複数存在し，アプリケーションによって，適正のある FPGA 回路アーキテクチャは異なる。各アーキテクチャにおいて，最適な設計を検討し，高速，かつ，小規模な AI 推論処理を行うアーキテクチャを確立する必要がある。そこで，代表的な AI 推論回路アーキテクチャであるパイプライン型，共通の演算器を全層で実装する時分割型について調査し，アーキテクチャの設計を行った。図 1 にアーキテクチャの比較を示す。

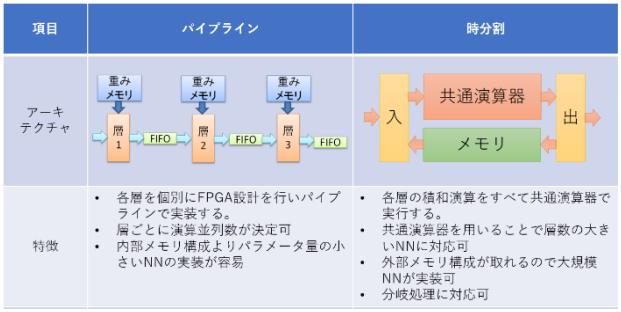



図 1. アーキテクチャ比較

以下で説明するパイプライン型アーキテクチャは，外部メモリを使用しない，比較的小規模なニューラルネットワーク(以降 NN という)向きのアーキテクチャである。

### 4.1.1 基本アーキテクチャ

図 2 にパイプライン型アーキテクチャの構成を示す。

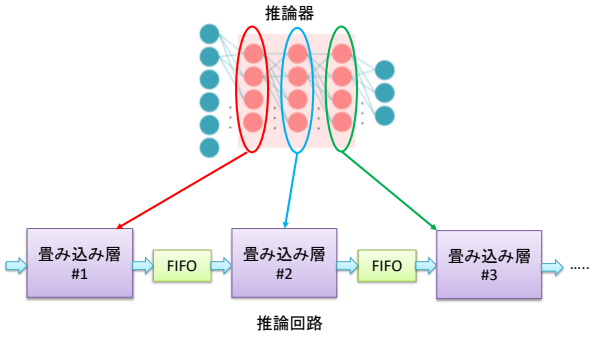



図 2. パイプライン型アーキテクチャ構成[1]

本アーキテクチャは，NN の各中間層に畳み込み演算モジュールを配置し，並列実行の構成とする。加えて，層間の遅延量を調整する FIFO を実装することで，畳み込み層間でデータ待ち状態を抑制し，高速化を実現する。

### 4.1.2 畳み込み層アーキテクチャ

畳み込み層の詳細について図 3 に示す。図 3 のとおり畳み込み層は(1)～(4)から構成されている。図中の(1)～(4)について説明する。

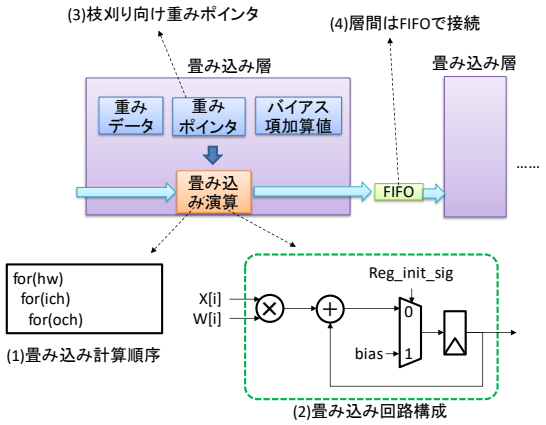



図 3. 畳み込み層のアーキテクチャ詳細[1]

(1) 畳み込み計算順序

畳み込み処理は，入力チャネル方向(ich)，出力チャネル方向(och)，空間方向(hw)の 3 次元に対して積和演算を行う。この 3 次元の計算順序を変えることで回路規模・メモリ量に影響がある。表 3 のとおり，No.6 の ich=>och=>hw の計算順序がメモリ，回路規模とも小規模に実装できることが解り，パイプライン型アーキテクチャに採用した。

表 3. 畳み込み次元と回路規模・メモリ量への影響

| No | 演算順序 | メモリ | 回路規模 |
|---|---|---|---|
| 1 | hw ⇒ ich ⇒ och | ○ | × |
| 2 | ich ⇒ hw ⇒ och | △ | ○ |
| 3 | hw ⇒ och ⇒ ich | ○ | × |
| 4 | och ⇒ hw ⇒ ich | △ | × |
| 5 | och ⇒ ich ⇒ hw | ○ | × |
| 6 | ich ⇒ och ⇒ hw | ○ | ○ |

(2) 畳み込み回路構成

畳み込み演算を行う場合，一般的に乗算後にバイアス項加算を行うが，乗算器，累積加算器に加え，加算器が必要となり，大規模化の要因となる。本構成では，バイアス項を初期値とし，入力選択信号(Reg_init_sig)で入力値を選択することで，加算器を追加使用しない構成で小規模な回路を実装することができた。

(3) 枝刈り向け重みポインタ(コンパクト化)

枝刈りは，処理性能向上，回路規模削減，メモリ量削減策として，従来は畳み込み層のノードを削減していたが，推論精度への影響が大きい枝を刈る可能性があった。そこで，枝刈りの条件に推論精度に影響のある積和演算を識別する重みポインタを付加することで，処理性能同等時の推論精度向上を図った。図 4 に重みポインタのイメージを示す。

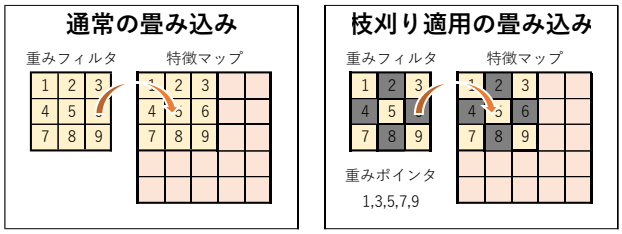



図 4. 枝刈り向け重みポインタ

図 4 の例に示すように，通常の畳み込みでは，1～9 の場所全てに対して畳み込み演算を行うが，枝刈り時は 1, 3, 5, 7, 9 のみ演算を行う。本枝刈り手法は，枝刈り量に伴い格納している内部メモリ量の削減と，推論処理時間の高速化を同時に実現できるアーキテクチャである。

また，ここで用いる重みには量子化を適用することで，推論精度が仕様を満たす範囲でビット幅を削減している。これにより，回路規模，内部メモリが削減でき，小規模に回路実装できる。

(4) 層間 FIFO

層間のデータ受け取り待ちを解消するための FIFO である。畳み込み層間で処理時間に乖離がある場合，データ受け取り待ちが発生する。層間 FIFO を実装することで，前の畳み込み層が出力した値を FIFO に取り込み，畳み込み演算を継続することができるため，データ受け取り待ちを解消することができる。

### 4.1.3 アーキテクチャ設計まとめ

以上の基本アーキテクチャ，畳み込み層アーキテクチャの設計を通じて修得した，高速化，小規模化，コンパクト化技術について表 4 に示す。

表 4. 高速化・小規模化・コンパクト化修得技術

| |
|---|
| 高速化 |
| 各畳み込み層をモジュール化・並列実行する |
| 層間 FIFO を用いてデータ受け取り待ち遅延を解消する |
| 小規模化 |
| 各畳み込み層の演算並列数に合わせ回路規模を最適化する |
| 回路規模・層間 FIFO が最小となる演算順序(ich=>och=>hw)とする |
| コンパクト化(枝刈り) |
| 枝刈り率に応じ重み量を削減する重みポインタ構成とする |
| 枝刈り率に応じ処理性能が高速化される重みポインタ構成とする |
| コンパクト化(量子化) |
| 量子化ビット幅に応じて回路規模，内部メモリ量が削減される構成とする |

表 4 に示すとおり，国内留学時の目標であった技術を修得し，FPGA 実装ガイドラインの作成，推論アーキテクチャの FPGA 実装を実現した。

## 4.2 性能見積手法の検討(課題②)

4.1 章で説明したパイプライン型アーキテクチャを FPGA に実装するためには図 5 の手順で実装する。

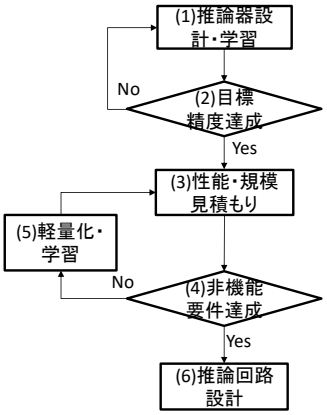



図 5. パイプライン型アーキテクチャ実装フロー例[1]

(1), (2)は 4.1 章のアーキテクチャ設計と，学習である。(3)～(6)は所望の FPGA に実装するための性能・回路規模見積と推論回路設計である。なお，国内留学では推論技術修得を優先するため，(1)，(5)の学習は修得技術の対象外とした。

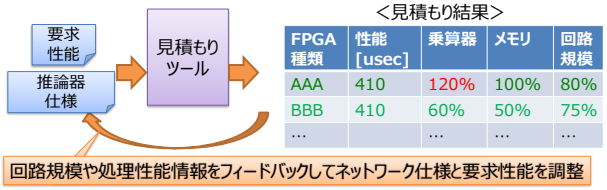



図 6. 見積ツールイメージ[1]

性能見積の検討では，図 6 のとおり見積ツールを作成した。このツールは要求性能・推論器仕様(ネットワーク仕様)を入力とすることで，小規模な演算並列数の組合せを決定し，実装可能な FPGA 種類を出力することができる。

ツール構成は処理性能，乗算器(DSP)・内部メモリ，回路規模(LUT 数)見積の 3 つからなる。

### 4.2.1 処理性能見積

以下の式を満たすように要求性能を満足する必要最小限な演算並列数を求める。

Max(各層の畳み込み性能) ＜ 目標スループット[cycle]

### 4.2.2 DSP/内部メモリ見積

DSP とは，FPGA 内に搭載されている演算処理プロセッサーである。DSP 数は，畳み込み層の演算並列数と，量子化ビット幅から求められる。内部メモリは，重み，重みポインタ，バイアス項の格納に用い，各データの容量と量子化ビット幅，各層の演算並列数から決定する。

### 4.2.3 LUT 見積

推論回路全体の LUT(Look Up Table)は，ツールの最適化により異なってくるため，算出式ではなく畳み込み 3x3，1x1(以降 Conv3x3，1x1)，Max Pooling の層について，何通りかの演算並列数の論理合成結果を元に，各層の演算並列数から LUT を求める。この演算並列数の全層での総和を求めることで推論回路全体の LUT を算出する。

### 4.2.4 性能見積手法の検討まとめ

4.2 章まで説明してきた，パイプライン型アーキテクチャの設計，性能見積手法をガイドライン化した。この手法を用いることで，要求性能を満たす高速かつ小規模な AI 推論回路を実装できる。

## 4.3 高位ソフトウエア IP(課題③)

AI 推論回路の実装は，従来の FPGA 設計以外の専門知識が必要となり難易度が高い。そこで，専門知識を必要とせず，高速かつ小規模な AI 推論回路を設計可能とするために，高位ソフトウエア IP を作成した。

これは，VivadoHLS 向けの C/C++言語を用いて設計し，任意の演算並列数で Conv3x3，1x1，Max-Pooling 層を実現する。本 IP では重み，ポインタ，バイアス項などの内部メモリ構成を指定する疑似コードを「#pragma」を用いて実装している。疑似コードの実装例を畳み込み処理を元に以下に示す。

【畳み込み処理の疑似コード(2)】

```
// CONV_N : 入力チャネル数 などの CNNパラメータ
for (int c = 0; c < CONV_N ; c++ ){
  // OP:並列演算数 LOOP_UNROLL指定
  for ( int op = 0; op < OP ; op++ ){
    //積和演算
```

### 4.3.1 高位ソフトウエア IP まとめ

高位ソフトウエア IP は，演算並列数等をパラメータ化し，高位合成ツール(VivadoHLS)に所望の CNN（convolutional neural network：畳み込みニューラルネットワーク）パラメータと演算並列数でパイプライン型アーキテクチャを生成することができる。

## 4.4 アーキテクチャ自動生成環境構築(課題④)

AI 推論回路を実装するためには，要求性能，ネットワーク規模が異なるアプリケーションごとに，演算並列数，メモリ構成を決定し，回路実装を行う必要がある。この仕様確定から RTL 生成の期間は，FPGA 向け AI 推論回路の専門家でも，40 人日程度の時間がかかる。

専門家以外が設計した場合は，更に期間が必要になり期間短縮する必要がある。そこで，ここまで設計したアーキテクチャ，性能見積手法，高位ソフトウエア IP を組み込んだ，パイプライン型アーキテクチャ自動生成環境を構築した。図 7 に自動生成環境を示す。

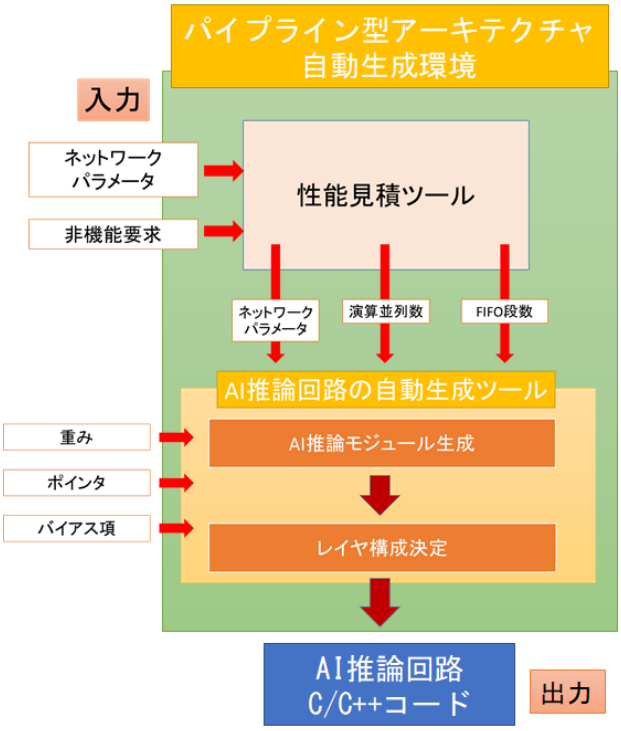



図 7. パイプライン型アーキテクチャの自動生成環境

この自動生成環境は，性能見積ツールと，AI 推論回路の自動生成ツールから構成される。入力はネットワークパラメータと非機能要求(処理性能，回路規模など)であり，出力は VivadoHLS 向けの AI 推論回路 C/C++コードである。ここで，重み，バイアス項，ポインタなどのパラメータを設定することで，C/C++コードが生成される。

### 4.4.1 性能見積ツール

性能見積ツールとは，4.2 章で説明した処理性能，回路規模，内部メモリ量の見積手法を用いて，要求性能を満たし，回路規模が小さい演算並列数，FIFO 段数を見積もるものである。本機能を用いることで，要求性能を満たすために必要な演算並列数(DSP 数)，回路規模(LUT 数)，内部メモリ量(BRAM 数)を見積もることができ，実装対象となる FPGA を選定するのに必要な情報が得られる。

### 4.4.2 AI 推論回路の自動生成ツール

このツールは，AI 推論モジュール生成部と，生成した AI 推論モジュールをネットワークパラメータに従い組み合わせるレイア構成決定部から構成される。パイプライン型アーキテクチャ自動生成ツールの概念図を図 8 に示す。

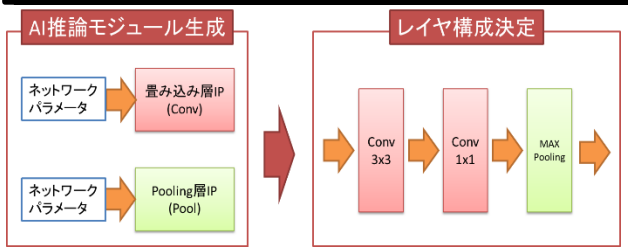



図 8. パイプライン型アーキテクチャ
自動生成ツールの概念図[2]

AI 推論モジュール生成部では任意の並列数に対応した高位ソフトウエア IP(畳み込み層 IP, Pooling 層 IP)にネットワークパラメータと，演算並列数を設定することで，AI 推論モジュール(Conv3x3, Conv1x1, Max-Pooling)を生成できる。レイヤ構成決定部では，AI 推論モジュール生成部で生成したモジュールをネットワークパラメータに従い組み合わせる。層間はデータ受け取り待ちを解消し，パイプライン実行を可能とする FIFO を挿入する。

### 4.4.3 AI アーキテクチャ自動生成環境まとめ

本自動生成ツールを用いることで，パラメータを設定するだけで，要求仕様を満たす小規模な回路を生成することが可能となった。本ツールを用いることで，性能と回路規模のトレードオフを自動で最適化することができ，実装期間も 40 人日から 1 時間以下と今回は大幅に削減することができた。

## 5. 成果

検討した AI 推論アーキテクチャ，見積手法，高位ソフトウエア IP 及び，自動生成技術を適用し，以下を実現した。AI 推論アーキテクチャについては関連特許を 4 件出願，性能見積[1]，ソフトウエア IP[2]，自動生成環境[2]については学会にて発表済。

(1) 仕様確定から RTL 設計までの実装期間短縮

FPGA 向け AI 推論回路の専門家が，実装に 40 人日を要する回路を，専門知識がないユーザーでも 1 時間以下で実装可能となる開発環境を構築した。

(2) FPGA デバイスの自動選定

ネットワーク仕様，非機能要求を入力として，実装可能な FPGA を自動選定するツールを開発した。

本アーキテクチャは，従来の監視カメラで取得した画像の認識，物体検出への適用はもちろん，コスト面で AI の適用が難しかった家電，エレベーター，高精度地図などへの適用が考えられる。また，国内留学・出向中に表 5 に示す技術を修得した。

表 5. AI 推論アーキテクチャ修得事項

| 修得技術 |
|---|
| アーキテクチャの FPGA 実装 |
| 推論処理の高速化 |
| 推論回路の小規模化 |
| コンパクト化(枝刈り・量子化) |
| 処理性能・回路規模見積 |
| 高位ソフトウエア IP 化 |
| アーキテクチャ自動生成 |

国内留学当初に予定していた AI の FPGA 実装を当社で受託可能とするための基礎技術(FPGA 実装，高速化，小規模化，コンパクト化)に加え，処理性能・回路規模見積，高位ソフトウエア IP 化，アーキテクチャ自動生成技術を修得することができた。

## 6. むすび

3 年間の国内留学・出向を終え，留学の目的である高速かつ，小規模な AI 推論回路の FPGA 実装技術の修得は達成できた。加えて，高位設計を用いた AI 推論回路開発フローについても確立できた。今後は，修得した知見や検討した手法を，実オーダー，勉強会を通じて技術展開することで，当社内でのコンパクト化 AI の FPGA 実装技術を高めていく所存である。

## 謝　　辞

今回の執筆にあたり，ご指導とご助言を頂いた三菱電機 (株) 情報技術総合研究所 コネクテッドインダストリーシステム技術部 LSI 設計技術G各位に深く感謝の意を表する。

## 商　　標

(1) Maisart は，三菱電機(株)の登録商標である。
(2) Vivado HLS は，米国及びその他各国の Xilinx 社の商標である。

## 参 考 文 献

(1) 山本亮, 岩河秀知, 小川吉大 : FPGA 向け Deep NN 推論回路の性能と回路規模見積, 電子情報通信学会大会(2021)
(2) 岩河秀知, 山本亮, 杉原堅也, 小川吉大 : FPGA 向け IP を用いた AI 推論回路生成環境の構築, 電子情報通信学会大会(2020)