アプリケーションノート

ルネサス RA8 ファミリ

RA8 MCU における Ethos-U NPU の使用方法

はじめに

Arm Ethos-U ニューラル・プロセッシング・ユニット（NPU）は、マイクロコントローラ上で機械学習（ML）の推論処理を高速化するために設計された、超低消費電力のNPUです。
Renesas RA8シリーズのMCU（RA8P1）には、このEthos-U NPUが内蔵されています。

Ethos-U NPUはホストCPUと連携して動作し、コンピュータビジョン、音声認識、異常検知といったアプリケーションにおいて、効率的なAI／MLの高速化を提供します。

機械学習の計算処理をCPUからEthos-U NPUへオフロードすることで、システム全体の性能と電力効率が向上します。これにより、大きな消費電力の増加を伴うことなく、エッジデバイス上でのAI活用が可能になります。

ターゲットデバイス

- RA8P1

必要なリソース

開発環境およびソフトウェア

- Renesas e2 studio （RUHMI Frameworkをインストール済み）
- Renesas Flexible Software Package （FSP）v6.0.0 (https://github.com/renesas/fsp)

前提条件および対象読者

本アプリケーションノートの読者は、Renesas e² studio の利用経験があることを前提としています。

また、事前知識として、以下の参考資料を参照してください。

- RA Flexible Software Package Documentation: Ethos-U (rm_ethosu)
- Arm Ethos-U55 NPU Technical Reference Manual
- Renesas RUHMI Framework Quick Start Guide（R11QS0065）

本アプリケーションノートの対象読者は、Ethos-U NPU を搭載した Renesas RA MCUを用いてAIアプリケーションを現在開発している、または今後開発を予定している方です。

目次

1. Ethos-U NPUについて

多くの機械学習（ML）アプリケーションは、ニューラルネットワーク（NN）の推論処理を中心に構成されています。
これらの処理は汎用プロセッサ上でソフトウェア実行することも可能ですが、ハードウェアアクセラレータを活用することで、性能を大幅に向上させることができます。

Ethos-U NPU は、小型かつ低消費電力なプロセッサとして設計されており、機械学習のニューラルネットワーク推論において、処理時間の短縮とメモリ使用量の削減を実現します。

Arm は Ethos-U NPU のハードウェアRTL（Register Transfer Level）設計を提供しており、MCU ベンダはこれを自社マイコンに組み込むことで、低消費電力を維持したまま AI/ML 処理能力を強化できます。

ルネサスでは、Ethos-U NPU の性能を最大限に引き出すため、密結合メモリ、キャッシュ、効率的なデータパス、AI 向けソフトウェアライブラリを含むハードウェアおよびソフトウェアの両面を最適化しています。

また、Ethos-U NPU にはベンダが設定可能なパラメータが存在しますが、ルネサスは RA8P1 向けに最適な構成を選定しており、エッジ AI アプリケーションに適した高性能かつ低消費電力なシステムを実現しています。

表1 RA8P1上のEthos-U55 NPU仕様

項目	仕様
1サイクルあたりの並列8×8 MAC演算数	256 MACS/サイクル
共有RAM	48KB

NPU は、レベルトリガ型の割り込み信号を介して、ホスト CPU や MCU 内部メモリと通信します。この割り込み信号は、NPU が処理を完了したとき、または エラーが発生したときにアサート（High になる）されます。
NPU ドライバは APB バスを使用して NPU の各種設定を行います。一方、AXI バスは、以下のデータ転送に使用されます。

- コマンドストリーム
- 重み（Weight）およびバイアス（Bias）データ
- スクラッチ用テンソル
- 入力データ
- NPU からの出力データ

用途に応じて APB と AXI の各バスを使い分けることで、効率的かつ高速な NPU 制御とデータ処理を実現しています。

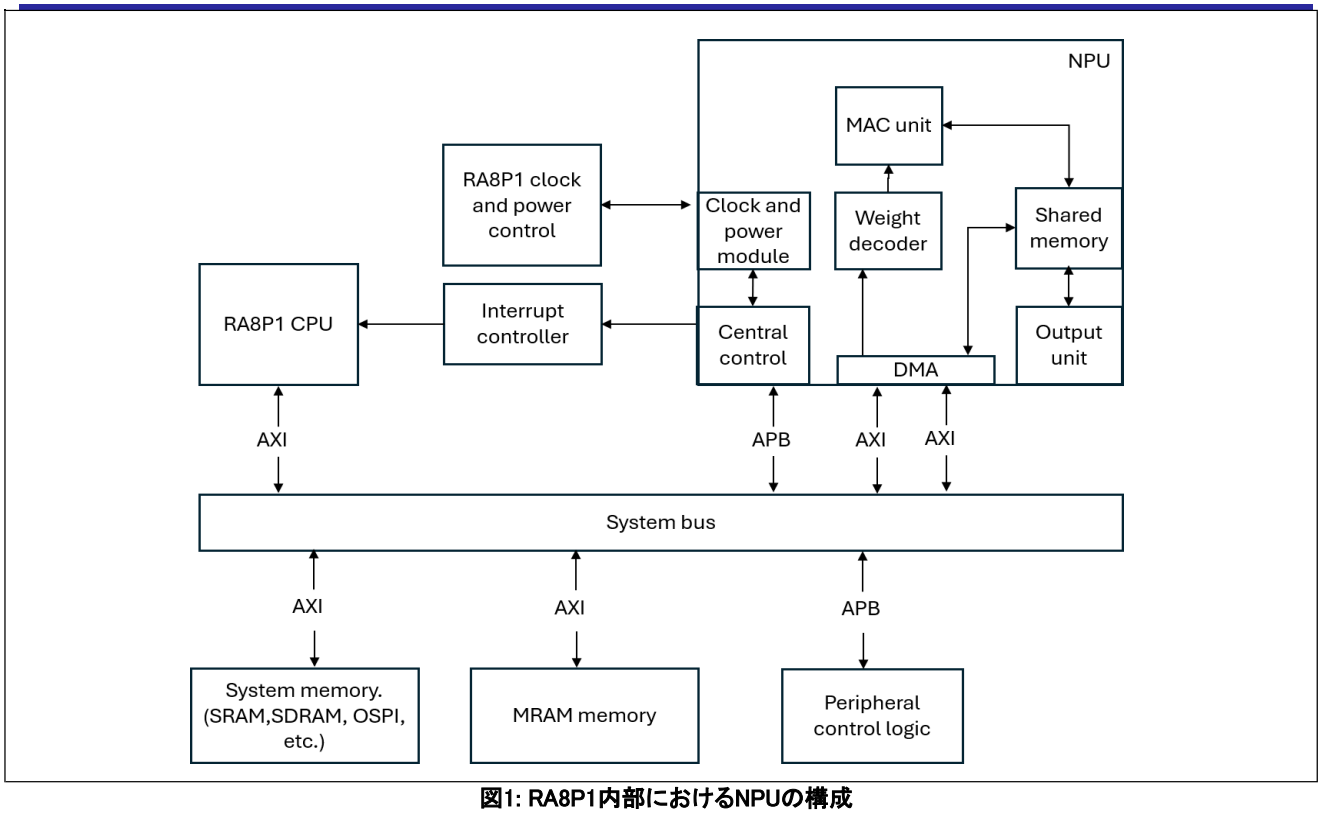


図1: RA8P1内部におけるNPUの構成

NPUの構造や設定方法の詳細については、Arm NPU Technical Reference Manualを参照してください。
Arm Ethos-U55 NPU Technical Reference Manual

2. Ethos-U NPUへのAIモデルの展開

図2は、AIモデルを展開するために必要となる主要なソフトウェア構成を示します。
浮動小数点（float）モデルをNPUで使用する場合は、まずモデルを量子化する必要があります。

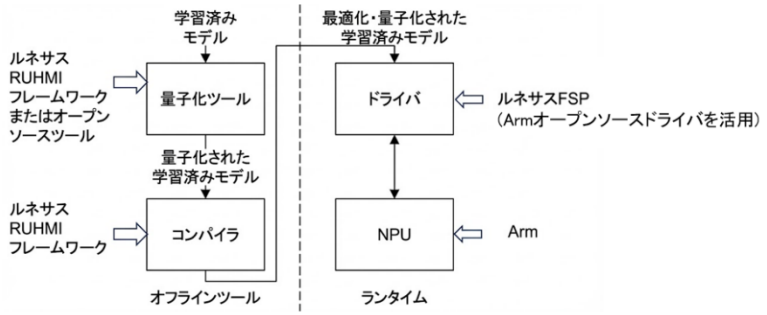


図2: NPU利用時のソフトウェアスタック

推論処理の実行時には、アプリケーションがドライバを呼び出し、NPU に対して コマンドストリームの格納場所を伝え、ニューラルネットワークの実行を開始させます。

コマンドストリームには、NPUが自律的に演算を行うために必要な処理手順が記述されており、NPU はこのコマンドストリームに従って、CPU の介入なしに推論処理を進めます。推論処理が完了すると、NPU は 割り込み（IRQ）を発行し、ドライバに処理完了を通知します。

ArmはEthos-U NPU向けのツールを提供していますが、RenesasはArmのツールをベースに、さらに高度な機能を備えたツール群を提供しています。

- **Renesas RUHMI Framework**
  GUIベースのツールであり、モデルのコンパイル時にさらなる最適化を施すことで、Ethos-U NPUによる推論パフォーマンスを向上させます。また、CPUを用いた推論のコンパイル最適化にも対応しています。なお、本フレームワークにはEdgeCortix® MERAの技術が採用されています。
- **Renesas Flexible Software Package（FSP）**
  FSP v6.0.0以降では、Ethos-Uアクセラレーション対応のTensorFlow Lite Microが統合されており、AIアプリケーションの開発を容易にします。

Renesasが提供する高性能なAIモデルコンパイラおよびEthos-Uドライバは、モデル推論性能を最大限に引き出します。
オープンソースを用いたモデルコンパイル方法については、「Appendix」章を参照してください。

3. 組み込みアプリケーション向けのモデルの準備

3.1 Ethos-U NPUがサポートする演算子

ArmのGitHubリポジトリに含まれる SUPPORTED_OPS.md ファイルには、Ethos-U NPU上で実行可能な TensorFlow Lite（TFLite）演算子の一覧と、各演算子に対する制約条件がまとめられています。
演算子がこれらの制約を満たさない場合、その演算子はNPUではなくCPU上で実行されるように割り当てられます。また、一覧に含まれていないTFLite演算子についても、変換は行われずCPUで実行されます。

この情報は、対象となるニューラルネットワークモデルがEthos-U NPUへ移植可能かどうかを判断する際の指針として利用できます。

Ethos_NPU_Supported_Operators_Through_Vela

Ethos-U NPUによるニューラルネットワーク処理のアクセラレーションを実現するために、TensorFlow Lite Micro（TFLM）では「カスタム演算子」と呼ばれる仕組みが提供されています。
演算子をEthos-U NPUへ割り当てるとは、コンパイルプロセス（本章で後述）において、その演算子をNPU向けのカスタム演算子へ変換することを意味します。

RUHMIコンパイラは、ニューラルネットワークモデルのコンパイル時にモデルを解析し、Ethos-U NPUによる高速化が可能な演算子の組み合わせを抽出します。
抽出された演算子群は、NPU向けのカスタム演算子としてまとめられ、NPUへ処理が割り当てられます。

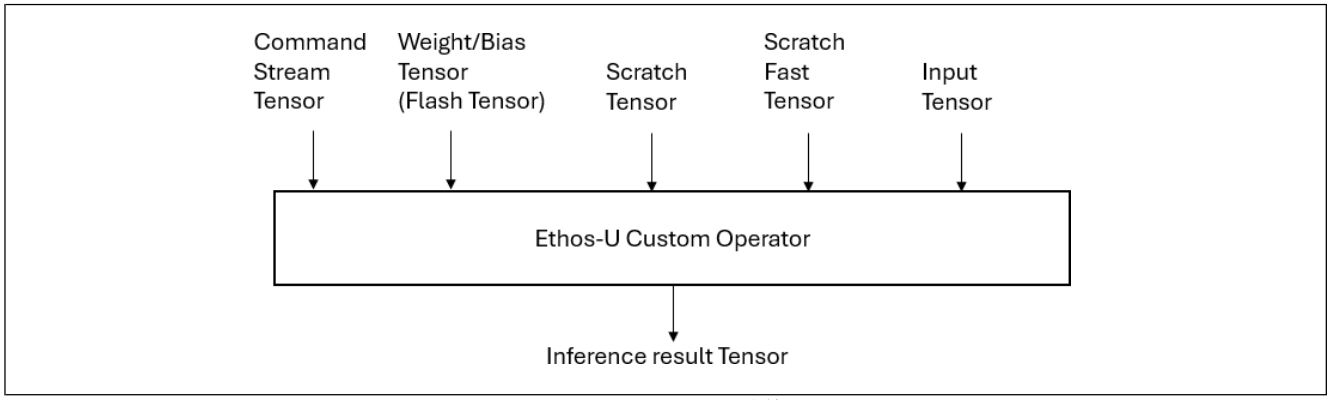


図3: Ethos-Uカスタム演算子

以下のArmリファレンスマニュアルには、Ethos-U NPUでサポートされるデータ型や、NPUの機能を組み合わせることで生成される追加の演算子について説明されています。

Arm Ethos-U55 NPU Technical Reference Manual

Ethos-U NPUを用いてモデルの推論を実行するためにモデルを移植する際には、必要に応じてオープンソースツールのNetronを使用し、元のモデルに含まれる演算子を可視化することができます。
このツールは、TensorFlow Lite（.tflite）、ONNX（.onnx）、PyTorch（.pt、.pth）、Keras（.h5）など、さまざまなフレームワークに対応しています。Netronを用いることで、浮動小数点モデルや量子化済みモデルを表示し、AIモデル内にどのような演算子が含まれているかを確認できます。

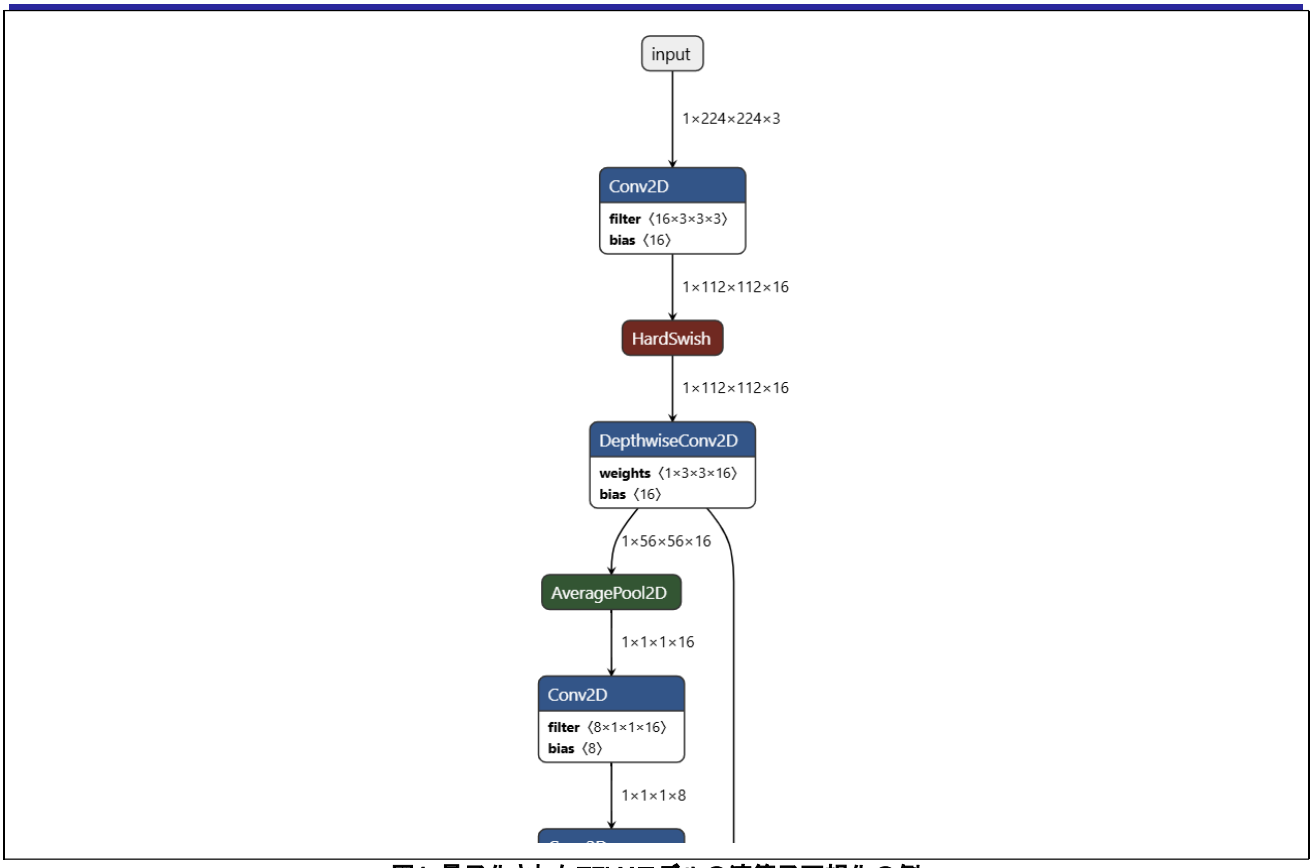


図4: 量子化されたTFLMモデルの演算子可視化の例

3.2 Ethos-U NPUが対応するモデル

Ethos-U NPUは、8ビットおよび16ビット整数で量子化された畳み込みニューラルネットワーク（CNN）および再帰型ニューラルネットワーク（RNN）を対象としています。

NPUが扱う重みデータは、8ビット整数形式に対応しています。

Arm Ethos-U NPUは、複数のモデル形式に対応しています。Armのツールチェーンと統合されているため、TensorFlow Lite（TFLite）はネイティブにサポートされています。
ONNX形式の浮動小数点モデルは、TFLite形式へ変換した後に量子化およびコンパイルを行うことで、Ethos-U NPU上で効率的に実行できます。同様に、PyTorchで作成された浮動小数点モデルについても、TFLite形式またはONNX形式へ変換した後、Ethos-U NPU向けにコンパイルすることが可能です。

Armは、Ethos-U55向けに8ビット整数で量子化されたTFLiteモデルを公開しています。これらのモデルは、NPUを用いた推論の実行に利用できます。

Arm ML Zoo https://github.com/ARM-software/ML-zoo

これらのモデルは .tflite 形式のファイルで提供されており、組み込み機器やモバイルデバイスでの高速な推論を目的とした TensorFlow Lite モデルです。FlatBuffer形式で格納されているため、軽量で高速に読み込むことができます。
これらのモデルはRUHMI Frameworkへの入力として使用され、最終的に組み込みシステムへ組み込むための .c / .h ファイル にコンパイルされます。

3.3 AIモデルのコンパイル

3.3.1 全体の流れ

モデルのコンパイル処理では、主に以下の処理が行われます。

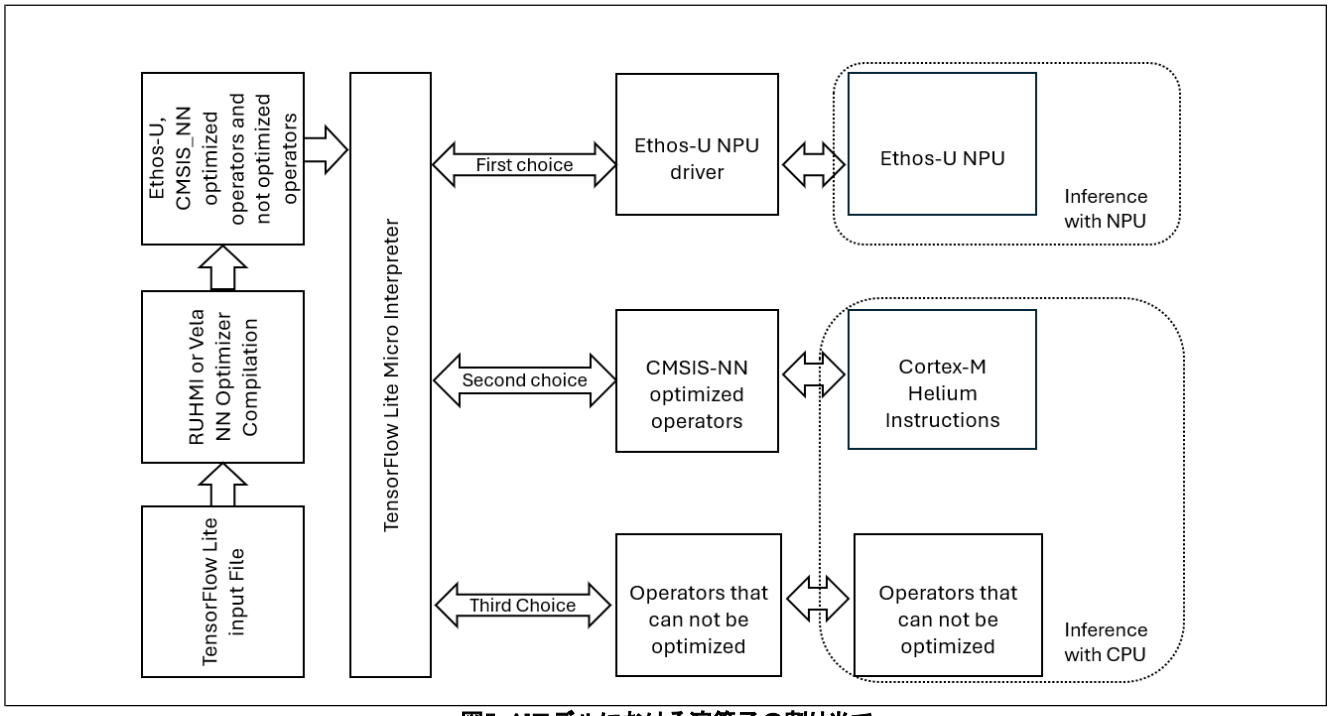


図5: AIモデルにおける演算子の割り当て

コンパイル時の主な処理内容：
　モデルのコンパイル時には、コンパイラによって次の処理が行われます。

1. モデル内の各演算子（レイヤ）を順に確認
2. 演算子がEthos-U NPUでサポートされている場合、その演算子はEthos-U NPUで実行されるように割り当て
3. Ethos-U NPUではサポートされていないが、CMSIS-NNでサポートされている演算子については、CMSIS-NNを用いて実行
   → CMSIS-NNをHelium（例：Cortex-M55、Cortex-M85）対応プロセッサ向けにコンパイルした場合、これらのベクトル命令を活用することで、AI推論時の性能および電力効率を大幅に向上させることができます。
4. Ethos-U NPUおよびCMSIS-NNのいずれでもサポートされていない演算子については、TensorFlow Lite Micro（TFLM）のリファレンスカーネルを使用し、CPU上で実行

3.3.2　ルネサスRUHMI Frameworkの使用

RUHMI Framework は、Vela ツールによる結果と比較して、より最適化された高い性能を提供します。
RUHMI Framework を e² studio 環境にインストールするには、「Renesas RUHMI Framework – Quick Start Guide (RA8P1)」に従ってください。

RUHMI Framework のGUI画面で、「Select Sample AI Application」ボタンからサンプルプロジェクトを選択できます。また、独自のAIモデル向けに新たにプロジェクト展開・コンパイルする場合は、「Use Your Project & AI Model」を選択します。これら 2 つのオプションの使い方については、RUHMI Framework クイックスタートガイドを参照してください。

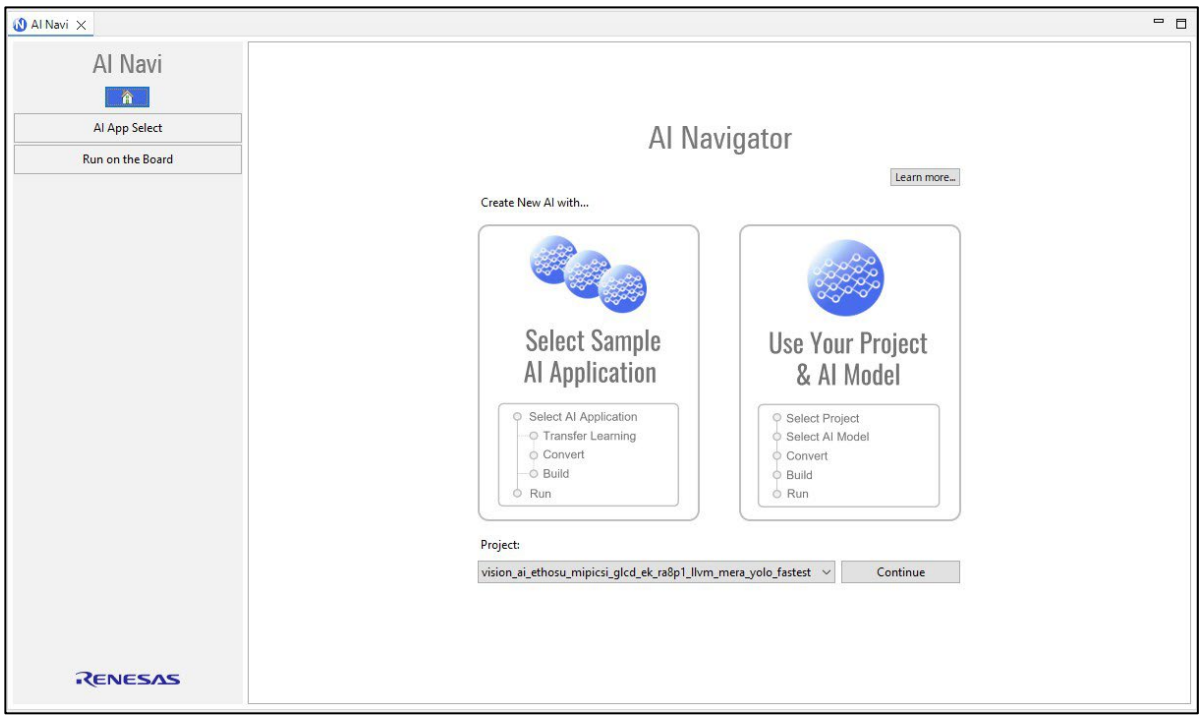


図6: RUHMIツール画面

以下はEthos-U NPU推論のためのRUHMI Framework出力の例です。

以下に、RUHMI によるコンパイル結果の例を示します。コンパイル中に、SRAM、SDRAM、および Flash（MRAM）のメモリ使用量が算出され、その結果が出力ファイルとして提供されます。
RUHMI Framework を使用する場合、現在 ONNX および PyTorch のフロントエンドは Quantizer フローでのみ使用することを想定しています。出力ファイルの利用方法の詳細については、RUHMI Framework クイックスタートガイドのドキュメントを参照してください。

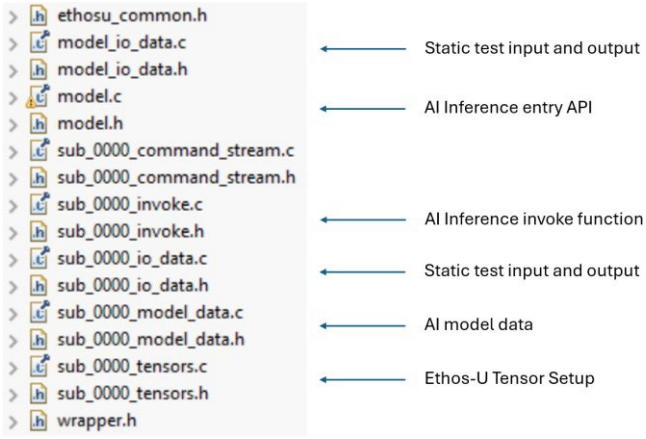


図7: Ethos-U を用いた MobileNet V1（0.25）の推論コンパイル結果

以下に、CPU 推論時における RUHMI Framework の出力結果の例を示します

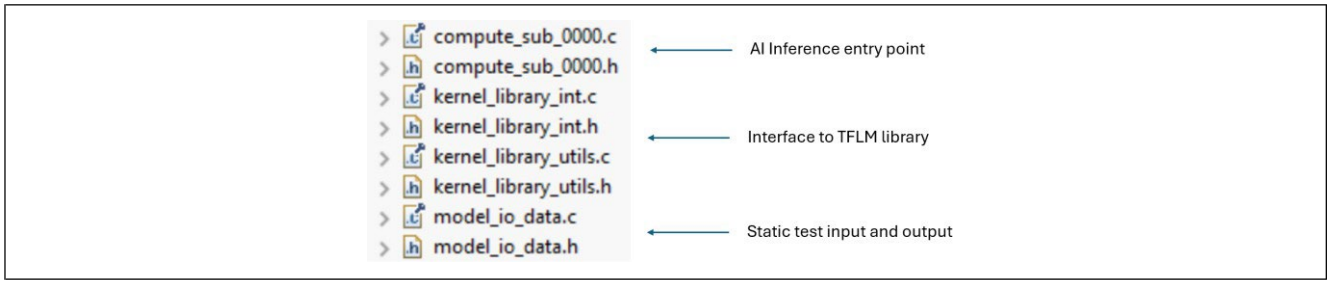


図8: CPU を用いた MobileNet V1 (0.25) の推論用コンパイル結果

これらの出力ファイルをリアルタイム AI 推論の実例と連携させる方法については、「Building a Vision AI Application using the RA8P1 MCU with Ethos-U55 NPU」を参照してください。

RUHMI のコンパイル結果には、静的なテスト入力および出力データが含まれており、これらを用いてコンパイル結果を迅速に評価するためのテストプロジェクトを構築することができます。テスト入力から生成された出力と、コンパイル結果に含まれる出力を比較することで、推論性能を評価する手法として利用できます。

また、コンパイル段階の最後には、使用したモデルの **MACs/Inference (MACs/Batch)** が出力されます。この値は、本モデルを使用した場合の消費電力を相対的に評価する指標の一つとなります。

4. Ethos-U NPU向けFSPスタック

Armはオープンソースの Ethos-U NPUドライバおよびコンパイラを提供しています。Renesas FSP は、Arm オープンソースの **TensorFlow Lite Micro（TFLM）ライブラリ** と連携するために、このドライバを統合しています。

4.1 概説

Ethos-U NPU または CPU を用いて AI 推論を実行する場合は、以下のスタックを使用します。このスタックは、
New Stack → AI → Google TFLM Core Lib
から選択できます。

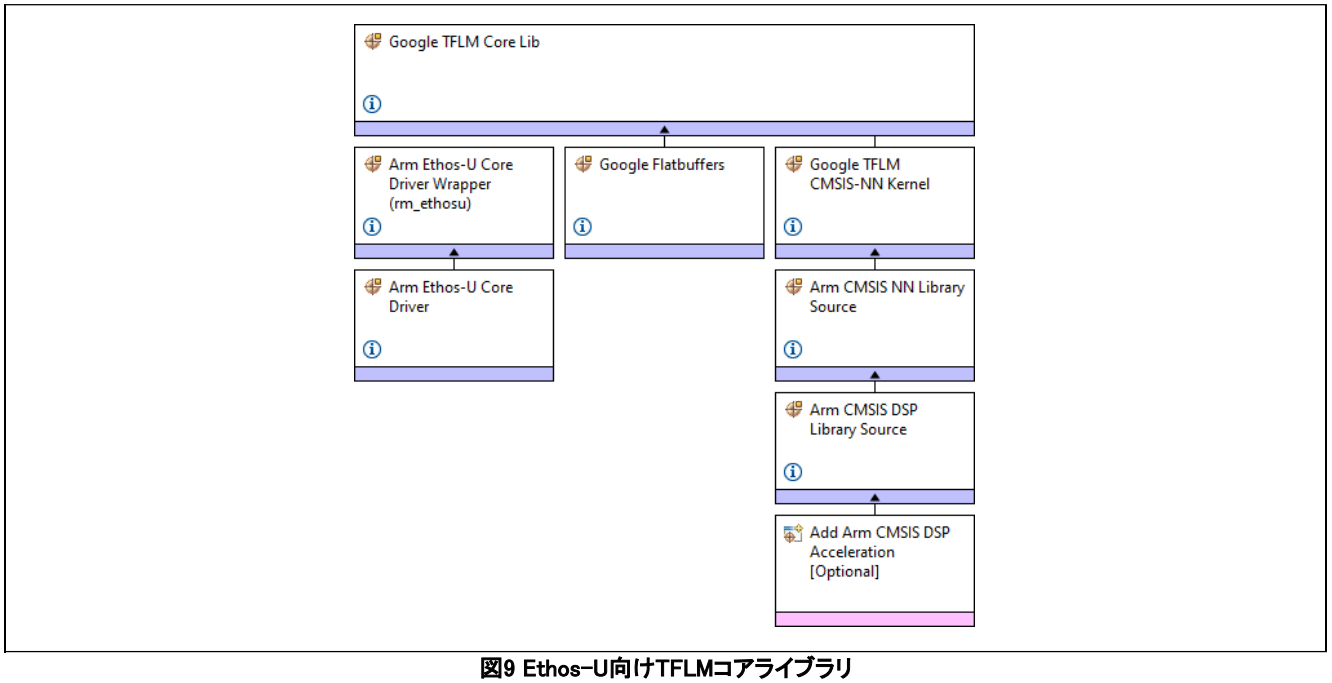


図9 Ethos-U向けTFLMコアライブラリ

図9に示した FSP スタックを展開すると、以下のソフトウェアコンポーネントがユーザプリケーションに生成されます。AI アプリケーションで使用される主要なソフトウェアコンポーネントの詳細については、該当するセクションを参照してください。

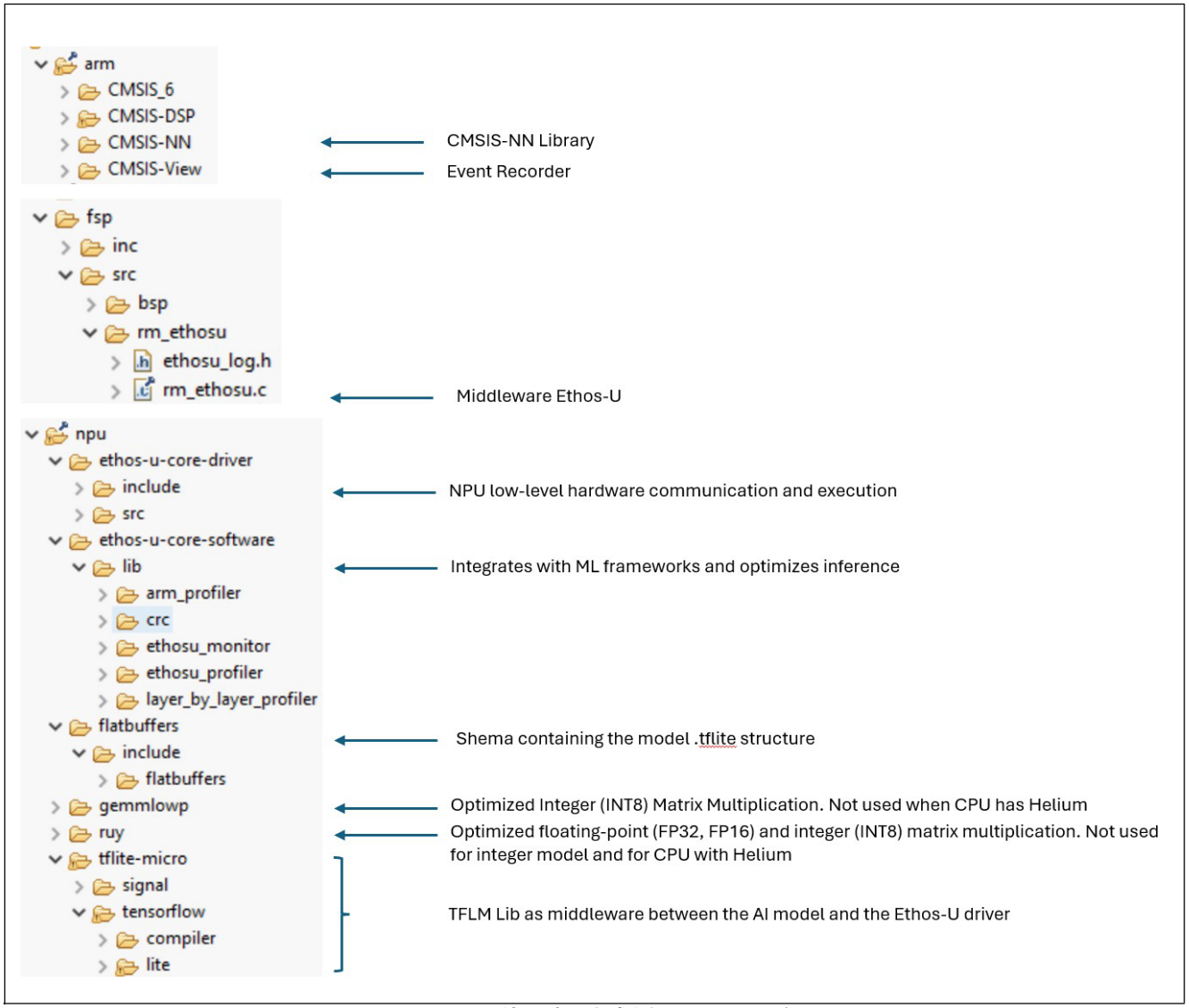


図10: NPU 使用時に生成される FSP コード

4.2 FSPドライバスタック

4.2.1 Google TFLMコアライブラリ

TFLM アプリケーションコードでは、モデル、オペレーターリゾルバ、テンソルアリーナ、およびそのサイズを指定するためにインタプリタを定義し、これらのパラメータを TFLM に渡します。

- TFLM インタプリタは、事前にコンパイルされたモデル配列へのポインタを使用して初期化されます。これによりファイルI/Oが不要となり、メモリ使用量を削減できます。
- TensorFlow Lite には Ethos-U デリゲートが用意されています。推論時に TFLM は Ethos-U のカスタムオペレーターを認識し、サポートされている演算（畳み込み、活性化、プーリング、全結合層など）を Ethos-U NPU にオフロードします。Ethos-U でサポートされていない演算は CPU 上で実行されます。
- CPU推論に関する注意事項：
    - Ethos-U Core Driver Wrapper および Core Driver は必須ではありませんが、TFLM コアライブラリでは Ethos-U スタックが提供する一部の標準デバッグログ機能を使用しているため、本スタックを含めることを推奨します。
    - Renesas RUHMI または MERA ツールでコンパイルされた AI モデルの出力を組み込みシステムで使用する場合、Ethos-U 推論および CPU 推論のいずれにおいても FlatBuffer は使用されません。ただし、統合されている上流の TFLM ライブラリが FlatBuffer ユーティリティに依存しているため、ビルドを正常に完了させるには FlatBuffer スタックを含める必要があります。

- TFLM コアライブラリは、オープンソースの Vela ツールを用いた推論にも対応しています。この場合、メモリ上に配置された .tflite 形式の NN モデルデータを読み込み、ニューラルネットワーク演算子の処理を実行します。

4.2.2 Arm Ethos-U Core Driver Wrapper

Arm Ethos-U Core Driver Wrapper（rm_ethosu）スタックでは、以下の設定が必要となります。本スタックの多くのプロパティは直感的に理解できる内容となっています。コールバックを設定することで、推論処理が完了したことをユーザに通知することができます。

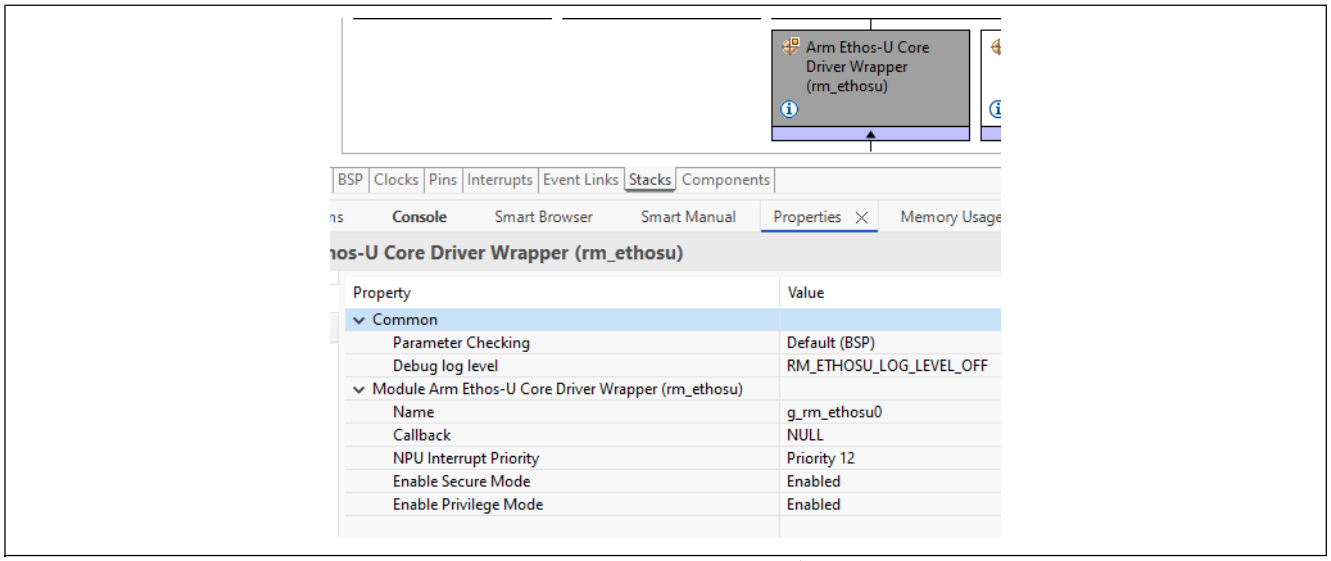


図11: FSP Ethos-U Core Driver ミドルウェア

Ethos-U ミドルウェアは Renesas により開発されたもので、アプリケーションから利用可能な高レベル API を提供します。以下は、アプリケーションから呼び出し可能な公開 API の一覧です。

- **RM_ETHOSU_Open**
  NPU を起動する関数です。Ethos-U NPU コアドライバに含まれる ethosu_init API を呼び出して NPU を初期化し、NPU 割り込みを有効化するとともに、コールバックを設定し、NPU を初期化済み状態として登録します。
  ethosu_init API は、NPU に対してソフトリセットを実行した後、CPU との通信のために AXI を再初期化します。
- **RM_ETHOSU_CallbackSet**
  推論処理が完了した際に呼び出されるコールバックを設定します。
- **RM_ETHOSU_Close**
  NPU 割り込みを無効化し、Ethos-U NPU コアドライバの ethosu_deinit API を呼び出してリソースを解放し、NPU を停止します。

デバッグログレベルプロパティの詳細については、**5.5.2 節**を参照してください。

4.2.3 Arm Ethos-U Core Driver

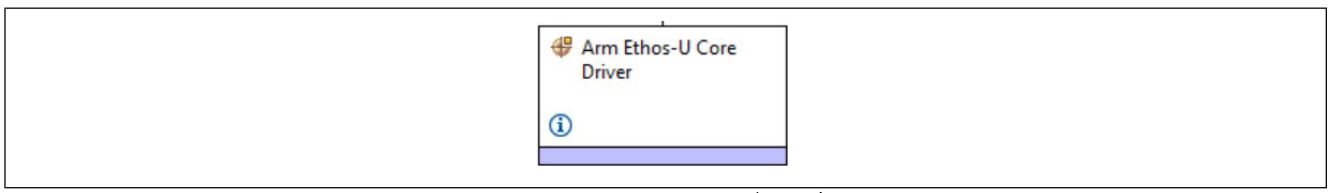


図12. Arm Ethos-Uコアドライバ

Ethos-U Core Driver スタックは、Ethos-U NPU ハードウェアと TFLM ランタイムをつなぐ低レベルのソフトウェア層です。本スタックには、Arm が提供するオープンソースの Ethos-U ドライバが組み込まれています。なお、このドライバは Arm により GitHub 上で公開されています。

Ethos-U Core Driver スタックには、ユーザが設定する項目はありません。

このスタックの主な機能は次のとおりです：

- Ethos-U NPU への推論演算開始の設定
- リソース制約のあるデバイスにおける、モデル実行用メモリバッファの管理
- AI モデルコンパイラによって挿入される Ethos-U カスタムオペレーターの処理
- MCU（Cortex-M）と Ethos-U NPU 間の安全かつ効率的な通信の確保
- PMU 用のコアソフトウェアドライバの提供（サイクルカウント、イベント監視、性能プロファイリング、リソース使用率解析）

4.2.4 Google TFLM CMSIS-NNカーネル

モデルが Ethos-U NPU と CPU の両方で部分的に実行される場合、または CPU のみで実行される場合には、CMSIS-NN ライブラリが性能最適化のために使用されます。このライブラリは、Cortex-M MCU に搭載されている Arm Helium テクノロジーを活用するよう設計されており、CPU 上でのニューラルネットワーク層の効率的な実行を可能にします。

CMSIS-NN を使用することで、Cortex-M85 上でサポートされている演算を実行する際の推論速度を大幅に向上させ、CPU 負荷を低減することができます。

5. 実運用を想定した AI アプリケーションの開発

5.1 AI 推論処理の概要

これまでの説明を踏まえ、AI モデル推論の基本的な処理手順を以下に示します。

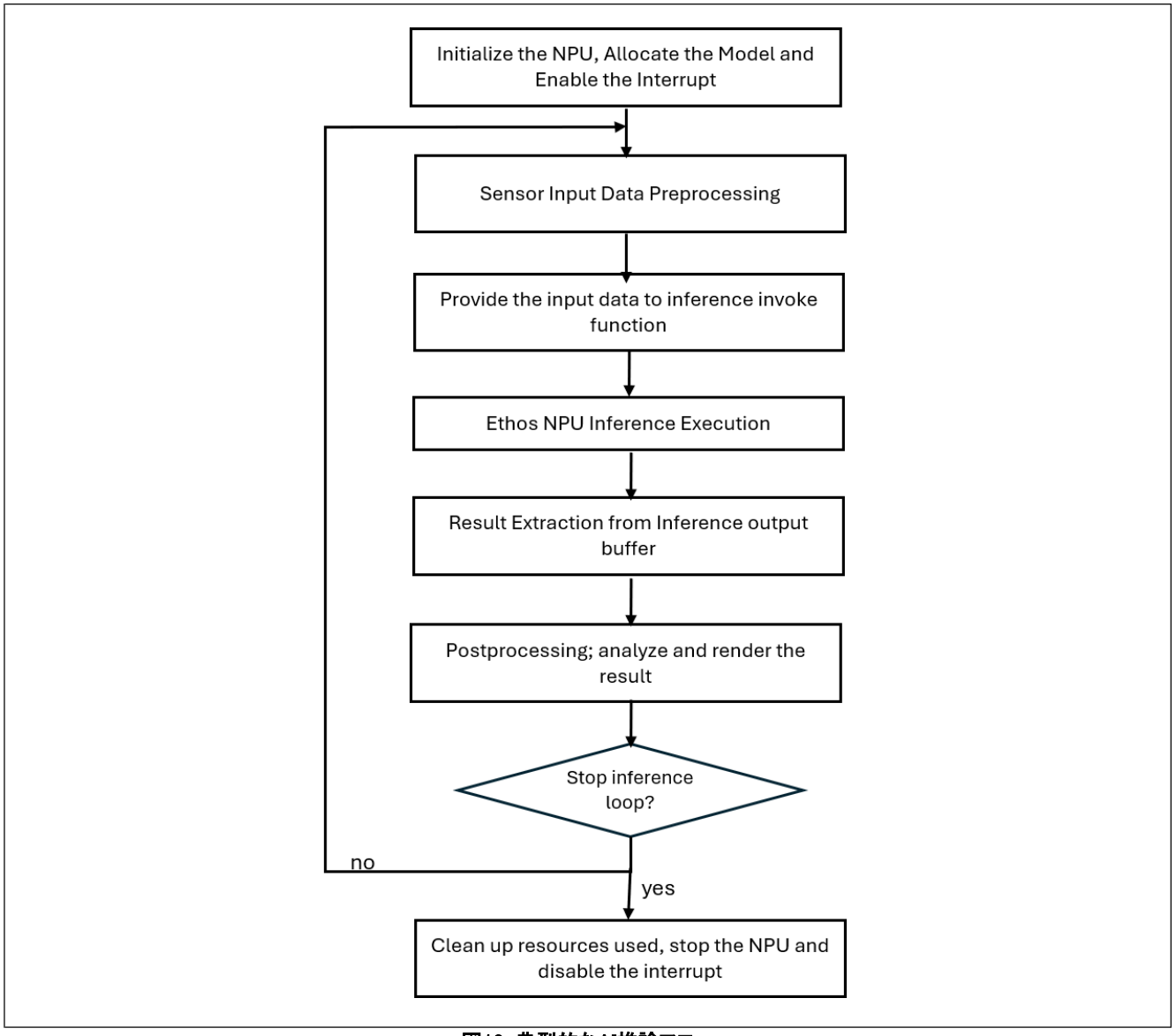


図13. 典型的なAI推論フロー

5.1.1 高い推論精度を実現するために

組み込みアプリケーションにおいて高い推論精度を得るためには、正確で安定した入力データを提供することが重要です。

- Ethos-U NPU では、入力テンソル、出力テンソル、および中間演算結果（テンソルアリーナ 領域）用に専用のメモリ領域が必要です。CPU のスタックやヒープと分離することで、メモリ競合を防ぐことができます。
- Ethos-U を用いた推論中は、入力テンソルのデータが変更されないようにする必要があります。そのため、センサから取得した実行時データは、推論中に更新されない専用バッファへコピーすることを推奨します。
- NPU のメモリアクセスを最適化するため、バッファは 32 バイト境界にアラインしてください。
- 利用可能な場合は、頻繁にアクセスされる小容量データや割り込み処理に TCM（Tightly Coupled Memory）を使用することで、レイテンシを低減できます。

5.1.2 高速な推論を実現するために

推論速度に最も大きく影響する要因は、入力テンソル、テンソルアリーナ、および出力テンソルが配置されているメモリのアクセス速度です。
AI の後処理は非同期で実行される場合が多いため、通常は出力テンソルの配置場所が推論速度に与える影響は限定的です。

5.2 メモリアーキテクチャに関する考慮事項

NPU を搭載した RA8 マイコンでは、コード格納用として MRAM が内蔵されており、同一プロセスのフラッシュメモリと比較して高速かつ低消費電力で動作します。
さらに、評価キットでは、大容量の SRAM、外付け高速 SDRAM、OSPI デバイスが提供されており、大規模 AI モデル、実行時の大容量センサデータ、および静的データの保存に対応しています。

以下に示す各メモリ領域の使用に関する指針は、**EK-RA8P1** を前提としています。

メモリの種類			容量	AI アプリケーションでの主な用途
MRAM			1MB	AI モデル格納、アプリケーションコード
SRAM	ユーザSRAM		1664 KB	AI モデル格納、センサデータ保存、テンソルアリーナ、推論結果、アプリケーションコードのスタック／ヒープ
	Cortex-M85 TCM	ITCM	128 KB	割り込み処理、その他の時間クリティカルな処理、センサデータ
		DTCM	128 KB	センサデータ保存、時間決定性が求められる頻繁にアクセスされるデータ
	Cortex-M33 TCM	ITCM	64 KB	割り込み処理、その他の時間クリティカルな処理、センサデータ
		DTCM	64 KB	センサデータ保存、時間決定性が求められる頻繁にアクセスされるデータ
データキャッシュ			16 KB（ECC 付き）	Tensor 演算の高速化、消費電力低減、機械学習処理の高速化のため有効化を推奨
命令キャッシュ			16 KB（ECC 付き）	命令フェッチの高速化、レイテンシ低減、効率的な分岐予測のため有効化を推奨
外付けSDRAM			64 MB（16M × 32bit）	AI モデル格納、実行時センサデータ保存、グラフィックス用フレームバッファ、テンソルアリーナ、アプリケーションコードのスタック／ヒープ
SiPフラッシュメモリ			8MB	AI モデル格納、グラフィックスデータ保存、実行頻度の低いコード
外付け OSPI メモリ			64 MB	AI モデル格納、静止画像データ保存、暗号化された機密情報、時間非依存なアプリケーションコード
NPU内部メモリ領域			48 KB	DMA バッファ（コマンドストリーム、テンソルデータ[入力・中間・出力]、ニューラルネットワーク重み、スクラッチバッファ）※アプリケーションコードからは使用不可

5.2.1 メモリ分割および領域割り当て

Ethos-U NPU を使用した AI アプリケーションのメモリ割り当てでは、以下の点を考慮する必要があります。

- システム性能：同じモデルであっても、高速なメモリを使用することで推論性能が向上します
- AI アプリケーションと同一システム上で動作する他のアプリケーション（例：グラフィックスアプリケーション）によるメモリ使用状況
- 対象とする AI アプリケーションの特性：モデルサイズやセンサデータ量の違いが、メモリ管理方針に大きく影響します

IDE では、アプリケーションコードに基づいたデフォルトのメモリ構成が自動的に設定されます。プロジェクトのメモリ構成は、
¥Debug¥memory_regions.lld および ¥Debug¥fsp_gen.lld
で定義されており、これらのリンカファイルは共通のリンカスクリプト fsp.lld から参照されます。

AI アプリケーションでは、たとえば以下のようにメモリ構成をカスタマイズすることが有効な場合があります。

- デュアルコア構成において、2 つのコア間で共有するメモリ領域を定義する。
- 複数の AI 推論インスタンスを使用する場合、それぞれに専用のメモリ領域を割り当てる。

デュアルコアおよびシングルコアの RA8 マイコンにおけるメモリ領域の割り当ては、Smart Configurator を使用して設定できます。

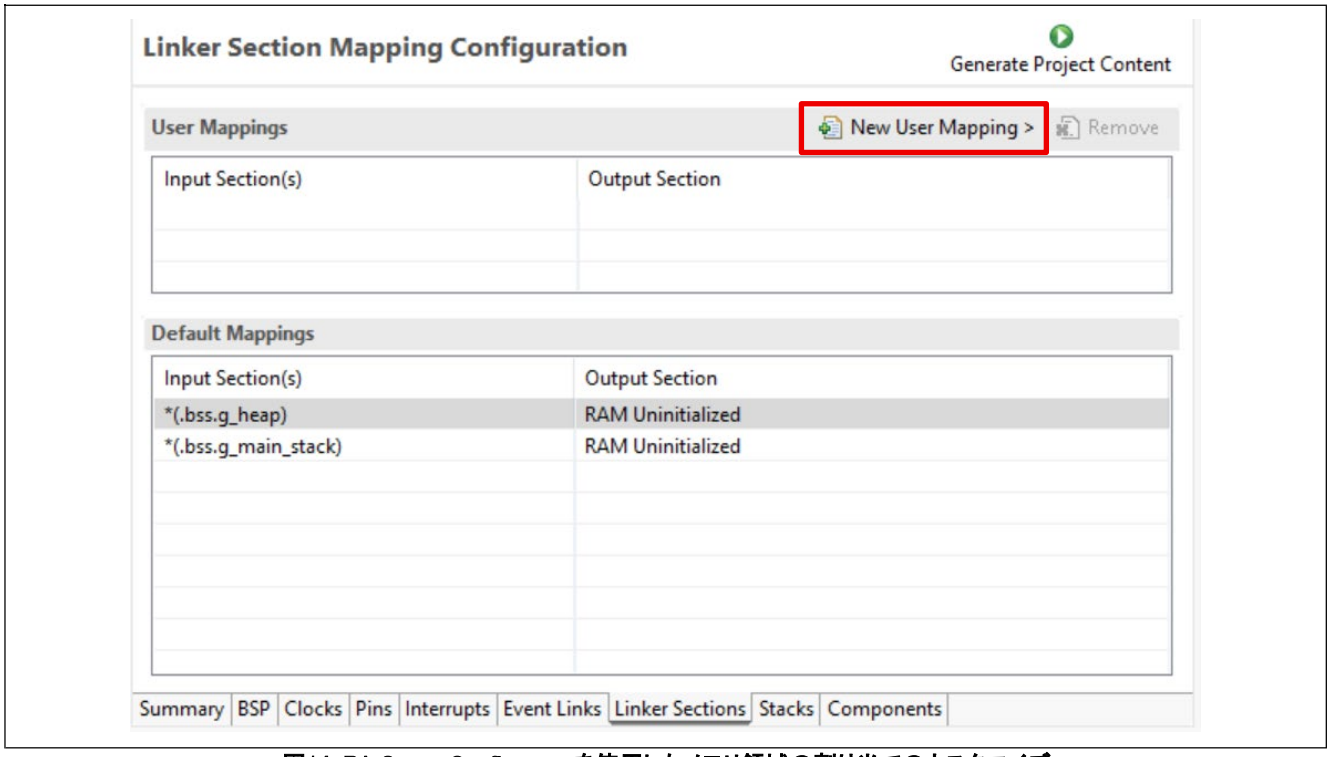


図14. RA Smart Configuratorを使用したメモリ領域の割り当てのカスタマイズ

5.2.2 コードフラッシュ、SRAM、外付け SDRAM の使い分け

コードフラッシュ（MRAM）、SRAM、および外付け SDRAM にデータを割り当てる際の一般的な考慮事項を以下に示します。

- SRAM は SDRAM よりも高速です。可能な限り、モデル入力となるセンサデータおよび テンソルアリーナ は SRAM に配置してください。
  テンソルアリーナ が SRAM に収まらない場合は、SDRAM を テンソルアリーナ 用メモリとして使用できます。これは、テンソルアリーナ を .sdram_noinit 領域に割り当てることで実現できます。
- デフォルトでは、AI モデルは不揮発性メモリ領域に格納されます。モデルや中間活性化データが内部コードフラッシュ容量を超える場合は、外付け DRAM（例：SDRAM）の使用が必要となることがありますが、その場合はメモリアクセスレイテンシが増加します。
  このような場合、モデルを外付けフラッシュ（例：OSPI）に格納し、MCU リセット時に SDRAM へコピーする方法が有効です。これは、モデルを .sdram_from_ospi0_cs1 領域に割り当てることで実現できます。
- モデルを SRAM 上で実行したい場合は、モデルを OSPI などの外付けフラッシュに格納し、MCU リセット時に SRAM へコピーすることも可能です。この場合、モデルを .ram_from_ospi0_cs1 領域に割り当てます。
- 以下のようなメモリ階層構成を採用することを推奨します。
  - 重みおよびバイアスパラメータ：Flash、MRAM、または OSPI
  - 入力／出力データおよび テンソルアリーナ：高速アクセス可能な SRAM

5.2.3 テンソルアリーナ の割り当て

テンソルアリーナ は通常、推論処理用のバッファとして SRAM または SDRAM に静的に割り当てられます。RUHMI または MERA を使用してモデルをコンパイルした場合、必要な テンソルアリーナ サイズはコンパイル時に生成される出力ファイルに明記されます。

以下は、MobileNet V1（0.25）整数モデルのコンパイル結果から抜粋した テンソルアリーナ メモリ定義の例です。この例では、テンソルアリーナ は SRAM に割り当てられています。

図15 に示すような定義は、RUHMI または MERA のコンパイル出力に含まれる sub_0000_invoke.c ファイル内でも確認できます。

```
// Define arenas with allocation and 16-byte alignment
__attribute__((aligned(16))) uint8_t sub_0000_arena[401408];
```

図15: Ethos-U 推論に必要な AI モデルの テンソルアリーナ サイズ

以下に、CPU 推論時における テンソルアリーナ サイズ定義の例を示します。
アプリケーションコードでは、この テンソルアリーナ 領域を SRAM または SDRAM 上に確保し、そのバッファを推論処理を呼び出す compute_sub API に渡す必要があります。

図16 に示すような定義は、RUHMI または MERA によるコンパイル結果に含まれる compute_sub_0000.h ファイル内でも確認できます。

```
/*
  kBufferSize_sub_0000 is a compile-time constant to be used by the user of compute_sub_0000 function
  to allocate a buffer with at least the specified size.

  Example of how to call the compute function:

    // it is possible to use either heap, stack or a custom data section to allocate this buffer
    uint8_t my_buffer[kBufferSize_sub_0000];

    int main() {
      ...
      compute_sub_0000(my_buffer, input, output);
    }
*/
enum BufferSize_sub_0000 {
  kBufferSize_sub_0000 = 605712
};
```

図16: CPU 推論に必要な AI モデルの テンソルアリーナ サイズ

5.2.4 DMAおよびキャッシュ管理

DMA およびキャッシュ管理に関する主な注意点を以下に示します。

- Ethos-U はメモリ転送に DMA を使用するため、バッファはキャッシュコヒーレントなメモリ領域に配置してください
- キャッシュ非対応メモリを使用する場合は、メモリバッファをキャッシュライン境界にアラインしてください
- 古いデータの参照を防ぐため、テンソル処理の前後でキャッシュのフラッシュおよび無効化を行ってください

AI 推論の前後でのデータキャッシュ処理の具体例については、Building a Vision AI Application using the RA8P1 MCU with Ethos-U55 NPU (R11AN0995) を参照してください。

5.3 消費電力と性能のトレードオフ

多くのアプリケーションにおいて、消費電力の低減と性能向上は相反する要件となります。システム全体の性能および電力効率を最大化するため、以下の点を考慮したアプリケーション設計が重要です。

- メモリアクセスパターンを最適化し、消費電力を低減する
- 外付けメモリの使用を最小限に抑え、エネルギーオーバーヘッドを削減する

5.4 パフォーマンス分析

Renesas FSP パッケージには、AI アプリケーションの性能解析に使用できる以下のソフトウェアコンポーネントが用意されています。

5.4.1 CMSIS-NN Event Recorder の使用

CMSIS-NN Event Recorder は、Arm の CMSIS（Cortex Microcontroller Software Interface Standard）に含まれる機能で、Cortex-M プロセッサ上で動作するニューラルネットワーク（NN）アプリケーションの性能を監視・解析できます。
CMSIS-NN を使用したプロジェクトに Event Recorder を組み込むことで、実行フロー、処理時間、およびボトルネックを可視化でき、効率的なデバッグおよび最適化が可能となります。

主な特長:

- **イベントアノテーション**:特定の関数や処理を監視するためのイベントマーカーをコードに挿入可能
- **タイミング解析**:イベントごとのタイムスタンプにより実行時間を測定可能(詳細情報は Arm Developer を参照)
- **リアルタイム監視**:プログラム実行中のイベントをリアルタイムで確認可能
- **低侵襲設計**:システム性能への影響を最小限に抑えた高精度なプロファイリング

Renesasでは、NPU スタックをプロジェクトに追加した場合、Event Recorder を Arm CMSIS-View ライブラリの一部としてFS パッケージに統合しています。

5.4.2 Ethos-Uパフォーマンスカウンタの使用

Arm Ethos-U Performance Monitoring Unit(PMU)は、Ethos-U55 や Ethos-U65 などの Ethos-U シリーズ NPU に搭載された性能監視機能です。ニューラルネットワーク処理の性能を詳細に解析し、リソース使用状況の最適化に役立てることができます。

Ethos-U PMU の主な機能:

- **サイクルカウンタ**:NPU が処理を実行するのに要したサイクル数を計測する 48 ビットカウンタ
- **イベントカウンタ**:メモリアクセスや命令実行などの特定イベントを監視可能な 32 ビットカウンタ(4 本)

PMU を活用することで、Ethos-U NPU 上でのモデル実行性能を把握し、ボトルネックの特定や性能改善に役立てることができます。ユーザプリケーションからは、推論処理に要した時間の計測などに PMU を使用できます。

使用可能な API は、NPU Core Driver の¥ra¥npu¥ethos_u_core_driver¥inc¥ethos_driver.hに定義されています。

利用方法の詳細については、FSP ユーザズマニュアルEthos-U (rm_ethosu)章の「Getting Started: Defining functions and PMU events for MCU performance profiling」を参照してください。

5.4.3 GPTタイマの使用

アプリケーションレベルでの性能測定には、専用タイマを使用する方法もあります。
RA8P1 MCUファミリには 32ビット汎用PWM タイマ(GPT)が搭載されており、システム性能の測定に利用できます。実装例については、Building a Vision AI Application using the RA8P1 MCU with Ethos-U55 NPU (R11AN0995) を参照してください。

5.5 デバッグサポート

TFLM ライブラリおよび NPU コアソフトウェアスタックでは、アプリケーションで選択したインタフェースに出力可能な、以下 2 種類のデバッグログ機構が提供されています。

5.5.1 TFLMデバッグログコールバックの使用

TFLMライブラリには、デバッグログ用のフック関数 RegisterDebugLogCallback が用意されています。この関数は
¥ra¥npu¥tflite-micro¥tensorflow¥lite¥micro¥cortex_m_generic_debug_log.cc
に定義されています。

ユーザは、デバッグ情報を出力・記録するためのコールバック関数を実装し、その関数ポインタを登録することで、ログ出力を制御できます。たとえば、SEGGER RTT や J-Link Consoleへログを出力する実装が可能です。実装例については Building a Vision AI Application using the RA8P1 MCU with Ethos-U55 NPU (R11AN0995) を参照してください。

5.5.2 Ethos-U NPUデバッグログの使用

Ethos-U NPU のデバッグログは、FSPスタック設定から有効化およびログレベルの指定が可能です。複数のデバッグログレベルが用意されており、¥ra¥fsp¥src¥rm_ethosu¥ethos_log.hで定義されているマクロ関数内で使用されます。
ethos_log.h に定義された printf ベースの出力は、将来的に SEGGER RTT や J-Link Console などへルーティング可能です。

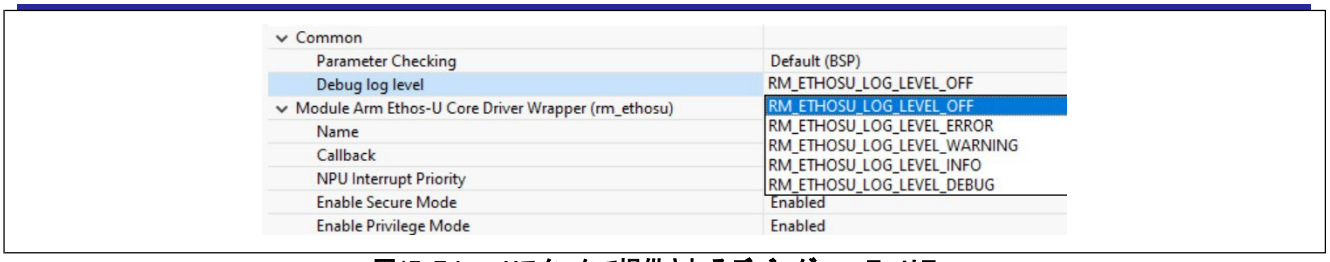


図17. Ethos-Uスタックで提供されるデバッグユーティリティ

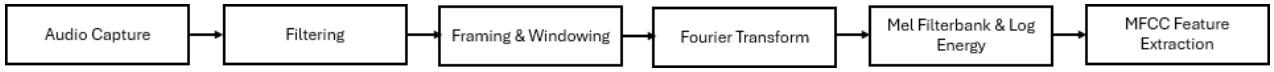
5.6 ユースケース分析

モデルのコンパイルと組み込みシステムへのデプロイは、静的な入力および出力機能を使用して比較的簡単に評価できますが、エンドツーエンドの開発には通常、センサデータの収集、前処理、および後処理のステップでの大規模な開発作業が含まれます。

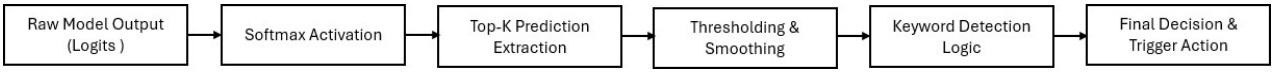
5.6.1 キーワードスポッティング

キーワードスポッティングAIアプリケーションの典型的な前処理と後処理をここに参考として提供します。正確な操作は、使用されるモデルとセンサに依存します。

キーワードスポッティングの前処理



キーワードスポッティングの後処理

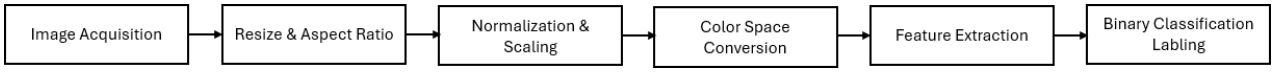


キーワードスポッティングモデルとテンソルアリーナ領域は通常1MBを大幅に下回ります。ユーザは、RA8P1の大容量SRAMとTCMを利用して、システムパフォーマンスを最適化し、パフォーマンスを向上させることができます。

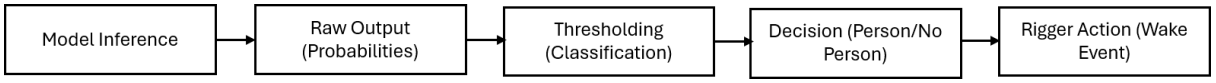
5.6.2 ビジュアルウェイクワード

ビジュアルウェイクワードAIアプリケーションの典型的な前処理と後処理をここに参考として提供します。正確な操作は、使用されるモデルとセンサに依存します。

ビジュアルウェイクワードの前処理



ビジュアルウェイクワードの後処理

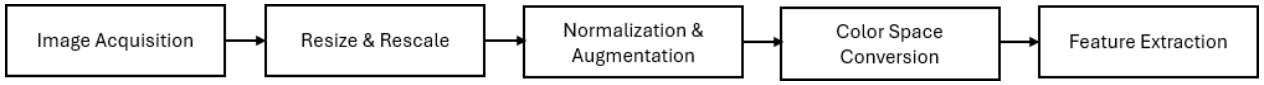


典型的なビジュアルウェイクワードモデルとテンソルアリーナ領域はそれぞれ1MBを大幅に下回ります。ユーザは、RA8P1の大容量SRAMとTCMを利用して、システムパフォーマンスを最適化し、パフォーマンスを向上させることができます。

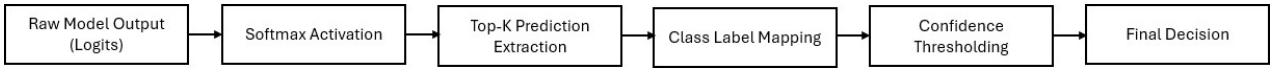
5.6.3 画像分類

画像分類は、オブジェクトの位置を指摘せずにビュー内に存在するオブジェクトを検出します。画像分類AIアプリケーションの典型的な前処理と後処理をここに参考として提供します。正確な操作は、使用されるモデルとセンサに依存します。

画像分類の前処理



画像分類の後処理



画像分類モデルサイズとテンソルアリーナ領域サイズは、数百KBから数MBまで変動します。大規模なモデルの場合、モデルは外部フラッシュ領域に保存され、実行時に使用するためにSDRAMにコピーされ、推論パフォーマンスが向上します。画像分類に関するアプリケーションプロジェクトの例については、R11AN0995を参照してください。

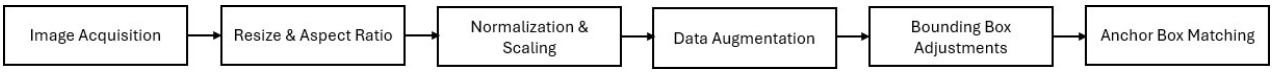
5.6.4 オブジェクト検出

オブジェクト検出は、どのオブジェクトが存在し、どこに位置しているかという質問に答えます。オブジェクト検出AIアプリケーションの典型的な前処理と後処理をここに参考として提供します。正確な操作は、使用されるモデルとセンサに依存します。

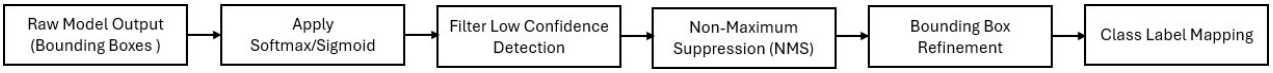
オブジェクト検出のモデルサイズは、検出可能なオブジェクトカテゴリの数に密接に関連していますが、モデルアーキテクチャにも依存します。

オブジェクト検出の特別なタイプは顔検出です。RA8D1を使用したオブジェクト検出の例については、アプリケーションノート「Reference System Design for Vision AI on EK-RA8D1」を参照してください。このアプリケーションプロジェクトで使用される顔検出モデルは、多くのオブジェクトカテゴリを検出するオブジェクト検出モデルよりも比較的小さいですが、以下の前処理と後処理のステップを示しています。

オブジェクト検出の前処理



オブジェクト検出の後処理：



オブジェクト検出モデルサイズとテンソルアリーナ領域サイズは、数百KBから数MBまで変動します。大規模なモデルの場合、モデルは外部フラッシュ領域に保存され、実行時に使用するためにSDRAMにコピーされ、推論パフォーマンスが向上します。

6. Appendix

6.1 オープンソースソリューションを使用したモデル量子化

モデル量子化は、オープンソースのツールおよびルネサスのRUHMIツールの両方でサポートされています。

オープンソース環境でのモデル量子化については、TensorFlow Liteの公式サイトにあるモデル最適化ページを参照し、量子化処理の内容をご確認ください。

RUHMIを用いたモデル量子化については、RUHMI Platform Quick Start Guide を参照してください。

6.2 Arm Velaを使用したモデルのコンパイル

量子化されたニューラルネットワークは、Ethos-U NPUオープンソースコンパイラを使用してオフラインでコンパイルされ、コマンドストリームを生成できます。Armは、このコンパイルを実行するためにオープンソースコンパイラVelaを提供しています。Velaコンパイラの詳細な使い方については、Arm公式サイト「Ethos-U Vela compiler」を参照してください。

6.2.1 オープンソースツールの使用

以下は、このツールを使用して整数量子化されたTFLiteモデルをc配列に変換し、Ethos-U NPUによるAI推論に使用するための概要です。

.iniファイルは、MCUおよびNPUハードウェアを構成するために使用されます。.iniファイルは、NPUコアクロックと、テンソルアリーナがNPU内部バス帯域幅に位置する間のメモリのメモリ帯域幅の比率を定義します。

生成された.hファイルには、モデルやモデルのサイズなどのメタデータが含まれています。Velaコンパイル前後のモデルファイル形式は同じですが、カスタムEthos-Uレイヤを使用するように変換されたレイヤは変更されることに注意してください。

このツールを使用すると、整数量子化されたTFLiteモデルをEthos-U NPUによるAI推論で使用するためのC配列形式へ変換できます。

- **.ini ファイル**

  MCUおよびNPUのハードウェア構成を設定するために使用します。このファイルでは、以下のような項目を定義します。

  - NPUのコアクロック
  - テンソルアリーナが配置されるメモリの帯域と、NPU内部バス帯域との比率

- **生成される .h ファイル**

  モデル本体に加えて、モデルサイズなどのメタデータが含まれます。

なお、Velaコンパイルの前後でモデルファイルの形式自体は同じですが、Ethos-U専用レイヤを使用するように変換されたレイヤについては、内部構成が変更されている点に注意してください。

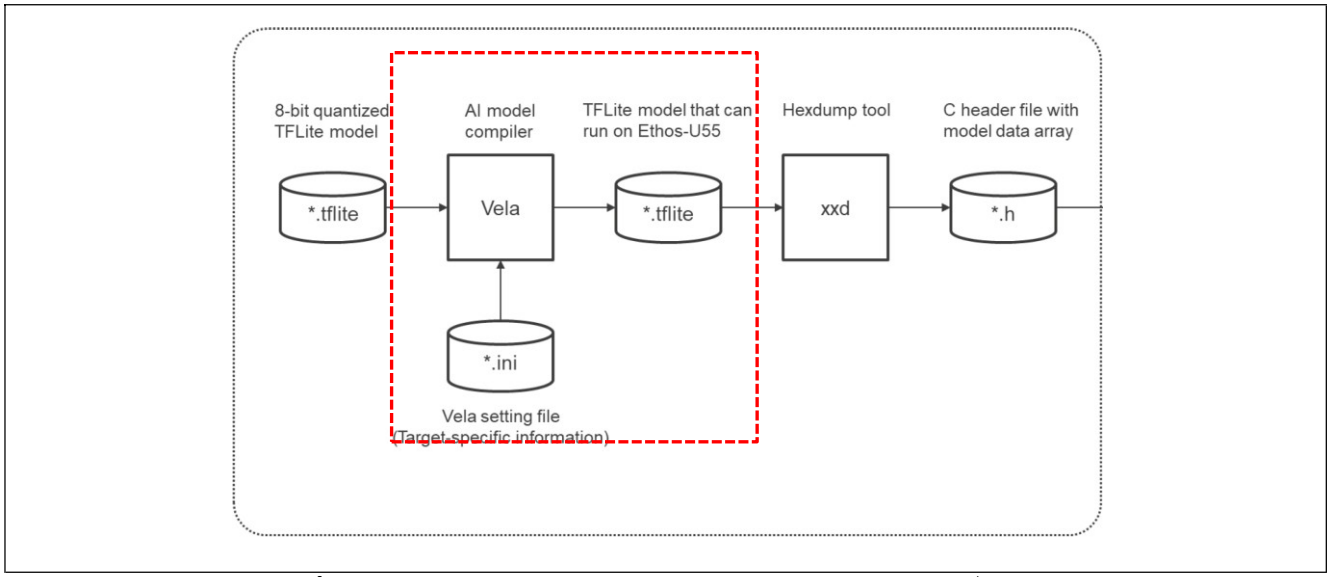


図18. オープンソースソリューションを使用したEthos-U NPU使用のためのモデルのコンパイル

CPUのみを使用してモデル推論を評価するには、ソフトウェアツールとVelaコンパイラをインストールする必要はありません。赤い点線のボックスに含まれるステップはスキップする必要があります。

7. 参考資料

1. RA Flexible Software Package (FSP) | Renesas
2. RA8P1 Group User's Manual: Hardware
3. Building a Vision AI Application using the RA8P1 MCU with Ethos-U55 NPU (R11AN0995)
4. Renesas RUHMI Framework Quick Start Guide(R11QS0065)
5. Renesas RUHMI Framework GitHub
6. Arm Ethos-U55 NPU Technical Reference Manual

8. ウェブサイトとサポート

製品についての情報、開発ツールやドキュメントのダウンロード、ならびにサポートの利用については、以下のURLをご参照ください。

- RA8P1 製品情報
- EK-RA8P1 製品情報
- RAファミリ
- Flexible Software Package (FSP)
- Renesas Engineering Community
- テクニカルサポートサービス

改版記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2026.3.6	-	初版（R01AN7712EU0100 の日本語版）

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 静電気対策

   CMOS 製品の取り扱いの際は静電気防止を心がけてください。CMOS 製品は強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレーやマガジンケース、導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。また、CMOS 製品を実装したボードについても同様の扱いをしてください。

2. 電源投入時の処置

   電源投入時は、製品の状態は不定です。電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。

3. 電源オフ時における入力信号

   当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」についての記載のある製品は、その内容を守ってください。

4. 未使用端子の処理

   未使用端子は、「未使用端子の処理」に従って処理してください。CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。

5. クロックについて

   リセット時は、クロックが安定した後、リセットを解除してください。プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。

6. 入力端子の印加波形

   入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、V_{IL}（Max.）から V_{IH}（Min.）までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、V_{IL}（Max.）から V_{IH}（Min.）までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。

7. リザーブアドレス（予約領域）のアクセス禁止

   リザーブアドレス（予約領域）のアクセスを禁止します。アドレス領域には、将来の拡張機能用に割り付けられている リザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。

8. 製品間の相違について

   型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。同じグループのマイコンでも型名が違うと、フラッシュメモリ、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ幅射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。回路、ソフトウェアおよびこれらに関連する情報を使用する場合、お客様の責任において、お客様の機器・システムを設計ください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品または本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を組み込んだ製品の輸出入、製造、販売、利用、配布その他の行為を行うにあたり、第三者保有の技術の利用に関するライセンスが必要となる場合、当該ライセンス取得の判断および取得はお客様の責任において行ってください。
5. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
6. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。
   標準水準：　コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット等
   高品質水準：輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等
   当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
7. あらゆる半導体製品は、外部攻撃からの安全性を 100%保証されているわけではありません。当社ハードウェア／ソフトウェア製品にはセキュリティ対策が組み込まれているものもありますが、これによって、当社は、セキュリティ脆弱性または侵害（当社製品または当社製品が使用されているシステムに対する不正アクセス・不正使用を含みますが、これに限りません。）から生じる責任を負うものではありません。当社は、当社製品または当社製品が使用されたあらゆるシステムが、不正な改変、攻撃、ウイルス、干渉、ハッキング、データの破壊または窃盗その他の不正な侵入行為（「脆弱性問題」といいます。）によって影響を受けないことを保証しません。当社は、脆弱性問題に起因しまたはこれに関連して生じた損害について、一切責任を負いません。また、法令において認められる限りにおいて、本資料および当社ハードウェア／ソフトウェア製品について、商品性および特定目的との合致に関する保証ならびに第三者の権利を侵害しないことの保証を含め、明示または黙示のいかなる保証も行いません。
8. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
10. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
12. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
13. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
14. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

注 1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社が直接的、間接的に支配する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

（Rev.5.0-1　2020.10）

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

弊社の製品や技術、ドキュメントの最新情報、最寄の営業お問合せ窓口に関する情報などは、弊社ウェブサイトをご覧ください。

www.renesas.com/contact/

商標について

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。すべての商標および登録商標は、それぞれの所有者に帰属します。