Vitis™ AIではじめるエッジAI

株式会社システム計画研究所／ISP

2023年9月1日
SWEST25@水明館

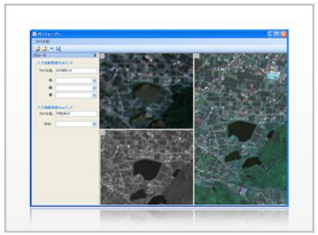
会社紹介

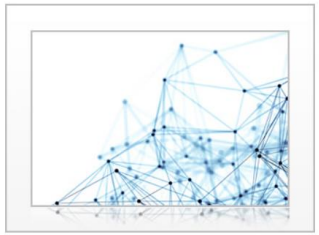
- 株式会社システム計画研究所／ISP
  - 1977年創業の独立研究開発型のソフトウェア開発会社です。


人工知能


医療情報


画像処理

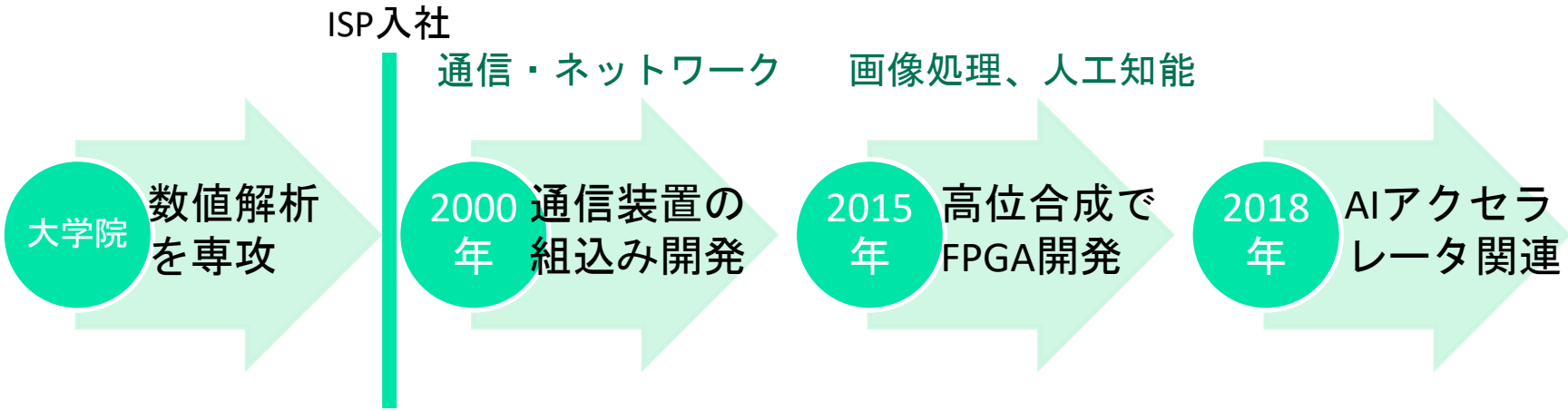

宇宙・制御


通信・ネットワーク

- SWESTとの関わり
  - SWEST18「技術者が知っておきたいDeep Learningの基礎と組込みでの利用」上島仁
  - SWEST19「ソフトウェアエンジニアでもできる、ハードウェアをやわらかく使う方法」満田賢一郎
  - SWEST22「ROS2リアルタイムの最新動向の紹介と、ROS2への期待　－　超小型宇宙探査機が広げる世界と背景にある技術 - 宇宙における組み込みソフトウェア開発事例紹介」長谷川敦史, 清水敏郎

自己紹介

経歴



- 組込みエンジニアとしてキャリアをスタート
- C++による高位合成でFPGA開発（RTLは書けません）
- AIアクセラレータ(HW)の論理設計をお手伝いなど
- SWEST25実行委員（PR／レジストレーション担当）

目次

- エッジAIについて
- エッジAIデバイスとしてFPGA
- Vitis™ AIの紹介
- Vitis™ AIライブラリについて
- ハードウェア：Kria KV260 ビジョンAI スターターキット
- Vitis™ AIライブラリを試してみる
- よりエッジAIらしいアプリケーションへ

エッジAIについて

- エッジAIとは「デバイス上（または物理的に近い場所）でAI処理を行うもの」
  ⇔クラウドAI（ネットワーク越しのクラウドでAI処理を実行）
- 利点
  - 低遅延処理が可能
  - プライバシーリスクの低減（ネットワークを介さない／データが集積しない）
  - 通信コストの削減
- ユースケース
  - スマートカメラ
  - 自動運転システムの一部
- 制約：一般的に学習は行わずAI推論機能のみ搭載

エッジAIのプラットフォーム

- エッジAIアクセラレータとして使われるデバイス
  1. GPU： 例 nvidia Jetson シリーズ
     利点：PC/Cloud/スパコンまでカバーするエコシステム
     欠点：消費電力や発熱の問題で採用できないケースもある

  2. 専用チップ：例 google Edge TPU, intel Movidius Myriad X等
     （Interface 2020年10月号「AIチップ図鑑＆実力大研究」）
     利点：一般的に低消費電力かつ高速な動作
     欠点：AIアーキテクチャの進化に置いて行かれる可能性
     エコシステムが育たず、消えていくチップも……

  3. FPGA：例 AMD, intel
     利点：専用チップほどではないが、一般的に低消費電力
     AIアーキテクチャに合わせて自由に設計可能
     欠点：設計・実装のコストが高い
     性能を上げるための工夫が重要

目次

- エッジAIについて
- エッジAIデバイスとしてFPGA
- Vitis™ AIの紹介
- Vitis™ AIライブラリについて
- ハードウェア：Kria KV260 ビジョンAI スターターキット
- Vitis™ AIライブラリを試してみる
- よりエッジAIらしいアプリケーションへ

エッジAIデバイスとしてFPGA

- FPGAはエッジAIデバイスに向いているのか？

  利点として「低消費電力」とあるが……

  - そもそもFPGAの動作周波数は300～400MHz程度
    →1GHzのGPUと比べて低消費電力になる
  - 消費電力を維持するためには、データをチップ内メモリ上に置いたままで多くの処理をする必要がある
    （外部DRAMとのやり取りは消費電力大）
    →AI処理のパラメータ数はFPGAのチップ内メモリに対して過大なためメモリアクセスの階層化パラメータ圧縮等の工夫が必須
  - 低い動作周波数でも処理性能（レイテンシ／スループット）を向上するためには、処理の並列度やパイプライン化の工夫が必須
    →様々な工夫が必要＝設計・実装コストが高い！

エッジAIデバイスとしてFPGAを利用するのは簡単ではない

FPGAをエッジAIデバイスとするための工夫

2016年頃から研究が活発化

- AIモデルのパラメータ圧縮
  - 単精度浮動小数点(32bit)→bit数を削減した数値表現へ変換
    - 8bit整数、8bit固定小数点数、3値化、2値化など
    - CPU/GPUでも8bit整数やbfloat16(16bit浮動小数点数)を利用

  これにより
    - チップ内メモリにより多くのパラメータを格納可能
    - 8bit整数／固定小数点数の演算器はfloat32に比べシンプル
      - 2値化／3値化まで行うとLook up tableで処理が完結
- AIモデルの最適化
  - Pruning(枝刈り)によりパラメータ数を削減
- 処理回路設計上の工夫
  - 処理の並列化、パイプライン化、メモリの階層化など
  - 研究者やベンチャー企業が鎬を削る

目次

- エッジAIについて
- エッジAIデバイスとしてFPGA
- Vitis™ AIの紹介
- Vitis™ AIライブラリについて
- ハードウェア：Kria KV260 ビジョンAI スターターキット
- Vitis™ AIライブラリを試してみる
- よりエッジAIらしいアプリケーションへ

Vitis™ AIの登場

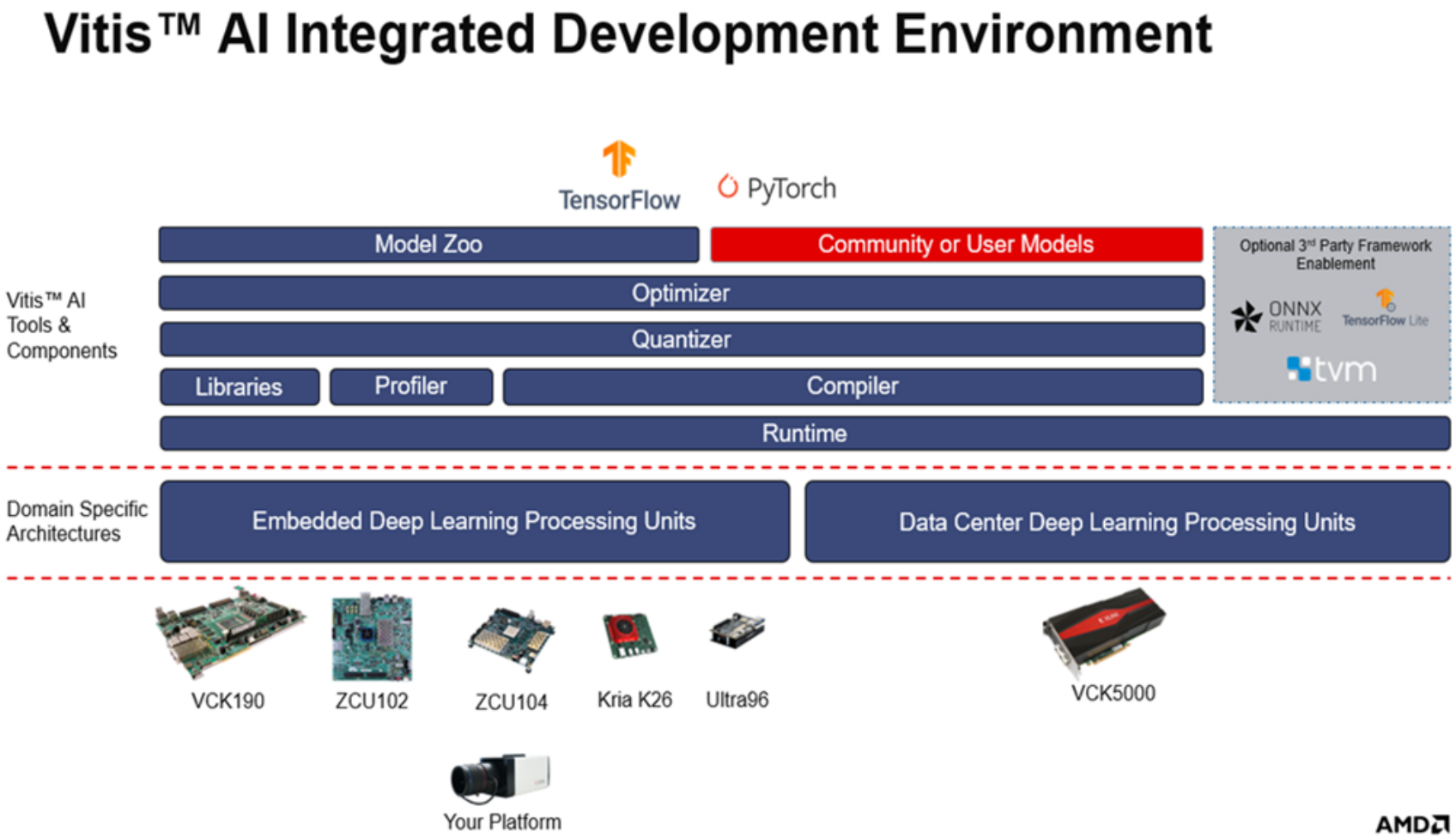
- AMDのデバイス、ボードを使用する包括的な AI 推論開発プラットフォームとしてリリース

  各デバイス向けにDPU(Deep Learning Processor Unit)をリリース

  - エッジデバイス：
    Zynq UltraScale+™ MPSoC、Versal™ Adaptive SoC、
    Kria™ K26 (SOM)
  - クラウドデバイス：Versal® AI コア シリーズ
  - データセンター向けカード：
    Alveo™ data center accelerator card
- 豊富な AI モデル、ツール、ライブラリ、サンプル デザインを提供

FPGAで必要な工夫が織り込み済み！

Vitis™ AI開発環境の概要



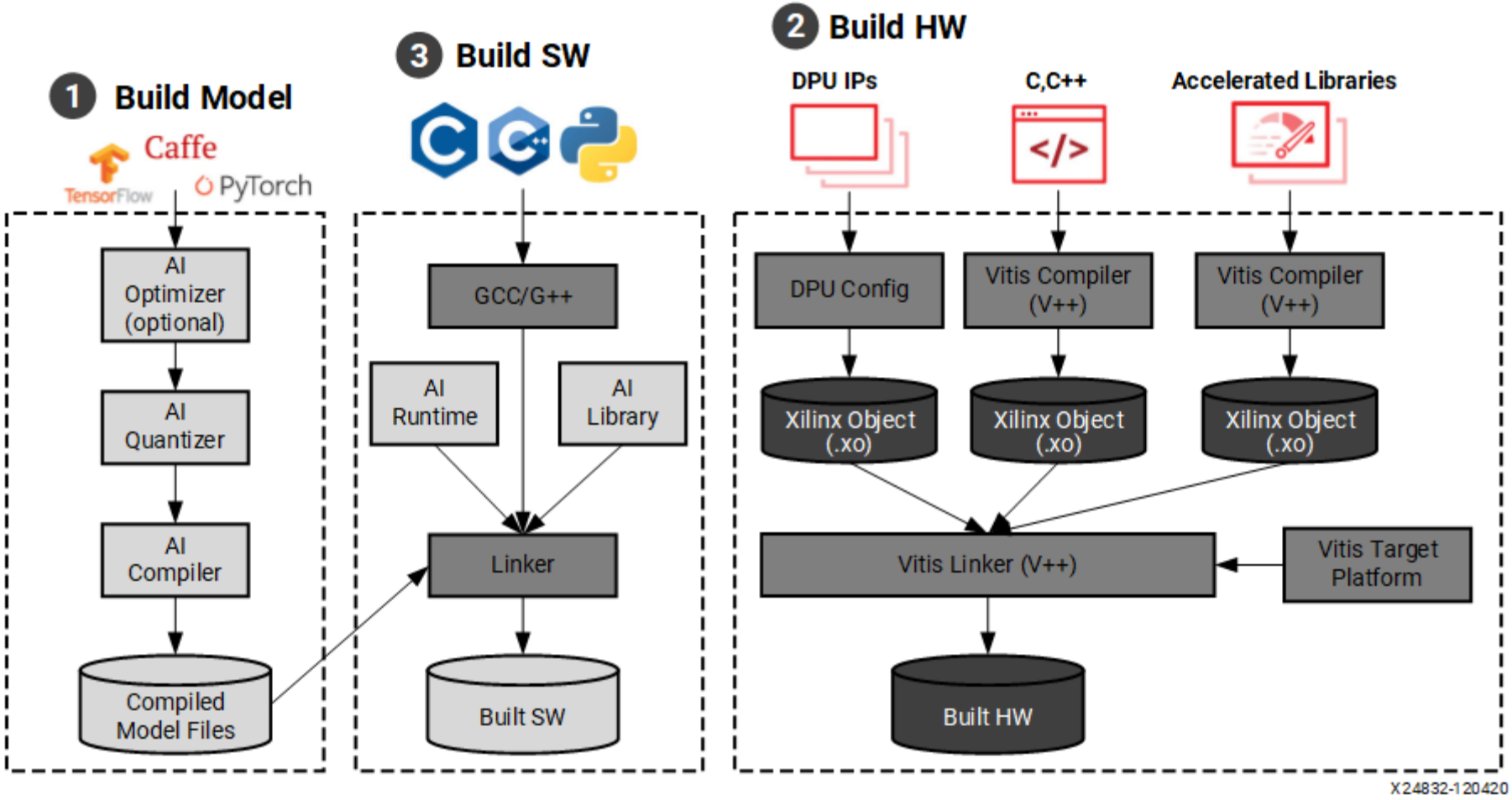
出典:Vitis AIユーザガイド(UG1414) ver.3.0
https://docs.xilinx.com/r/en-US/ug1414-vitis-ai/Vitis-AI-Overview

Vitis™ AIの構成

- Vitis-AIコンテナ（以下を含むDockerコンテナ）
  - AIフレームワーク：モデルの構築・学習（Pytorch/TensorFlow2）
  - Vitis-AIクオンタイザー：8bit量子化処理
  - Vitis-AIオプティマイザー：Pruning(枝刈り)
  - Vitis-AIコンパイラ：学習済みモデルをDPUの実行形式に変換
- Vitis-AI Model Zoo：DPUに対応した様々なモデルを提供
- Vitis-AIライブラリ：AIタスクに応じたモデルとAPIを提供
- VItis-AIランタイム（VART）：FPGAに配置されたアクセラレータ（DPUを含む）を駆動するためのAPIを提供

- これらを利用した開発フローは？

Vitis™ AI開発フローの概要



出典:Vitis AIユーザガイド(UG1414) ver.3.0
https://docs.xilinx.com/r/en-US/ug1414-vitis-ai/Development-Flow-Overview

Vitis™ AI開発フローの概要

- 1. Build Model
  - Vitis™ AI開発環境(Docker コンテナ）で以下を実施
    - AIフレームワーク（Pytorch/TensorFlow2）でモデルを構築・学習
    - Vitis-AIオプティマイザーで最適化（Optional)
    - Vitis-AIクオンタイザーで量子化
    - Vitis-AIコンパイラでDPU実行形式に変換
- 2. Build HW
  - Vitis/Vivadoを用いてFPGAに専用ハードウェアを作成
    - DPUの設定
    - Vitis HWプラットフォーム作成(必要に応じてVivadoも使用)
- 3. Build SW
  - Python/C/C++でアプリケーションを作成
  - Vitis-AIランタイム、Vitis-AIライブラリを使用
  - OSはlinux(petalinux/ubuntu)

Vitis™ AI開発フローの概要

- 1. Build Model
  - Vitis™ AI開発環境(Docker コンテナ）で以下を実施
    - AIフレームワ（　　）でモデルを構築・学習
    - Vitis-AIオプ（　　）
    - Vitis-AIクオ（　　）
    - Vitis-AIコンパイラでDPU実行形式に変換
- 2. Build HW
  - Vitis/Vivadoを（　　）アを作成
    - DPUの設定
    - Vitis HWプラ（　　）Vivadoも使用)
- 3. Build SW
  - Python/C/C++（　　）
  - Vitis-AIランタ（　　）用
  - OSはlinux(pet（　　）

まぁ、わかる

え、さっぱりわからん

わかる

Vitis™ AI開発フロー、はじめの一歩

ここまでの開発フローは「フルスクラッチ」の場合
手軽に始めるためには…

- 1. Build Model
  - AIモデルの構築・学習をする代わりにAMD提供のモデルを利用
    Vitis-AI Model Zoo＆ Vitis™ AIライブラリ
    →学習済み（一部、最適化済み）のモデルが利用可能
- 2. Build HW
  - AMD製「Kria KV260 ビジョン AI スターター キット」を利用
    →kria-apps-firmwareで提供されるHWが利用可能
- 3. Build SW
  - AMD製「Kria KV260 ビジョン AI スターター キット」を利用
    → Vitis™ AIライブラリ、 Vitis™ AIランタイムを導入済みの
    Petalinux/Ubuntu起動イメージが利用可能
  - Vitis™ AIライブラリを利用：タスクに応じたAIモデルとAPIが利用可能、実装の参考となるsampleプログラムも提供

目次

- エッジAIについて
- エッジAIデバイスとしてFPGA
- Vitis™ AIの紹介
- Vitis™ AIライブラリについて
- ハードウェア：Kria KV260 ビジョンAI スターターキット
- Vitis™ AIライブラリを試してみる
- よりエッジAIらしいアプリケーションへ

まずはVitis™ AIライブラリを眺めてみる

- Vitis™ AIライブラリのドキュメント
  Vitis AI ライブラリ ユーザー ガイド (UG1354) 3.0日本語版
  https://docs.xilinx.com/r/ja-JP/ug1354-xilinx-ai-sdk
  より抜粋

“対象者

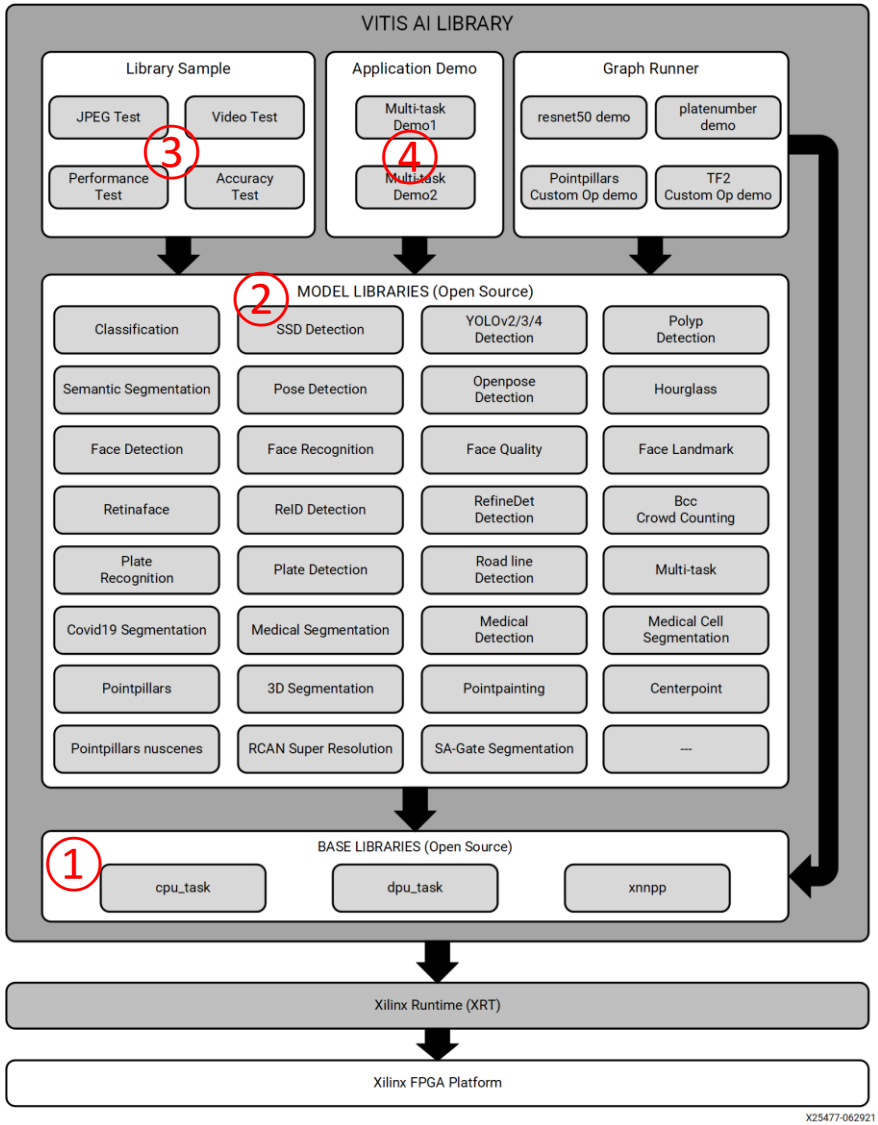
Vitis AI ライブラリの使用は、次に該当するユーザーを対象としています。

- ザイリンクスの学習済みモデルを使用して迅速にアプリケーションを構築する必要がある
- Vitis AI ライブラリ サポート ネットワーク リストを使用して、独自のデータセットで学習したカスタム モデルを使用する
- Vitis AI ライブラリでサポートされているモデルと類似したカスタム モデルと、Vitis AI ポストプロセス ライブラリを使用する”

Vitis™ AIライブラリの機能

- Vitis AIライブラリの機能（UG1354より抜粋）
  - フルスタックのアプリケーション ソリューション
    →アプリケーション開発の手段を提供
  - 最適化されたプリプロセスおよびポストプロセスの関数/ライブラリ
    →AIモデルを利用するための前処理・後処理も含まれている
  - オープンソースのモデル ライブラリ
    →様々なAIモデルを提供
  - DPU との統一されたオペレーション インターフェイスと、モデルのプリプロセスおよびポストプロセスのインターフェイス
    →FPGAに配置されたDPUを制御するためのIFも含まれている
  - 実用的なアプリケーション ベースのモデル ライブラリ、プリプロセスおよびポストプロセスのライブラリ、アプリケーション サンプル
    →実用的なアプリケーションを作成するためのサンプルもある

Vitis-AIライブラリの構成



出典：UG1354

① ベースライブラリ
- DPUとのプログラムIF
- 各モデルで利用可能なポストプロセス モジュール

② モデルライブラリ
- 一般的なネットワークを含むほとんどのオープンソース ニューラル ネットワーク運用を実装

③ ライブラリサンプル
- モデル ライブラリを短時間でテスト/評価するために使用

④ アプリケーションデモ
- Vitis AI ライブラリを使用したアプリケーション開発方法を示す

Vitis™ AIライブラリを利用した開発

- UG1354の「プログラミング例」を読む
  - “アプリケーション要件のカテゴリ”によると
    - Vitis™ AIライブラリにあるモデルを使用して、独自アプリケーションを構築する
    - Vitis™ AI ライブラリのモデルに類似したユーザー独自のカスタム モデルを使用する

  さらに
  - “アプリケーションに応じた API の選択”によると
    - モデル、プリプロセス、ポストプロスなどの AI アルゴリズムを使用した経験がなく、最大限の性能が必要な場合は、API_1 AI_Library を推奨
    - ザイリンクスのモデルを使用して迅速にアプリケーションを構築する必要がある場合は、アプリケーションに応じた API の選択を推奨
    - Vitis AI ライブラリ サポート ネットワーク リストを使用して、独自のデータで再学習したカスタム モデルを使用する場合は、API_1 AI_Library を推奨

→“API_1 AI_Library”を使えば良さそう

目次

- エッジAIについて
- エッジAIデバイスとしてFPGA
- Vitis™ AIの紹介
- Vitis™ AIライブラリについて
- ハードウェア：Kria KV260 ビジョンAI スターターキット
- Vitis™ AIライブラリを試してみる
- よりエッジAIらしいアプリケーションへ

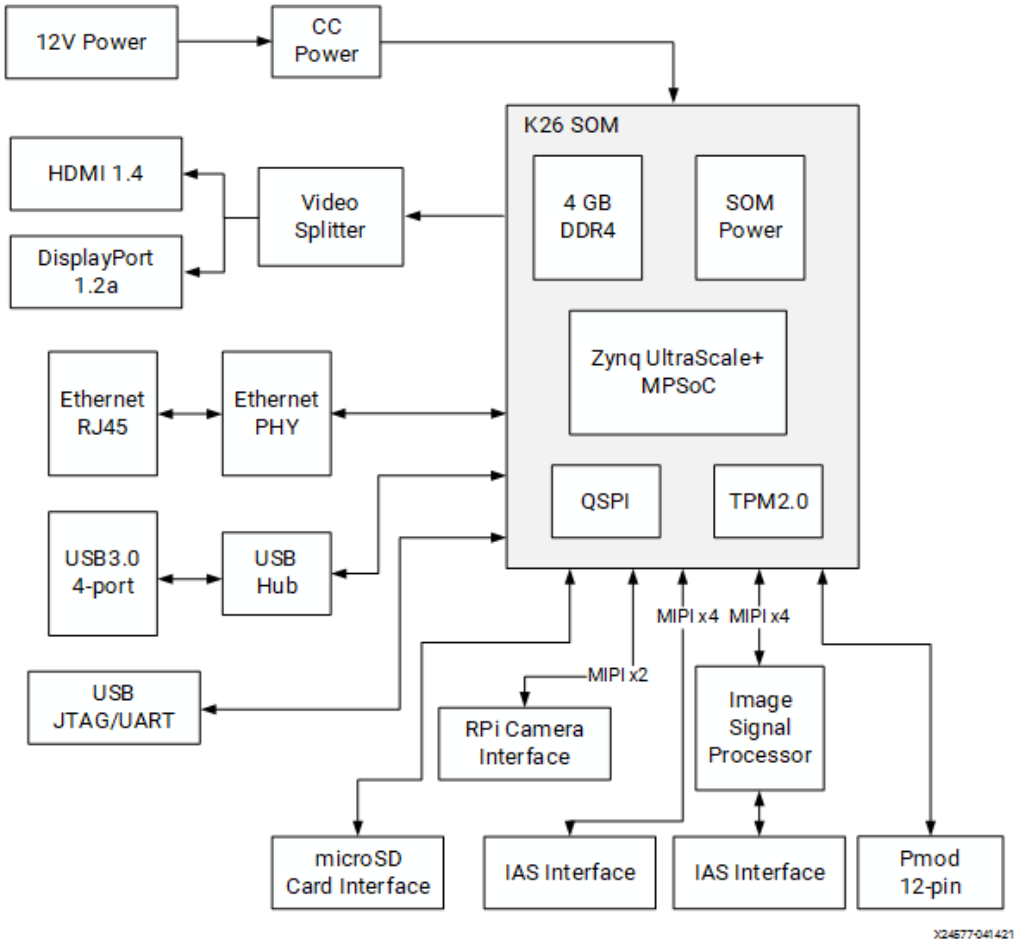
Kria KV260 ビジョン AI スターター キット

- AMDのKria SOMを搭載した開発プラットフォーム
  - AMDアダプティブSOC Zynq UltraScale+™ MPSoCを搭載したSOM Kria™ K26をベースにした製品
  - Vitis™ AIを利用したアプリケーションのサンプルも利用可能
  - Vitis™ AIランタイム、Vitis™ AIライブラリが導入済みのLinux環境SD cardイメージファイルを利用可能

出典： https://japan.xilinx.com/products/som/kria.html

KV260の構成

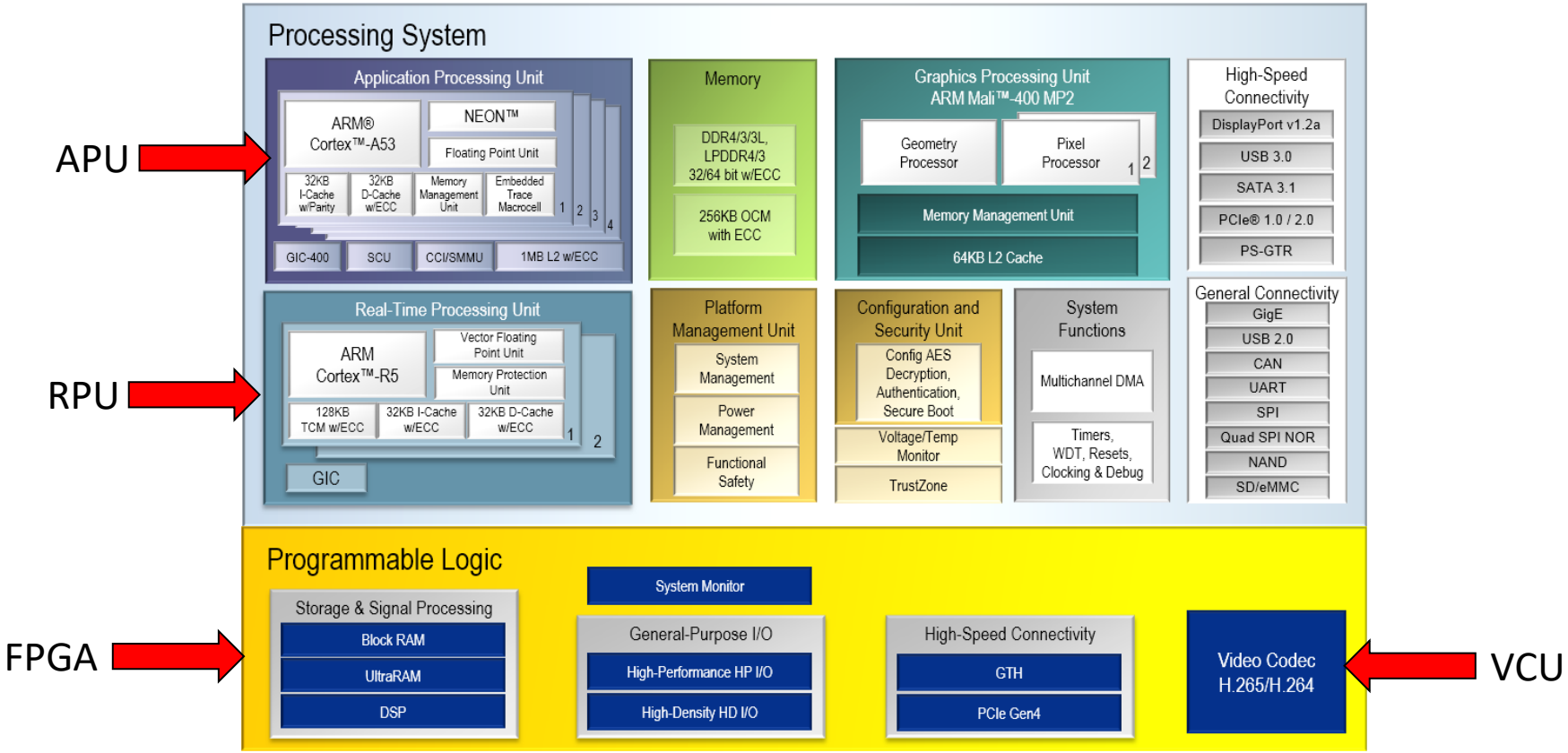
- K26 SOM+
  キャリアボード
- SOMにはSOC FPGA
  Zynq UltraScale+
  MPSoC
  - Ubuntu, Petalinuxが動作
- ペリフェラル
  - HDMI/DisplayPort
  - Ethernet
  - USB3.0
  - Rpi Camera I/F



出典：Kria KV260 ビジョン AI スターター キット データシート (DS986)

Zynq™ UltraScale+™ MPSoCの特徴

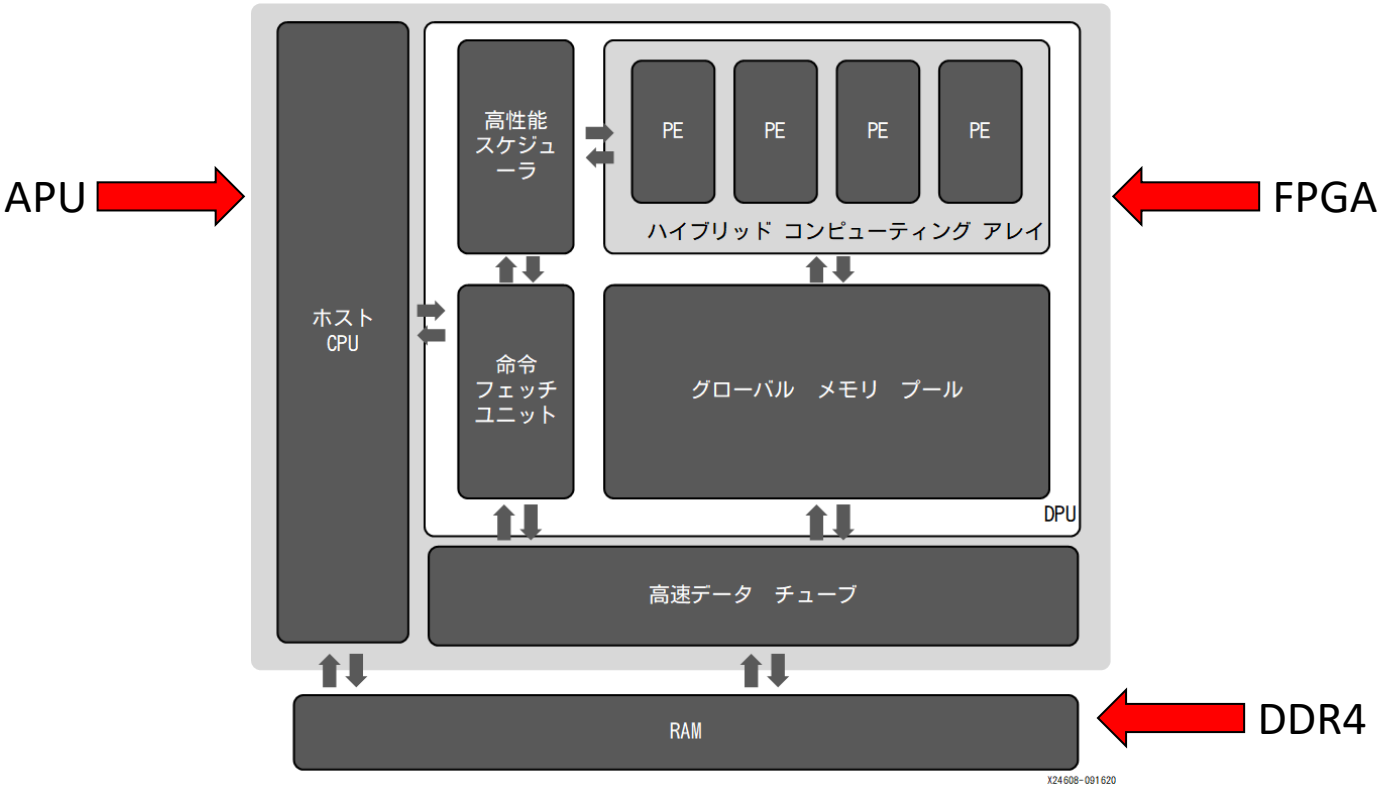
- K26 SOMはZynq UltraScale+ EVデバイスを搭載
  - APU/RPUともTOPPERS/FMPカーネルをサポート



出典：AMD Zynq™ UltraScale+™ MPSoC紹介ページ
https://japan.xilinx.com/products/silicon-devices/soc/zynq-ultrascale-mpsoc.html

Zynq® UltraScale+™ MPSoC 向けDPU

- DPUCZDX8Gのアーキテクチャ
  - 消費リソースはB4096, 3136, 2304～512まで可変
  - リソースに余裕があればx3コアまで拡張可能



出典:Vitis AIユーザガイド(UG1414) ver.3.0
https://docs.xilinx.com/r/ja-JP/ug1414-vitis-ai/Zynq-UltraScale-MPSoC-DPUCZDX8G

目次

- エッジAIについて
- エッジAIデバイスとしてFPGA
- Vitis™ AIの紹介
- Vitis™ AIライブラリについて
- ハードウェア：Kria KV260 ビジョンAI スターターキット
- Vitis™ AIライブラリを試してみる
- よりエッジAIらしいアプリケーションへ

Vitis™ AIライブラリのサンプルを試す準備

- HOST PCの準備：Vitis™ AIをインストール
  - 参照：Vitis AIユーザガイド(UG1414)“ホストのセットアップ”
- KV260の準備：起動用SDカードを作るだけ
  - 参照：Vitis AI ライブラリ ユーザー ガイド (UG1354)
    “ターゲットをセットアップする”
    - https://docs.xilinx.com/r/ja-JP/ug1354-xilinx-ai-sdk/ターゲットをセットアップする
  - Step1.ボード イメージをインストール
    - KV260用の起動イメージファイルを取得しSDカードに書き込み
    - Vitis™ AIモデルとライブラリのパッケージが導入済みのpetalinux
  - ~~Step2. AI モデル パッケージをインストール~~
    - Step1で利用したボード イメージファイルには導入済みのため不要
  - ~~Step3. AI ライブラリ パッケージをインストール~~
    - Step1で利用したボード イメージファイルには導入済みのため不要
  - KV260を起動（電源投入）

Vitis™ AIライブラリのサンプルを試す

- サンプルはKV260の以下に導入済み
  ~/Vitis-AI/examples/vai_library/samples
  - 参照：Vitis AI ライブラリ ユーザー ガイド (UG1354)
    “Vitis-AI-サンプルの実行”
    - https://docs.xilinx.com/r/ja-JP/ug1354-xilinx-ai-sdk/ Vitis-AI-サンプルの実行
- サンプルで使用する画像／動画データのダウンロード
  - 画像：vitis_ai_library_r3.0.x_images.tar.gz
  - 動画：vitis_ai_library_r3.0.0_video.tar.gz
  - “エッジの場合”にある手順に従い、KV260へSCPした後
    Vitis-AI/examples/vai_library　に解凍

- たとえば、facedetectのサンプルを試してみる

サンプル：facedetectの場合

- 場所：~/Vitis-AI/examples/vai_library/samples/facedetect

  以下のテスト用ビルド済みのバイナリとソースコードが配置
  - test_jpeg_facedetect：jpeg画像でのテスト
  - test_video_facedetect：動画でのテスト
  - test_performance_facedetect：モデルの性能テスト
  - test_accuracy_facedetect：モデルの精度テスト
    test_accuracy_facedetect_mt：精度テストのマルチスレッド版

  各サンプルには上記の４種類のテストサンプルが含まれる

- 実行結果の例
  - ./test_jpeg_facedetect densebox_320_320 sample_facedetect.jpg

サンプル：facedetectの場合

- 場所：~/Vitis-AI/examples/vai_library/samples/facedetect

バイ
eg画
画
tect
：モ
_m
のテ

ッド版

- ebox_320_320 sample_facedetect.jpg

Vitis™ AIライブラリのサンプルは55種類

caffe	TensorFlow	PyTorch
分類	分類	分類
顔検出	SSD 検出	ReID 検出
SSD 検出	YOLOv3 検出	顔認識
姿勢検出	医療画像検出	セマンティック セグメンテーション
セマンティック セグメンテーション	セマンティック セグメンテーション	Pointpillars
車線検出	RCAN 超解像	医療画像セグメンテーション
YOLOv2 検出	EfficientDet_D2	3D セグメンテーション
YOLOv3 検出	SuperPoint	Pointpillars_nuscenes: サラウンド ビュー
YOLOv4 検出	HFNet	CenterPoint: 4D レーダーに基づく 3D 検出
Openpose 検出		PointPainting: 画像と LiDAR のセンサー フュージョン
RefineDet 検出		深度推定
ReID 検出		ベイズ群衆計数
マルチタスク		マルチタスク V3
顔認識		ポリープ セグメンテーション
プレート検出		UltraFast 車線検出
プレート認識		FairMot
医療画像セグメンテーション		PSMNet
		SOLO
		CLOCs
		OCR
		TextMountain 検出
		車両分類
		OFA_YOLO 検出
		Monodepth2
		YOLOv5 検出
		BEVDet 検出
		cFlownet
		YOLOv6 検出

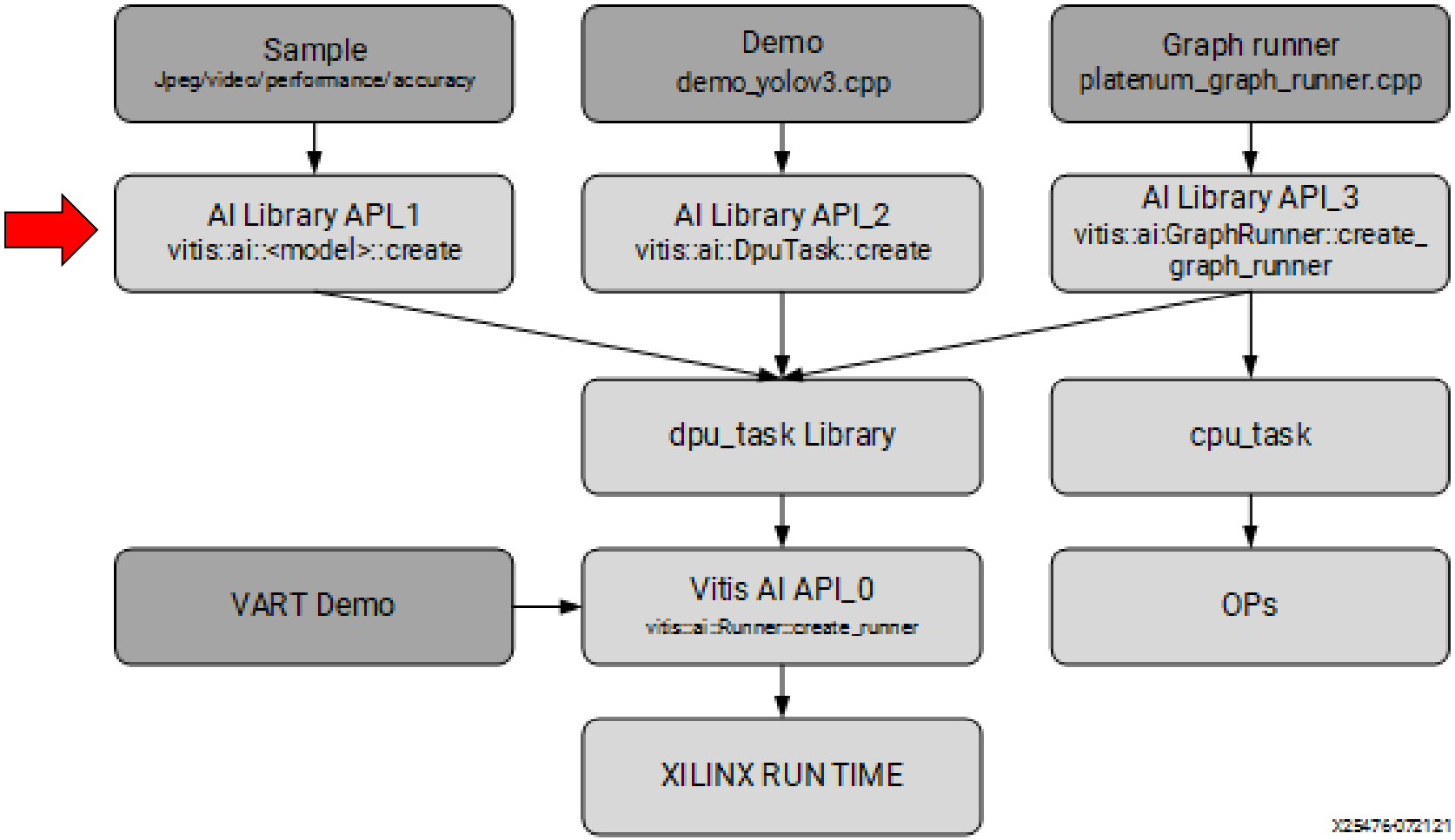
Vitis™ AIライブラリのAPI_1 AI_Libraryとは？

- Vitis™ AIライブラリのAPI_1 AI_Libraryを推奨する開発
  - モデル、プリプロセス、ポストプロスなどの AI アルゴリズムを使用した経験がなく、最大限の性能が必要な場合
  - ザイリンクスのモデルを使用して迅速にアプリケーションを構築する必要がある場合
  - Vitis AI ライブラリ サポート ネットワーク リストを使用して、独自のデータで再学習したカスタム モデルを使用する場合

- Vitis AI API_1 の特徴
  - 分類、検出、セグメンテーションなどの各種ビジョン タスクに関連する高レベルAPIベースのライブラリ セット
  - プリプロセスとポストプロセスを含むアルゴリズム フロー全体に最適化
  - Vitis™ AI Model Zoo のモデルをサポート

Vitis™ AIライブラリAPIとexampleの関係

- Vitis™ AIライブラリのサンプルはAPI_1を利用



出典：https://docs.xilinx.com/r/en-US/ug1354-xilinx-ai-sdk/Programming-Examples

例）vitis::ai::FaceDetect のAPI

- メンバ関数は以下のとおり
  - create：クラスFaceDetectの派生クラスのインスタンスを取得するファクトリ関数
  - run：facedetect ネットワークの実行結果を取得する関数
  - getThreshold：検出しきい値を取得する関数
  - setThreshold：検出しきい値を更新する関数

```
auto image = cv::imread("sample_facedetect.jpg");
auto network = vitis::ai::FaceDetect::create(
               "densebox_640_360",
               true);
auto result = network->run(image);
for (const auto &r : result.rects) {
   auto score = r.score;
   auto x = r.x * image.cols;
   auto y = r.y * image.rows;
   auto width = r.width * image.cols;
   auto height = r.height * image.rows;
}
```

出典：https://docs.xilinx.com/r/ja-JP/ug1354-xilinx-ai-sdk/vitis-ai-FaceDetect

補足：API_1 AI_Libraryが推奨されない場合

- 参照：Vitis AI ライブラリ ユーザー ガイド (UG1354)
  “プログラミング例”

  https://docs.xilinx.com/r/ja-JP/ug1354-xilinx-ai-sdk/プログラミング例
  - API_3 Graph_runner を推奨
    - モデルが複数のサブグラフに分割されている場合
    - モデルにカスタム op が含まれる場合

  →使用するAIモデルの構成による制限がある

  - API_2 DpuTask を推奨
    - カスタム プリプロセスまたはポストプロセス アルゴリズムを使用する必要がある場合

  →アプリケーションで使用する前処理／後処理による制限がある

  - API_0 VART を推奨
    - AI アルゴリズムの使用経験が豊富で、複数プラットフォーム上で AI アルゴリズムを開発および適用する必要がある場合

よりエッジAIらしいアプリケーションへ

- Vitis™ AIライブラリのAPI_1を用いることで
  - 様々なAIモデルが容易に扱える
  - 各AIモデルで有用な前処理・後処理がサポートされている

AI処理を行うアプリケーションSWは作れそう

でも、エッジAIデバイスとして成り立つのか？

- Vitis™ AIライブラリのサンプルはあくまでも
  - AIモデルをテストするためのコードのみ
  - HW構成も、FPGAにDPUが１つ載っているだけ

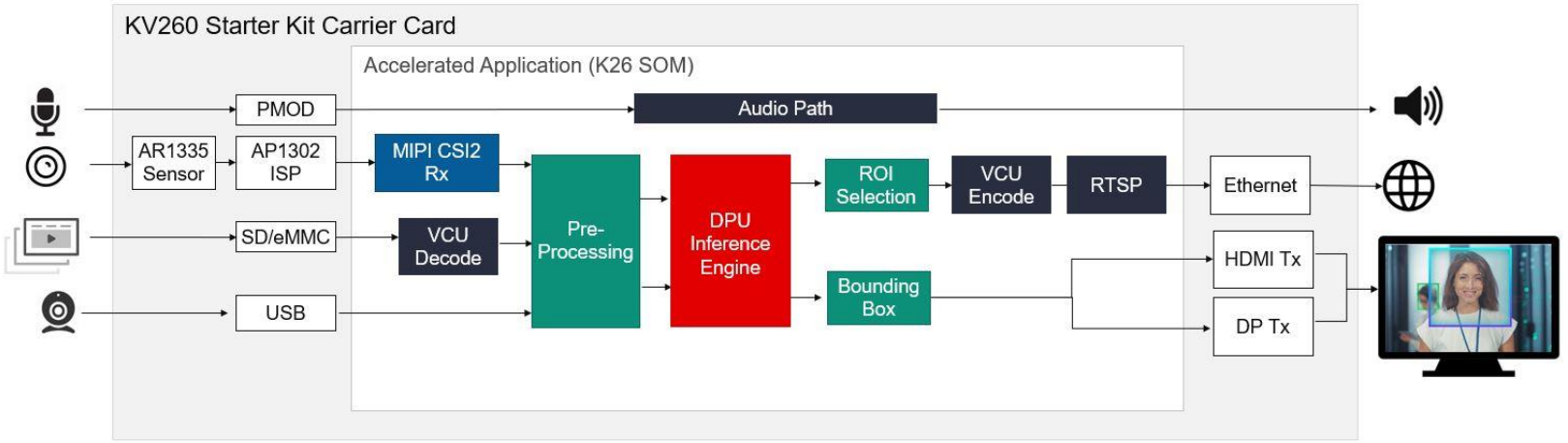
面白くない

Kria KV260 Vision AI Starter Kit Applications

- AMDがKV260向けに提供するアプリケーション
  - 参照：https://xilinx.github.io/kria-apps-docs/kv260/2022.1/build/html/index.html#kria-kv260-vision-ai-starter-kit-applications
- 以下のアプリケーションをTutorialとともに提供
  - Smart Camera
    顔検出機能を持つスマートカメラ
  - AI Box with ReID
    複数のカメラ入力からRe-Identificationで歩行者追跡
  - Defect Detection
    欠陥検出を行うマシンビジョンアプリケーション
  - NLP SmartVision
    音声での指示を識別してビジョンタスクを切り替えるデモ
  - AI Box Distributed ReID
    複数のAI Box with ReID分散管理するデモ

Smart Cameraの例

- スマートカメラの構成



出典：https://xilinx.github.io/kria-apps-docs/kv260/2022.1/build/html/docs/smartcamera/smartcamera_landing.html#smart-camera

- KV260のキャリアボード上のペリフェラル（薄灰色）のIOにあわせてSOM内部にIP（青色）やハードマクロ（黒色）を接続
- 中央に配置したDPU（赤色）
- タスクに合わせた前処理・後処理（緑色）

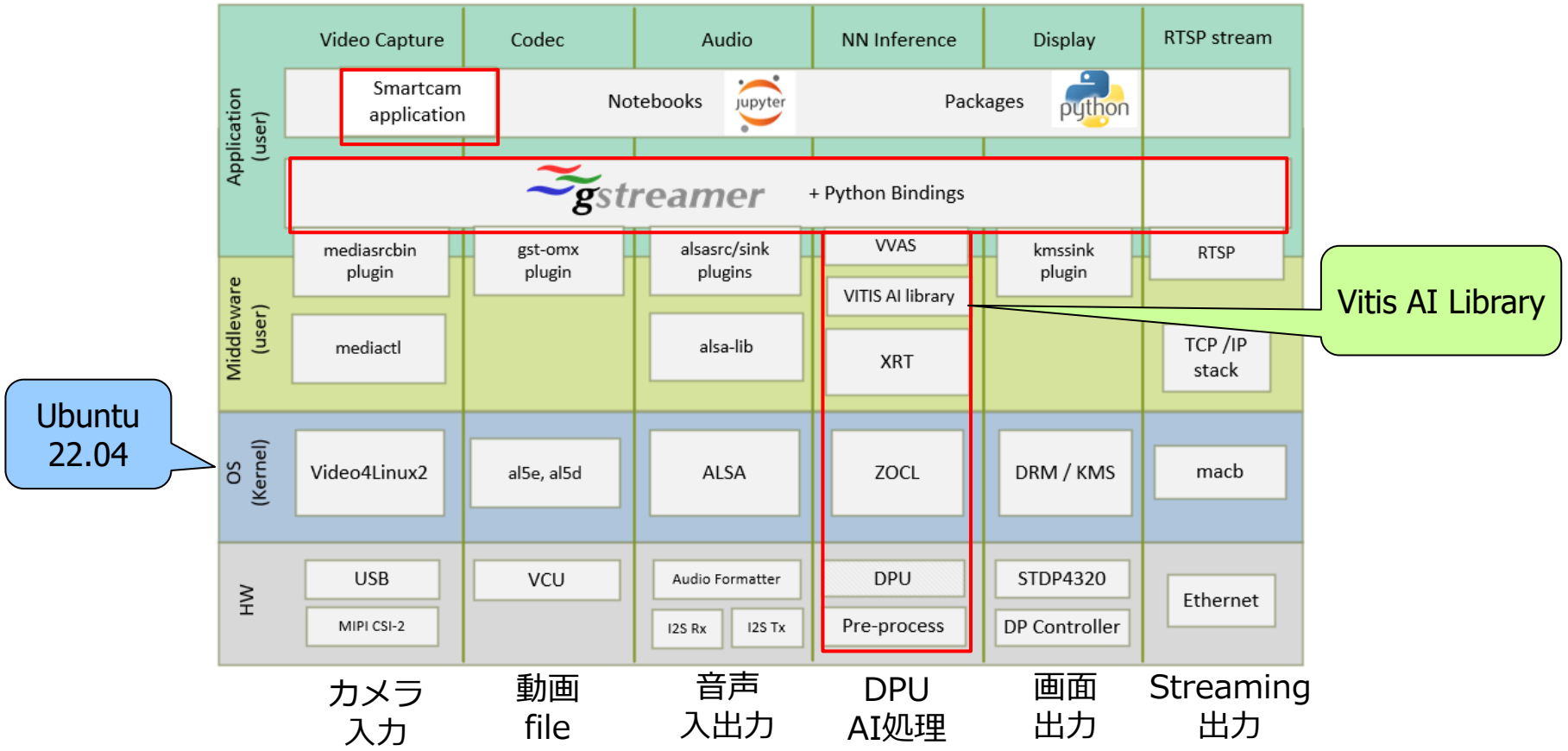
Smart Cameraの機能

- このSmart Cameraのアプリケーションは
  - 入力：動画ファイル、mipiカメラ、USB webcam
  - 出力：ディスプレイ(DP), file, Ethernet(RTSP)
  - AI処理：Vitis™ AIライブラリのfacedetect
    （ただし、小型のDPUになっているため若干性能減）

このプラットフォームを利用して、
AI処理を差し替える

Smart CameraのSWアーキテクチャ①

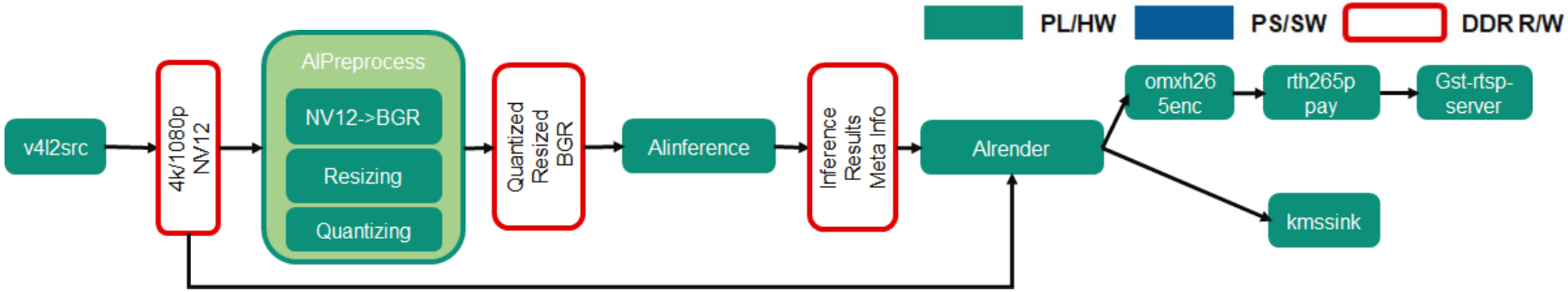
- ソフトウェアプラットフォーム（スタック）



出典：https://xilinx.github.io/kria-apps-docs/kv260/2022.1/build/html/docs/smartcamera/docs/sw_arch_platform.html#software-architecture-of-the-platform

Smart CameraのSWアーキテクチャ②

- AI処理周りのGStreamer パイプライン



  - 前処理、AI推論処理、後処理をFPGAで実施
  - タスクに応じてAI推論処理を入れ替え可能
- GStreamer パイプラインからAI処理へのIFはVVAS
  - Vitis Video Analytics SDK：AMD独自のGstreamer plugin
    - 前処理：vvas_xmultisrc Gstreamer plugin
    - AI推論：vvas_xfilter GStreamer plugin
    - 後処理：AI Rendering plugin
vvas_xfilter&libvvas_xboundingbox.so
  - VVAS pluginはjsonファイルで設定

https://xilinx.github.io/kria-apps-docs/kv260/2022.1/build/html/docs/smartcamera/docs/sw_arch_accel.html#gstreamer-pipeline

Smart CameraのAIモデル変更

- 参考：Customizing the AI Models Used in the Application
  - https://xilinx.github.io/kria-apps-docs/kv260/2022.1/build/html/docs/smartcamera/docs/customize_ai_models.html#customizing-the-ai-models-used-in-the-application

- このTutorialでは以下のAIモデルが差し替え対象
  - facedetect (densebox_320_320)
    →デフォルト：顔検出
  - refinedet (refinedet_pruned_0_96)
    →歩行者の検出
  - ssd (ssd_adas_pruned_0_95)
    →車両、歩行者などの位置を検出
  - yolov3（yolov3_coco）
    →一般物体検出、置き換えの手順が詳細に記載されている

AI処理（モデル）変更の概要

- 準備： Vitis-AIコンパイラで変換しなおす
  - ライブラリにあるコンパイル済みモデルはB4096向け
    Smart CameraのDPUはB3136のため再変換が必要
    - DPUとコンパイル時の情報が異なった場合、fingerprint不一致で実行エラーとなる

- VVAS pluginの設定変更
  - 前処理：vvas_xmultisrc Gstreamer plugin
    /opt/xilinx/kv260-smartcam/share/vvas/${AI_TASK}/preprocess.json
    色空間の変換パラメータなど設定
  - AI推論：vvas_xfilter GStreamer plugin
    /opt/xilinx/kv260-smartcam/share/vvas/refinedet/aiinference.json
    参照するAIモデルの情報、パラメータ、HWの情報などを設定
    Vitis™ AIライブラリを隠蔽している
  - 後処理：AI Rendering plugin
    vvas_xfilter&libvvas_xboundingbox.so
    drawresult.json

  →詳細はVVASの仕様を確認してください。

まとめ

- エッジAIデバイスとして、FPGAは利用価値がある
  ただし、FPGAで勝つためには様々な工夫が必要
- Vitis™ AIはAMDのAI 推論開発プラットフォーム
  エッジからクラウド／データセンターまで包括
- Vitis™ AIライブラリを活用すると、様々な学習済みモデルでAI推論タスクが可能（ Vitis™ AI Model zoo）
- Kria KV260 Vision AI Starter Kit Applicationsを活用
  よりエッジAIらしいアプリケーションが作成可能

エッジAIの「はじめの一歩」を気楽に始めましょう

参考資料

[1] Vitis AI ユーザー ガイド (UG1414) v3.0

https://docs.xilinx.com/r/ja-JP/ug1414-vitis-ai

[2] Vitis AI ライブラリ ユーザー ガイド (UG1354) v3.0

https://docs.xilinx.com/r/ja-JP/ug1354-xilinx-ai-sdk

[3] AMD Vitis AI Documentation v3.0

https://xilinx.github.io/Vitis-AI/3.0/html/index.html

[4] Vitis-AI v3.0 レポジトリ

https://github.com/Xilinx/Vitis-AI/tree/v3.0

[5] AMD SOM Landing Page

https://xilinx.github.io/kria-apps-docs/home/build/html/index.html

[6] Xilinx Wiki Kria K26 SOM

https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/1641152513/Kria+K26+SOM

参考資料

[7] Xilinx Wiki Canonical Ubuntu

https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/1413611532/Canonical+Ubuntu

[8] Zynq UltraScale+ MPSoC 向け DPUCZDX8G 製品ガイド (PG338)

https://docs.xilinx.com/r/ja-JP/pg338-dpu