Versal ACAP AI エンジン

アーキテクチャ マニュアル

AM009 (v1.1) 2021 年 4 月 22 日

XILINX®

改訂履歴

次の表に、この文書の改訂履歴を示します。

セクション	改訂内容
2021 年 4 月 22 日 バージョン 1.1	
全般	『Versal ACAP AI エンジン レジスタ リファレンス』(AM015) へのリンクを追加。
メモリ エラー処理	メモリマップド AXI4 アクセス時のパフォーマンス カウンターの動作を明記。
ALU、スカラー関数、およびデータ型変換	オーバーフロー例外の `Float2fix` 変換についてのインプリケーションの説明を追加。
2020 年 7 月 16 日 バージョン 1.0	
初版	N/A

目次

XILINX

第 1 章

概要

Versal ACAP の概要

Versal™ ACAP (Adaptive Compute Acceleration Platform) はスカラー エンジン、適応型エンジン、およびインテリジェント エンジンを中心に、最先端のメモリおよびインターフェイス テクノロジを組み合わせることによって、あらゆるアプリケーションで強力なヘテロジニアス アクセラレーションを実現します。Versal ACAP で最も重要な点は、ソフトウェア開発者やデータ サイエンティストがハードウェア開発者と同様にハードウェアとソフトウェアをプログラムおよび最適化できることにあります。Versal ACAP は、さまざまなツール、ソフトウェア、ライブラリ、IP、ミドルウェア、およびフレームワークでサポートされ、業界標準のデザイン フローを活用できます。

TSMC 社の 7nm FinFET プロセス テクノロジを採用した Versal ポートフォリオは、ソフトウェア プログラマビリティと特定分野に向けたハードウェア アクセラレーションに適応性を兼ね備え、現代の急速なイノベーションに対応できるようにした初のプラットフォームです。6 つのシリーズで構成されるこのデバイス ポートフォリオは、独自のアーキテクチャによりクラウド、ネットワーク、無線通信、エッジ コンピューティング、エンドポイントなど、幅広い市場における多くのアプリケーションで優れたスケーラビリティと AI 推論能力を発揮します。

Versal アーキテクチャは、異なるタイプのエンジン、さまざまなコネクティビティおよび通信機能、ネットワーク オン チップ (NoC) を組み合わせており、デバイスの高さおよび幅全体にスムーズなメモリ マップド アクセスが可能です。インテリジェント エンジンは、適応型の推論および高度な信号処理演算向けの SIMD VLIW AI エンジンと、固定小数点、浮動小数点、および複素 MAC 演算向けの DSP エンジンです。適応型エンジンは、高い演算密度を達成できるよう設計されたプログラマブル ロジック ブロックとメモリです。スカラー エンジンには Arm® Cortex®-A72 および Cortex-R5F プロセッサが含まれ、計算負荷の高いタスクを可能にします。

VersalAI コア シリーズは、現在のサーバー クラス CPU の 100 倍以上の演算性能を達成する AI エンジンを備え、AI 推論を飛躍的に高速化します。このシリーズは、動的ワークロードに対応したクラウドや、超高帯域ネットワークなど幅広いアプリケーションをサポートすると同時に、最先端の安全性とセキュリティ機能を提供します。ソフトウェアおよびハードウェア開発者だけでなく、AI およびデータ サイエンティストも高い演算密度を活用してあらゆるアプリケーションの性能を高速化できます。

Versal プライム シリーズは、Versal プラットフォームの基盤となるミッドレンジ デバイスで、さまざまな市場で幅広く使用できます。具体的なアプリケーションとしては、100G ～ 200G ネットワーク機器、データセンターのネットワークおよびストレージ アクセラレーション、通信テスト装置、放送機器、航空宇宙/防衛機器などがあります。このシリーズはメインストリームの 58G トランシーバーおよび最適化された I/O および DDR コネクティビティを統合し、幅広いワークロードにおいて低レイテンシの高速化と性能を達成します。

Versal プレミアム シリーズ は、消費電力とフットプリントを最小限に抑えた適応型プラットフォームで、画期的なヘテロジニアス統合、超高性能演算、コネクティビティ、セキュリティを実現します。このシリーズは、ワイヤード通信、データセンター、テスト/計測などの広帯域幅で演算負荷の高いアプリケーションにおける要件を十分に満たすよう設計されています。Versal プレミアム シリーズ ACAP には、112G PAM4 トランシーバー、600G イーサネット用の統合ブロック、600G Interlaken、PCI Express® Gen5、および高速暗号化エンジンが含まれます。

Versal アーキテクチャのすべての 資料は、https://japan.xilinx.com/versal から参照できます。

設計プロセス別のコンテンツ ガイド

ザイリンクスの資料は、開発タスクに関連する内容を見つけやすいように、標準設計プロセスに基づいて構成されています。Versal™ ACAP デザイン プロセスのデザイン ハブは、ザイリンクス ウェブサイトからアクセスできます。この資料では、次の設計プロセスについて説明します。

- システム/ソリューション プランニング: システム レベルのコンポーネント、パフォーマンス、I/O、およびデータ転送要件を特定します。ソリューションの PS、PL、および AI エンジン へのアプリケーション マップも含まれます。この設計プロセスに該当するトピックは、次のとおりです。
  - 第 1 章: 概要 では、AI エンジン アーキテクチャについて概説します。次の内容が含まれます。
    - AI エンジン アレイの概要
    - AI エンジン アレイ階層
    - 性能
  - 第 2 章: AI エンジン タイルのアーキテクチャ では、メモリ モジュールとインターコネクト、および AI エンジンとメモリ モジュールの連携動作について説明します。
  - 第 3 章: AI エンジン アレイ インターフェイスのアーキテクチャ では、PL および NoC への AI エンジン アレイ インターフェイスについて概説します。
  - 第 4 章: AI エンジンのアーキテクチャ では、プロセッサのファンクション ユニットおよびレジスタ ファイルについて説明します。
  - 第 5 章: AI エンジンのコンフィギュレーションとブート では、ブートおよびリコンフィギュレーション時にプロセッシング サブシステムから AI エンジン アレイをコンフィギュレーションする方法について説明します。
- AI エンジン開発: AI エンジン グラフおよびカーネルの作成、ライブラリの使用、シミュレーションのデバッグおよびプロファイリング、アルゴリズムの開発を実行します。PL と AI エンジン カーネルの統合も含まれます。この設計プロセスに該当するトピックは、次のとおりです。
  - 第 2 章: AI エンジン タイルのアーキテクチャ
  - 第 4 章: AI エンジンのアーキテクチャ

AI エンジン アレイの機能

一部の Versal ACAP は AI エンジン アレイを内蔵しています。AI エンジン アレイは、アレイ状に並んだ AI エンジン タイルと AI エンジン アレイ インターフェイス (ネットワーク オン チップ (NoC) インターフェイス タイルとプログラマブル ロジック (PL) インターフェイス タイルを含む) で構成されます。次に、それぞれの機能を示します。

AI エンジン タイルの機能

- プログラマブル ロジック (PL) 外部のシリコンに個別の構築ブロックとして統合
- 各 AI エンジンに信号処理や機械学習など多くのアプリケーションに最適化された高性能 VLIW (Very-Long Instruction Word) SIMD (Single-Instruction Multiple-Data) ベクター プロセッサを内蔵
- 8 バンクのシングル ポート データ メモリ (合計 32KB)
- AI エンジンと Versal デバイスのプログラマブル ロジック間で確定的なスループットおよび高速データフローを実現するストリーミング インターコネクト
- 受信ストリームからローカル メモリ、およびローカル メモリから送信ストリームへデータを移動するダイレクト メモリ アクセス (DMA) を AI エンジン タイルに内蔵

- 外部マスターから内部 AI エンジン タイルへアクセスするためのトランザクション ベースの共有スイッチド インターコネクトを備えたコンフィギュレーション インターコネクト (メモリ マップド AXI4 インターフェイス経由)
- AI エンジンとタイル DMA 間、および AI エンジンと外部マスター間 (メモリ マップド AXI4 インターフェイス経由) で AI エンジンの同期を実行するハードウェア同期プリミティブ (ロックなど)
- デバッグ、トレース、およびプロファイル機能

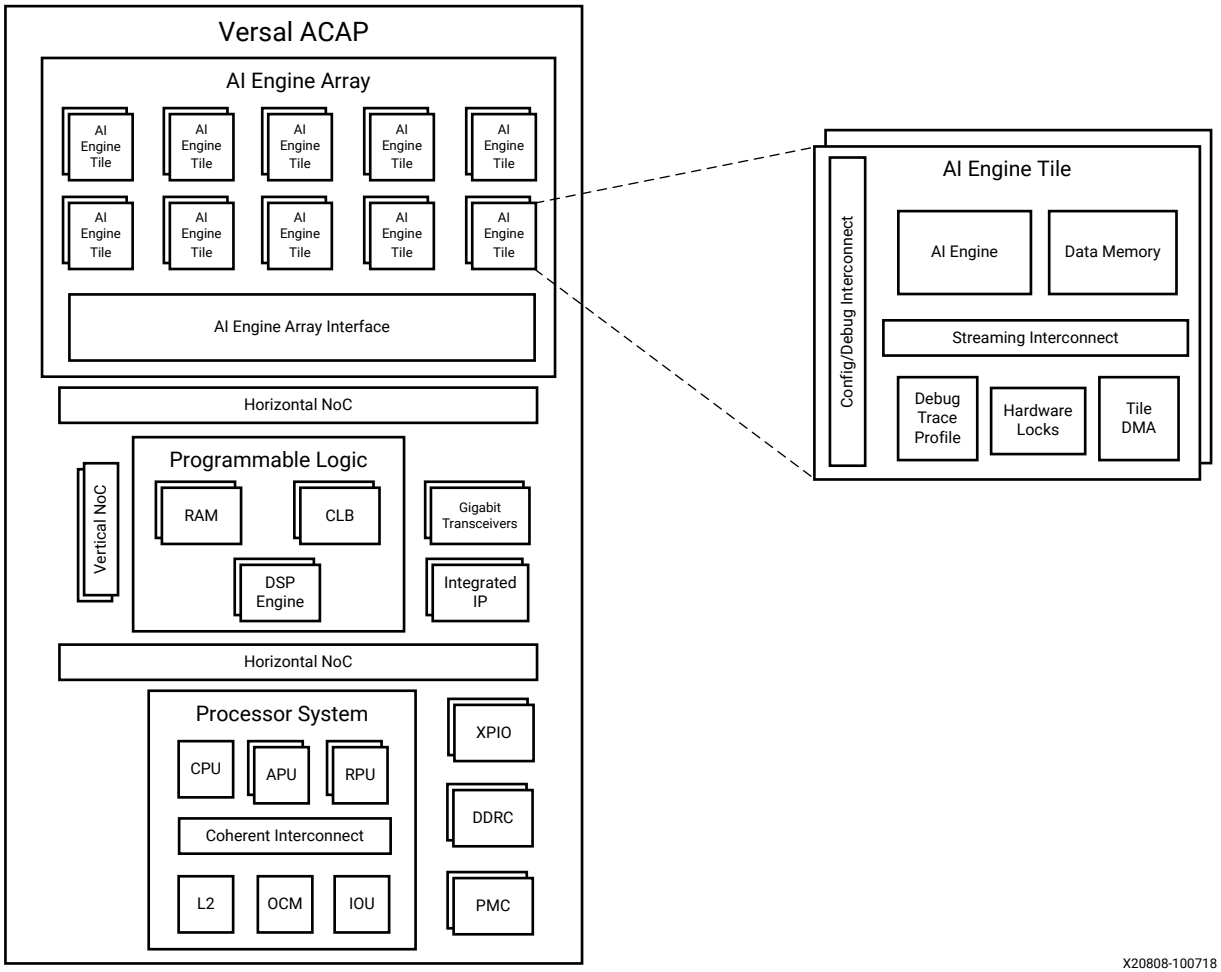
NoC および PL リソースへの AI エンジン アレイ インターフェイス

- AI エンジン アレイ インターフェイスの NoC インターフェイス タイルにはダイレクト メモリ アクセス (DMA) があり、AI エンジン アレイとの間で送受信するメモリ マップドおよびストリーム トラフィックを管理
- メモリ マップド AXI4 インターフェイスを経由したコンフィギュレーションおよび制御インターコネクト機能
- AI エンジン タイルのストリーミング インターコネクト機能を利用するストリーミング インターコネクト
- AI エンジンからプログラマブル ロジック (PL) へのインターフェイスにより、AI エンジン クロックと PL クロック間で非同期クロック乗せ換えを実行
- AI エンジンから NoC へのインターフェイス ロジックにより、NoC マスター ユニット (NMU) および NoC スレーブ ユニット (NSU) コンポーネントに接続
- AI エンジン タイルのロック モジュールの機能を利用したハードウェア同期プリミティブ (ロックなど)
- AI エンジン タイルのすべての機能を利用したデバッグ、トレース、およびプロファイル機能

AI エンジン アレイの概要

次の図に、AI エンジン アレイを内蔵した Versal ACAP (Adaptive Compute Acceleration Platform) の概略ブロック図を示します。このデバイスは、プロセッシング システム (PS)、プログラマブル ロジック (PL)、および AI エンジン アレイで構成されます。

図 1: **Versal デバイスの最上位ブロック図**

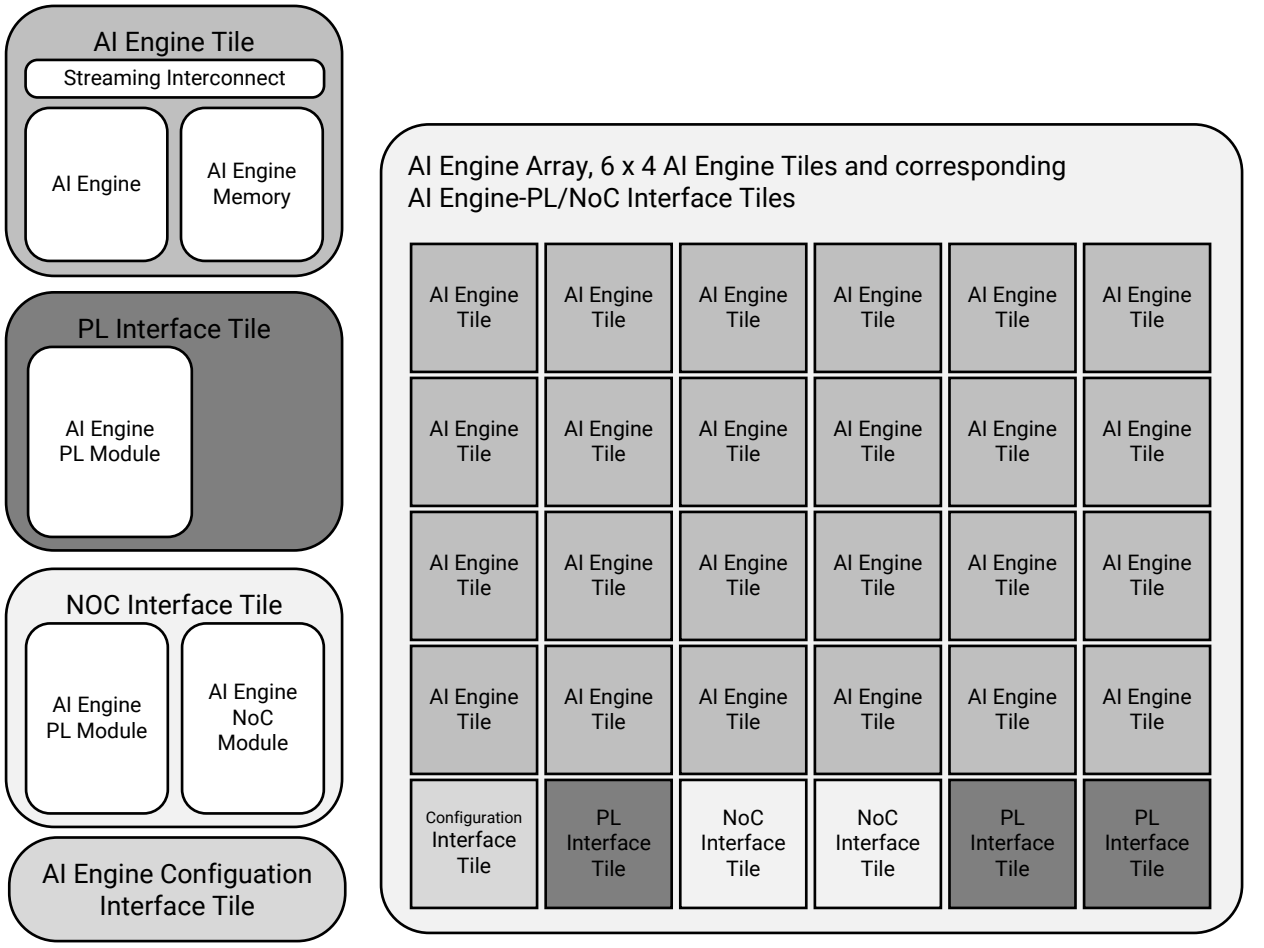


AI エンジン アレイは、AI エンジン アーキテクチャの最上位の階層です。これは、AI エンジン タイルを 2 次元に配列したものです。各 AI エンジン タイルは、VLIW (Very-Long Instruction Word) プロセッサ、内蔵メモリ、およびストリーミング、コンフィギュレーション、デバッグ用のインターコネクトを統合しています。AI エンジンは AI エンジン アレイ インターフェイスを使用して、NoC 経由で Versal デバイスのその他のブロックと通信するか、PL と直接通信します。 AI エンジン アレイは、プロセッシング システム (PS) およびプラットフォーム管理コントローラー (PMC) にも NoC 経 由で接続します。

AI エンジン アレイ階層

AI エンジン アレイは、AI エンジン タイルと AI エンジン アレイ インターフェイス タイル (AI エンジン アレイの一番下の行) で構成されます。インターフェイス タイルには、AI エンジンから PL へのインターフェイス タイルと AI エンジンから NoC へのインターフェイス タイルがあります。また、各 AI エンジン アレイには 1 つだけコンフィギュレーション インターフェイス タイルもあります。この中には、AI エンジン クロック生成用の PLL およびその他のグローバル制御機能が含まれます。次の図に、AI エンジン アレイの完全なタイル階層の概念図を示します。各タイルの詳細は、第 2 章: AI エンジン タイルのアーキテクチャ および 第 3 章: AI エンジン アレイ インターフェイスのアーキテクチャ を参照してください。

図 2: **AI エンジン アレイのタイル階層**



AI エンジンのアプリケーション

Versal™ ACAP AI エンジンは、次世代ワイヤレス、機械学習やその他の演算負荷の高いアプリケーションの需要の加速度的な増加に対応することを目的に開発されました。AI エンジン、デュアルコア Arm® Cortex®-A72 および Cortex-R5F プロセッサ (PS)、および次世代プログラマブル ロジック (PL) と高帯域幅 NoC が緊密に統合されて、ACAP の新たなアーキテクチャを構成しています。AI エンジンと PL が相互に補完し合い、強力な処理能力に見合った機能を実行します。カスタム メモリ階層、AI インターコネクト上のマルチキャスト ストリーム機能、および AI に最適化されたベクター命令をサポートする Versal ACAP AI エンジンは、さまざまな演算負荷の高いアプリケーションに最適化されています。アプリケーションの例として、従来の無線機能に加えてワイドバンド/マルチバンド機能をサポートする先進の無線システム、5G ワイヤレス通信 (ベクター DSP ベースの ASIC は不要)、許容範囲内の性能で確定的なレイテンシおよび低いニューラル ネットワーク レイテンシを実現する、データセンター アプリケーションにおける機械学習推論の高速化が挙げられます。

性能

AI エンジン アレイは、すべてのタイルおよびアレイ インターフェイス ブロックで 1 つのクロック ドメインを使用します。-1L スピード グレード デバイスの場合、AI エンジンの性能目標は 1GHz (V_{CCINT} = 0.70V) です。また、AI エンジン アレイにはほかのブロックへのインターフェイス用のクロックもあります。次の表に、AI エンジン アレイの各種クロックとその性能目標をまとめます。

表 1: **AI エンジンに関連するクロック ドメイン**

クロック	性能目標 (-1L)	ソース	AI エンジン クロックとの関係
AI エンジン アレイ クロック	1GHz	AI エンジン PLL	N/A
NoC クロック	800MHz	NoC クロッキング	非同期、NoC 内でクロック乗せ換え
PL クロック	500MHz	PL クロッキング	非同期、AI エンジン アレイ インターフェイス内でクロック乗せ換え
NPI クロック	300MHz	NPI クロッキング	非同期

メモリ エラー処理

メモリ エラーの検出と訂正

各 AI エンジンには 32KB のデータ メモリと 16KB のプログラム メモリがあります。多くの AI エンジン タイルを持つデバイスでは、ソフト エラーからの保護が必要となるため、そのための対策がとられています。プログラム メモリの 128 ビット ワードは、2 つの 8 ビット ECC (64 ビットごとに 1 つ) で保護されます。8 ビット ECC は、64 ビット ワード内の 2 ビット エラーを検出し、1 ビット エラーを検出および訂正できます。2 つの 64 ビット データと 2 つの 8 ビット ECC フィールドは、それぞれのペア内でインターリーブされ (距離は 2)、ビット分離が拡大するようにしています。

各データ メモリ モジュールには、8 つのメモリ バンクがあります。最初の 2 つのメモリ バンクは、4 つの 32 ビット フィールドのそれぞれを 7 ビット ECC で保護します。7 ビット ECC は 2 ビット エラーを検出し、1 ビット エラーを検出および訂正します。最後の 6 つのメモリ バンクは、128 ビット ワードの各 32 ビットを偶数パリティ ビットで保護します。4 つの 32 ビット フィールドは、距離が 4 でインターリーブされます。

プログラム メモリとデータ メモリはいずれもエラー挿入をサポートしています。プログラム メモリには、メモリ マップド AXI4 を介してエラーを挿入できます。同様に、データ メモリ バンクには AI エンジン DMA またはメモリ マップド AXI4 を介してエラーを挿入できます。

メモリ マップド AXI4 が AI エンジンのデータ メモリに対して読み書きを実行する場合、2 つの要求がメモリ モジュールに送信されます。AI エンジンのパフォーマンス カウンターでは、ECC/パリティ イベントでイベントが二度カウントされる場合があります。メモリ アクセスは重複しますが、機能への影響はありません。イベントおよびパフォーマンス カウンターの詳細は、第 2 章: AI エンジン タイルのアーキテクチャ を参照してください。

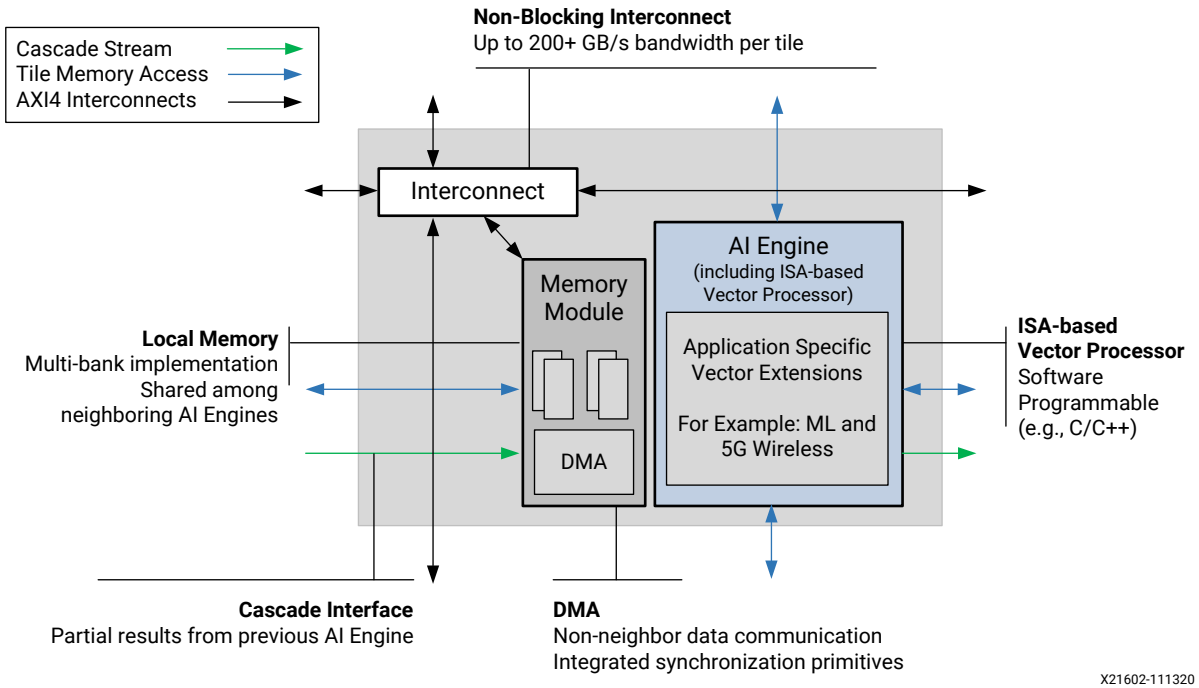
内部メモリ エラー (訂正可能および訂正不可能) が発生すると内部イベントが生成され、通常のデバッグ、トレースおよびプロファイリング メカニズムを使用してエラー状態が報告されます。これらを用いて、PMC/PS への割り込みを生成することもできます。

第 2 章

AI エンジン タイルのアーキテクチャ

次に、AI エンジン タイルのアーキテクチャ、主要な構築ブロック、および AI エンジン タイルのコネクティビティを含む最上位ブロック図を示します。

図 3: **AI エンジン タイルのブロック図**



AI エンジン タイルは、次のハイレベル モジュールで構成されます。

- タイル インターコネクト
- AI エンジン
- AI エンジン メモリ モジュール

タイル インターコネクト モジュールは、AXI4-Stream およびメモリ マップド AXI4 入出力トラフィックを処理します。メモリ マップド AXI4 および AXI4-Stream インターコネクトの詳細は、この後のセクションで説明します。AI エンジン メモリ モジュールは、8 つのメモリ バンクに分割された 32KB データ メモリ、メモリ インターフェイス、DMA、およびロックで構成されます。DMA は受信方向と送信方向の両方にあり、各メモリ モジュールには 1 つのロック ブロックがあります。AI エンジンは、4 方向すべてのメモリ モジュールに 1 つの連続したメモリ ブロックとしてアクセスできます。メモリ インターフェイスは、AI エンジンから生成されたアドレスに基づいて、メモリ アクセスを正しい方向へマップします。AI エンジンは各 1 個のスカラー プロセッサとベクター プロセッサ、3 個のアドレス ジェネレーター、および 16KB のプログラム メモリで構成されます。また、アキュムレータ出力を次の AI エンジン タイルに転送するためのカスケード ストリーム アクセスもあります。AI エンジンの詳細は、第 4 章: AI エンジンのアーキテクチャ で説明します。AI エンジンと AI エンジン メモリ モジュールの両方に制御、デバッグ、およびトレース ユニットがあります。これらユニットの一部については、この章で後述します。

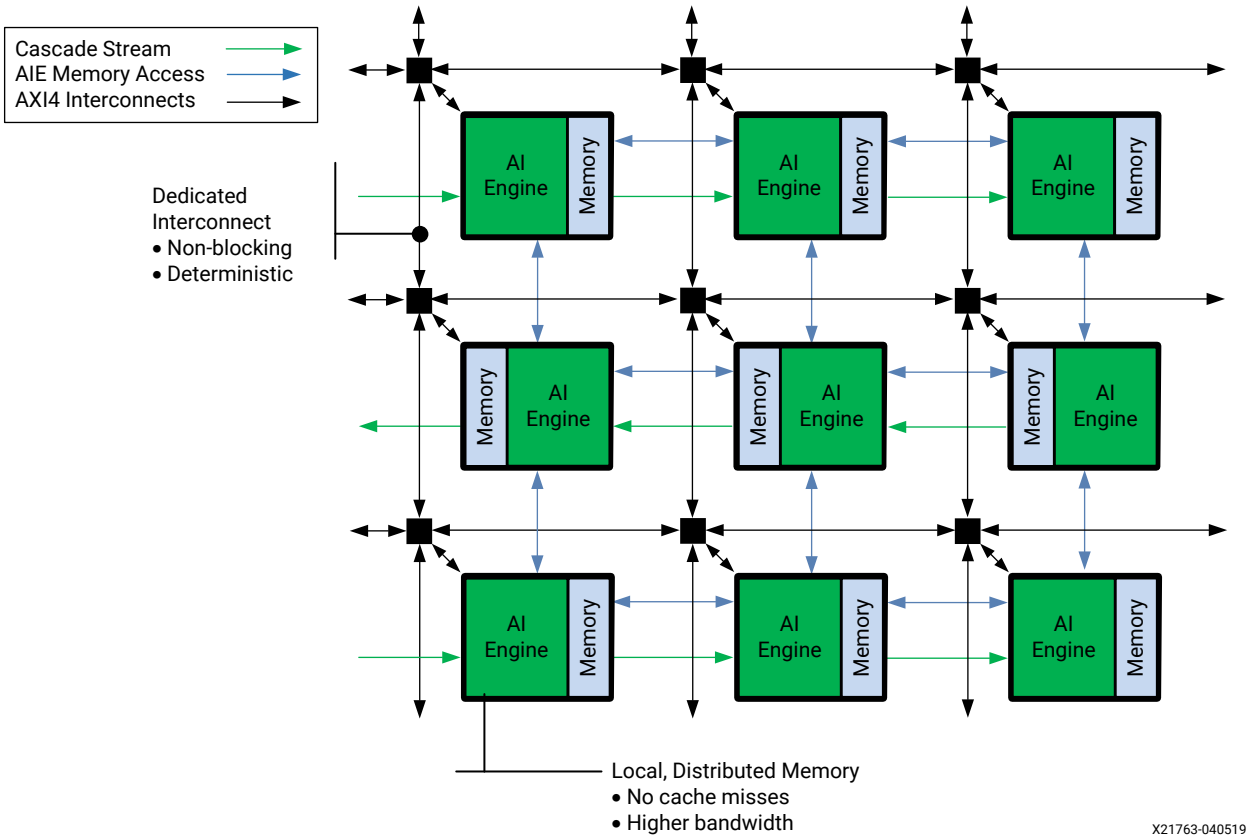
- 制御およびステータス レジスタ
- イベント、イベント ブロードキャスト、およびイベント アクション
- プロファイリング用のパフォーマンス カウンターとタイマー

次の図に、AI エンジン タイルおよび専用インターコネクト ユニットを配列した AI エンジン アレイを示します。AI エンジン アレイ内での主なデータ移動手段として、隣接する AI エンジンのローカル メモリを使用してデータを共有します。各 AI エンジンは、次に示す最大 4 つのメモリ モジュールにアクセスできます。

- AI エンジン自身のモジュール
- 上側のモジュール
- 下側のモジュール
- 右側または左側のモジュール (行および AI エンジンとメモリ モジュールの位置関係による)

格子状のパターンに配列されたアレイの端にある AI エンジンは、アクセスできるメモリ モジュールの数が 1 つまたは 2 つ少なくなります。

図 4: **AI エンジン アレイ**



柔軟な専用インターコネクトとの組み合わせにより、AI エンジン アレイは確定的な性能、低レイテンシ、および高帯域幅を実現します。モジュラー型のスカラー アーキテクチャのため、アレイにタイルを追加して演算性能を高めることができます。

カスケード ストリームは、最下行から最上行へ向かって水平方向に移動します。ある行の端までカスケード ストリームが到達すると、その上の行のタイルの入力に接続されます。したがって、カスケード ストリームの移動方向は、1 行ごとに反転します (ある行で左から右へ移動したら、その上の行では右から左へ移動)。最上行の端まで到達し、それ以上接続がなくなると、カスケード ストリームの移動は終了します。このように移動方向が変化するため、タイル内の AI エンジンとメモリ モジュールの位置関係は 1 行ごとに反転します。

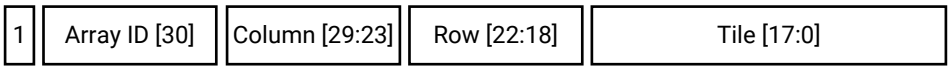
メモリ マップド AXI4 インターコネクト

各 AI エンジン タイルには、メモリ マップド AXI4 インターコネクトが 1 つあります。外部ブロックは、このインターコネクトを介して AI エンジン タイル内のレジスタまたはメモリに対して書き込みまたは読み出しを実行します。AI エンジン アレイ内のメモリ マップド AXI4 インターコネクトは、ネットワーク オン チップ (NoC) に接続できる任意の AXI4 マスターによって AI エンジン アレイ外部から駆動できます。AI エンジン タイルにあるすべての内部リソース (メモリを含む)、および AI エンジンと AI エンジン メモリ モジュールのすべてのレジスタは、メモリ マップド AXI4 インターフェイスにマップされます。

各 AI エンジン タイルにはメモリ マップド AXI4 スイッチが 1 個あり、下方向からのすべてのメモリ マップド AXI4 アクセスをこのスイッチが受け取ります。アドレスがこのタイルに対するものである場合、アクセスが発生します。それ以外の場合は、上方向にある次のタイルへアクセスが渡されます。

次の図に、AI エンジン タイルにおけるメモリ マップド AXI4 のアドレス指定方式を示します。下位 18 ビットはタイル アドレス (範囲 `0x00000` ～ `0x3FFFF`) を表し、次の 5 ビットで行位置、次の 7 ビットで列位置を表します。

図 5: **AI エンジンのメモリ マップド AXI4 インターフェイス アドレス**



X22324-022119

AI エンジンの内部メモリ マップド AXI4 インターコネクトはフル機能のメモリ マップド AXI4 プロトコルのサブセットで、次の制限事項があります。

- 書き込みアドレス前の書き込みデータは禁止
- 書き込みデータに対する `WSTRB` 信号は 1 つのみ
- 1 ～ 4 つの 32 ビット ワードのバーストのみ
- 32 ビット固定サイズ

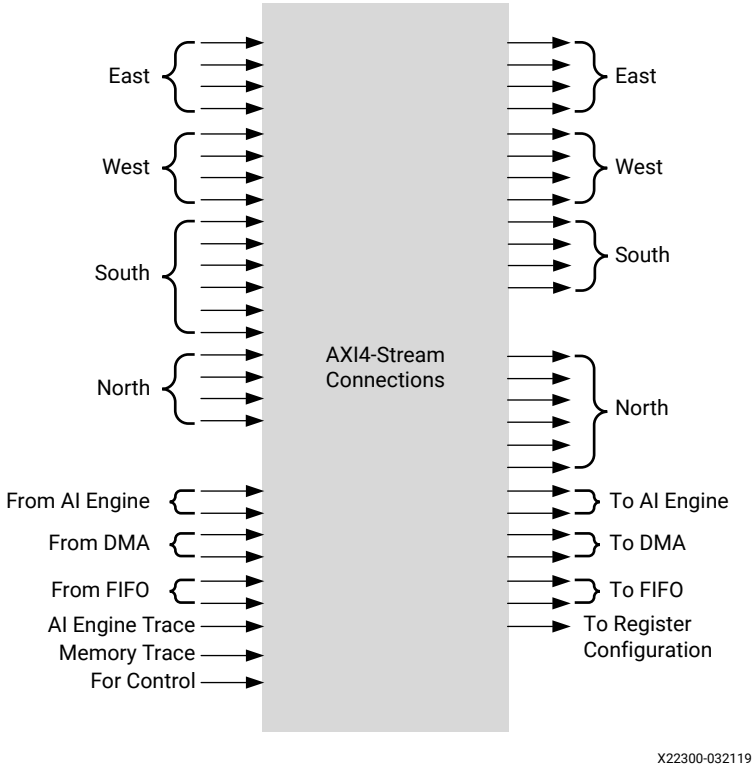
AXI4-Stream インターコネクト

各 AI エンジン タイルには 1 つの AXI4-Stream インターコネクトがあります。ストリーム スイッチとも呼ばれるこのインターコネクトは完全にプログラム可能な 32 ビット AXI4-Stream クロスバーで、メモリ マップド AXI4 インターコネクトを利用して静的に構成します。このインターコネクトはバック プレッシャーを実行し、AXI4-Stream の帯域幅を完全に利用できます。次に、AXI4-Stream スイッチの概略ブロック図を示します。このスイッチには、マスター ポート (スイッチからデータを出力) とスレーブ ポート (スイッチへデータを入力) があります。次の図に、AXI4-Stream インターコネクトを示します。AXI4-Stream インターコネクトの構築ブロックは、次のとおりです。

- ポート ハンドラー
- FIFO

- アービタ
- ストリーム スイッチ コンフィギュレーション レジスタ

図 6: **AXI4-Stream スイッチの概略ブロック図**



各ポートには入出力ストリームの経路を選択するポート ハンドラーがあります。各スイッチには、チェーン接続可能な 2 つの FIFO バッファー (深さ 16 段、幅 32 ビット データ + 1 ビット TLAST) があり、これを使用してストリームにバッファー機能を追加できます。各スイッチには、パケット交換用のプログラマブル アービタが 6 つあります。

各ストリーム ポートは、コンフィギュレーション レジスタのパケット交換ビットを使用して回路交換またはパケット交換ストリームのいずれかに構成できます。パケット交換ストリームは、ほかの論理ストリームとポート (および物理ワイヤ) を共有できます。ほかのパケット交換ストリームとの間でリソース競合の可能性があるため、レイテンシは確定的ではありません。回路交換ストリームでは、ワード送信のレイテンシは確定的です。帯域幅が制限されている場合は、内蔵のバック プレッシャー機能のために性能が低下します。

次の表に、AXI4-Stream タイル インターコネクトの帯域幅 (-1L スピード グレード デバイスの場合) をまとめます。

表 2: **AI エンジン AXI4-Stream タイル インターコネクトの帯域幅**

接続の種類	接続の数	データ幅 (ビット)	クロック ドメイン	接続あたりの帯域幅 (GB/s)	全帯域幅 (GB/s)
下側から上側へ	6	32	AI エンジン (1GHz)	4	24
上側から下側へ	4	32	AI エンジン (1GHz)	4	16
右側から左側へ	4	32	AI エンジン (1GHz)	4	16
左側から右側へ	4	32	AI エンジン (1GHz)	4	16

AI エンジン タイルのプログラム メモリ

AI エンジンには、VLIW 命令の格納に使用できる 16KB のローカル プログラム メモリがあります。プログラム メモリには、次の 2 つのインターフェイスがあります:

- メモリ マップド AXI4 インターフェイス
- AI エンジン インターフェイス

外部マスターは、メモリ マップド AXI4 インターフェイスを介してプログラム メモリに対して読み出しまたは書き込みを実行できます。AI エンジンは 128 ビット幅のインターフェイスを利用してプログラム メモリから命令をフェッチします。AI エンジンからプログラム メモリへは読み出しのみ可能で、書き込みはできません。メモリ マップド AXI4 と AI エンジンから同時にプログラム メモリにアクセスするには、プログラム メモリを複数のバンクに分割し、メモリの相互排他的部分にアクセスします。同じバンクに対して複数のアクセスがある場合は、アクセス間の競合を回避し、優先順位を割り当てるために、アービトレーション ロジックが必要になります。

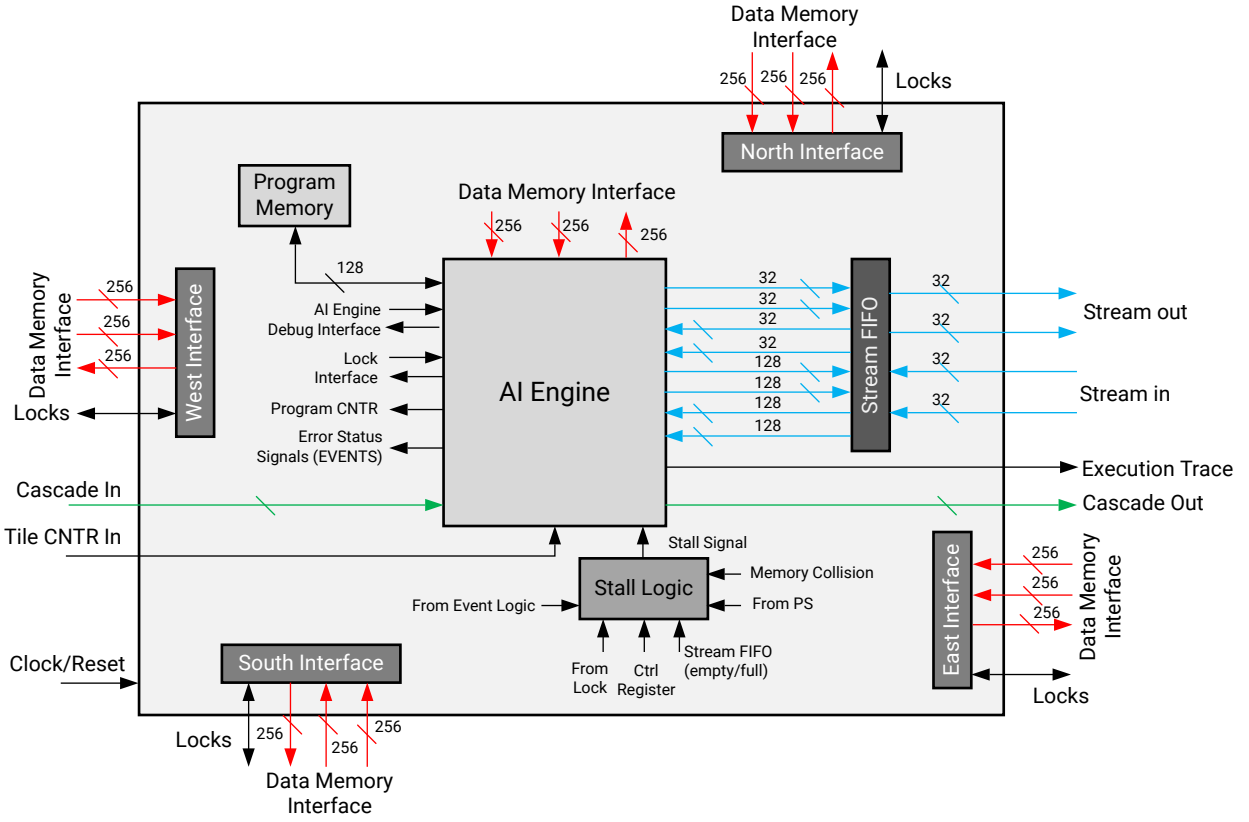
AI エンジンのインターフェイス

AI エンジンには複数のインターフェイスがあります。次に、これらインターフェイスを含むブロック図を示します。

- データ メモリ インターフェイス: AI エンジンは、4 方向すべてのデータ メモリ モジュールにアクセスできます。これらは、1 つの連続したメモリとしてアクセスされます。AI エンジンには 2 つの 256 ビット幅ロード ユニットと 1 つの 256 ビット幅ストア ユニットがあります。AI エンジンから見ると、ロード (2) とストア (1) のスループットはそれぞれ 1 サイクルあたり 256 ビットです。
- プログラム メモリ インターフェイス: AI エンジンは、この 128 ビット幅のインターフェイスを介してプログラム メモリにアクセスします。1 クロック サイクルで 1 命令をフェッチできます。
- ダイレクト AXI4-Stream インターフェイス: AI エンジンには 2 つの 32 ビット入力 AXI4-Stream インターフェイスと 2 つの 32 ビット出力 AXI4-Stream インターフェイスがあります。各ストリームは入力側と出力側の両方で FIFO に接続されており、AI エンジンは 4 サイクルあたり 4 ワード (128 ビット) または 1 サイクルあたり 1 ワード (32 ビット) でストリームにアクセスできます。
- カスケード ストリーム インターフェイス: 複数のカスケード ストリームを使用してチェーンを構成することにより、AI エンジンの 384 ビット アキュムレータ データを別の AI エンジンへ転送できます。入力ストリームと出力ストリームの両方に 384 ビット幅の 2 段 FIFO があり、AI エンジン間で最大 4 つの値をこの FIFO に格納できます。
- デバッグ インターフェイス: このインターフェイスは、メモリ マップド AXI4 インターフェイスを介して AI エンジンのすべてのレジスタに対して読み出しまたは書き込みを実行できます。
- ハードウェア同期 (ロック) インターフェイス: 2 つの AI エンジン間、または AI エンジンと DMA 間の同期に使用します。AI エンジンは、4 方向すべてのロック モジュールにアクセスできます。
- ストール処理: AI エンジンは、各種ソースからさまざまな理由でストールできます。たとえば、外部メモリ マップド AXI4 マスター (PS など)、ロック モジュール、エンプティまたはフル AXI4-Stream インターフェイス、データ メモリ競合、およびイベント ユニットからのイベント アクションなどによるストールが可能です。
- AI エンジン イベント インターフェイス: この 16 ビット幅のインターフェイスを使用して、さまざまなイベントを設定できます。
- タイル タイマー: タイル内の 64 ビット タイマー値を読み出すための入力インターフェイスです。

- 実行トレース インターフェイス: AI エンジンで生成したパケット ベースの実行トレースを AXI4-Stream 経由で送信できる 32 ビット幅のインターフェイスです。

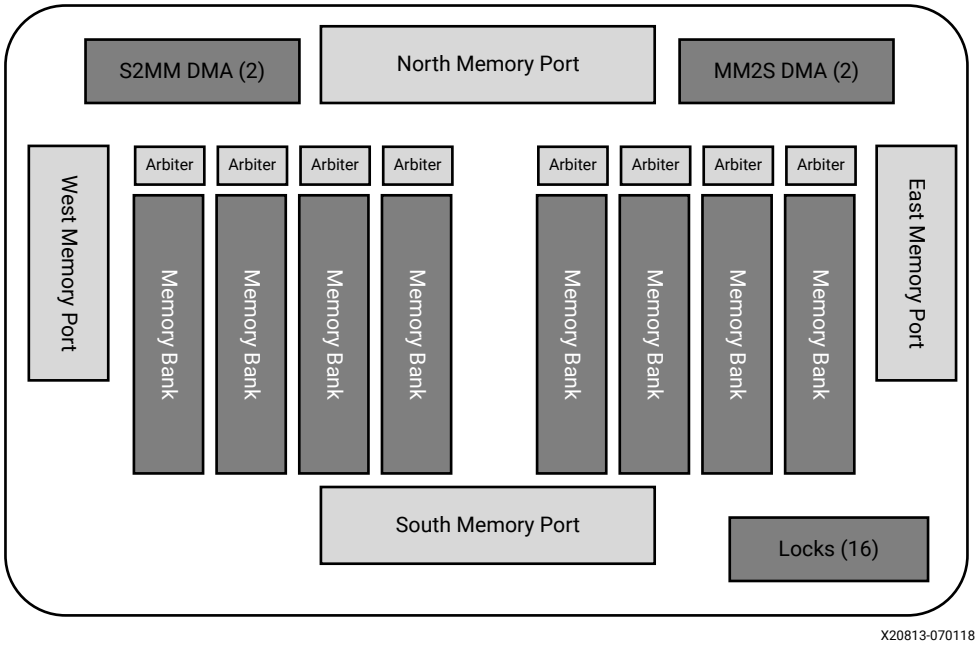
図 7: **AI エンジンのインターフェイス**



AI エンジンのメモリ モジュール

AI エンジンのメモリ モジュールは、8 つのメモリ バンク、2 つの S2MM 入力 (Stream to Memory Map) DMA、2 つの MM2S 出力 (Memory-Map to Stream) DMA、および 1 つのハードウェア同期 (ロック) モジュールで構成されます (下図参照)。上下左右の 4 方向それぞれに、偶数番号と奇数番号ごとに独立したポート、3 つのアドレス ジェネレーター、2 つのロード、1 つのストアがあります。

図 8: **AI エンジンのメモリ モジュール**



- メモリ バンク: AI エンジンのメモリ モジュールには 8 つのメモリ バンクがあります。1 メモリ バンクは 256 ワード x 128 ビットのシングル ポート メモリです。各メモリ バンクには、各 32 ビット ワードに対するライト イネーブルがあります。バンク [0-1] には ECC 保護があり、バンク [2-7] にはパリティ チェックがあります。バンク [0] はメモリ モジュールのアドレス 0 から開始します。ECC 保護は、32 ビット ワードに対して 1 ビットのエラーを検出および訂正し、2 ビットのエラーを検出します。

- メモリ アービトレーション: 各メモリ バンクには、すべてのリクエスターに対してアービトレーションを実行する専用のアービタがあります。メモリ バンクのアービトレーションをラウンド ロビン方式で実行することにより、リクエスターがアイドルになるのを防ぎます。1 クロック サイクルで 1 つの新しい要求が処理されます。同じサイクルで同じメモリ バンクに対して複数の要求がある場合、メモリへのアクセスを許可されるのは 1 サイクルあたり 1 つの要求のみです。それ以外のリクエスターは 1 サイクルの間ストールし、次のサイクルでハードウェアがメモリ要求を再送します。

- タイル DMA コントローラー: タイル DMA には、AI エンジン タイル内のストリーム スイッチに対して 2 つの受信ストリームと 2 つの送信ストリームがあります。タイル DMA コントローラーは、ストリーム データをメモリへ格納する S2MM (32 ビット データ) とメモリの内容をストリームに書き込む MM2S (32 ビット データ) の 2 つのモジュールで構成されます。各 DMA 転送は DMA バッファー ディスクリプターで 定義し、DMA コントローラーは 16 個のバッファー ディスクリプターにアクセスします。これらのバッファー ディスクリプターには、コンフィギュレーション用のメモリ マップド AXI4 インターコネクトを使用してアクセスすることもできます。各バッファー ディスクリプターには、DMA 転送に必要なすべての情報に加え、現在の DMA 転送が完了した後に DMA

コントローラーが実行する次の DMA 転送へのポインターも含めることができます。DMA コントローラーは、AI エンジンと DMA または AI エンジン アレイ外部の任意のメモリ マップド AXI4 マスターと DMA 間の同期メカニズムである 16 個のロックにもアクセスできます。各バッファー ディスクリプターにはロックを関連付けることができます。これは、メモリ マップド AXI4 インターコネクトを使用してバッファー ディスクリプターで 設定します。

- ロック モジュール: AI エンジンのメモリ モジュールには、AI エンジン、タイル DMA、および外部メモリ マップド AXI4 インターフェイス マスター (PS など) の間で同期をとるためのロック モジュールが含まれます。ハードウェア ロックは 16 あり、AI エンジンの各メモリ モジュール ロック ユニットに対して 1 ビットのデータ値を設定します。各ロックには、同時要求を管理するためのアービタがあります。ロック モジュールは、4 方向すべての AI エンジン、ローカル DMA コントローラー、およびメモリ マップド AXI4 からのロック要求を処理します。

AI エンジンのデータ移動アーキテクチャ

このセクションでは、AI エンジン アレイ内、および AI エンジン タイルとプログラマブル ロジック (PL) 間のデータ通信について、いくつかの例を挙げて説明します。

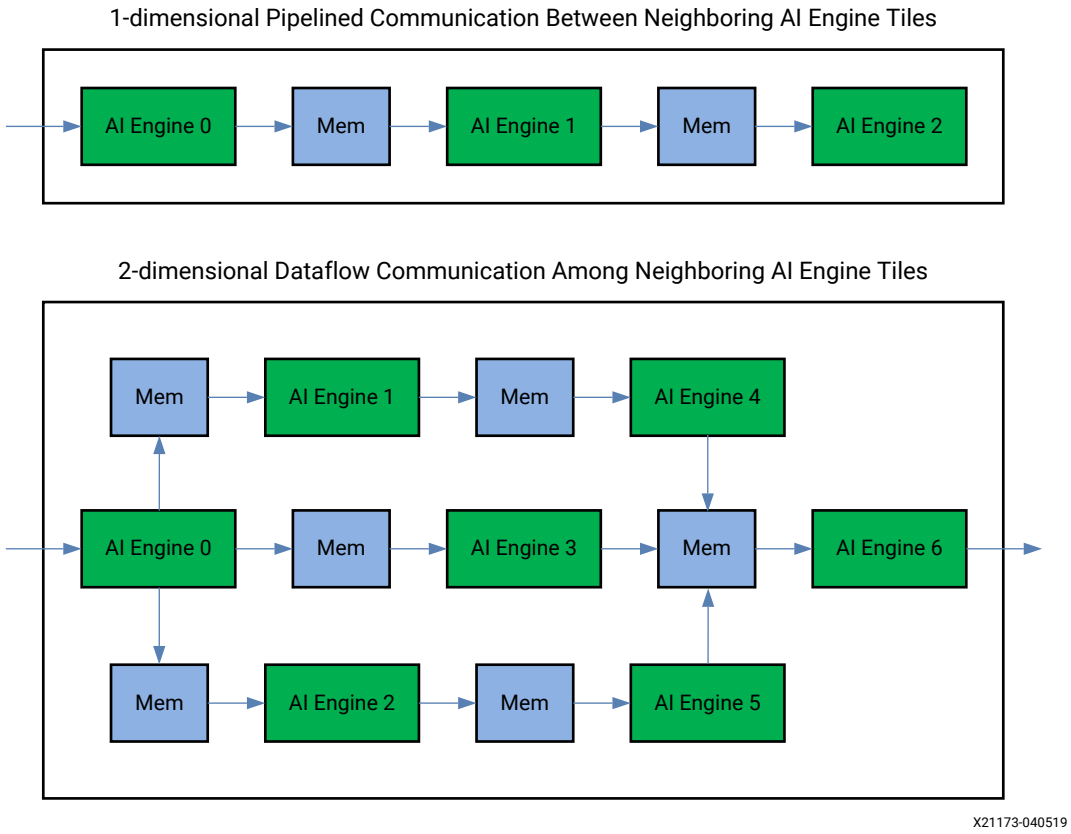
共有メモリを使用した AI エンジンから AI エンジンへのデータ通信

共有メモリを使用した AI エンジンから AI エンジンへのデータ通信

1 個の AI エンジンに複数のカーネルが含まれる場合、2 つの連続するカーネルどうしは共有メモリ内の共通バッファーを使用して通信できます。隣接している別々の AI エンジンに含まれるカーネルどうしは、共有メモリ モジュールを使用して通信します。次の図に示すように、データ移動はシンプルなパイプラインで処理することも、複数の並列パイプ ステージを使用して処理することもできます。2 つの AI エンジン間の通信は、別々のメモリ バンク上でピンポン バッファー (図では省略) を使用してアクセス競合を防ぎます。この場合、同期にはロックを使用します。この通信方法では、DMA と AXI4-Stream インターコネクトは必要ありません。

次の図に、AI エンジン タイル間のデータ通信を示します。これらは、AI エンジン タイルおよび共有メモリ モジュールの論理表現です。

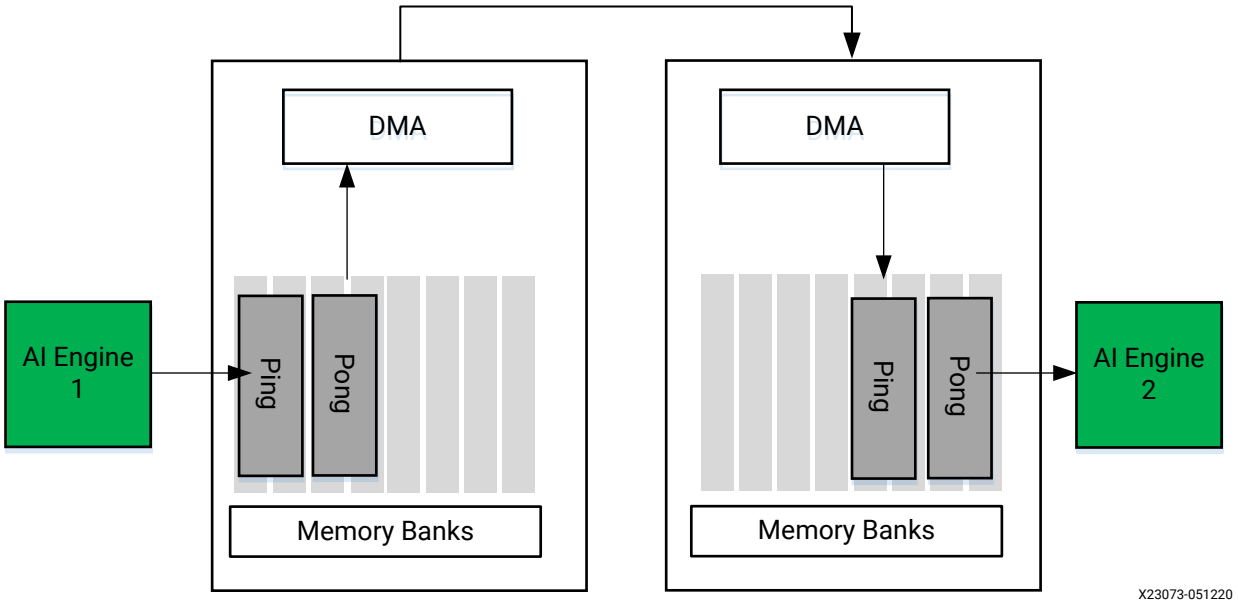
図 9: **共有メモリを使用した AI エンジンから AI エンジンへのデータ通信**



メモリと DMA を使用した AI エンジン タイルから AI エンジン タイルへのデータ通信

前のセクションで説明した通信は、1 個の AI エンジン タイル内、または隣接する 2 つの AI エンジン タイル間のものです。隣接しない AI エンジン タイルの場合、次の図に示すように各 AI エンジン タイルに関連付けられたメモリ モジュール内で DMA を使用して同様の通信を確立できます。各メモリ モジュール内のピンポン バッファーは、共有メモリを使用した AI エンジンから AI エンジンへのデータ通信 で説明した方法と同様に、ロックを使用して同期をとります。ただし通信レイテンシが長くなり、必要なメモリ リソースが増大するという大きな違いがあります。

図 10: **隣接しない 2 つの AI エンジン タイル間のデータ通信**

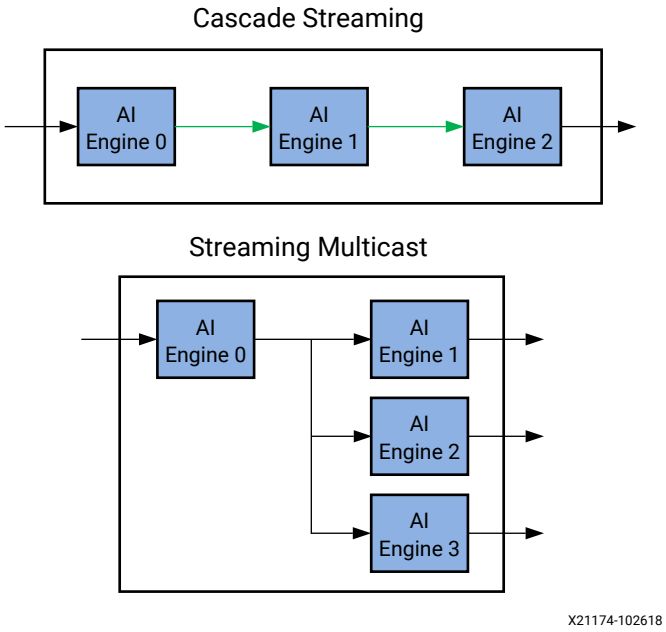


AXI4-Stream インターコネクトを使用した AI エンジン タイルから AI エンジン タイルへのデータ通信

AI エンジンどうしは、DMA やメモリを使用せず AXI4-Stream インターコネクトを介して直接通信できます。次の図に示すように、AI エンジンから別の AI エンジンへは、ストリーミング インターフェイス経由で逐次的にデータを送信することも、マルチキャスト通信アプローチを使用して同じ情報を任意の数の AI エンジン タイルに送信することもできます。ストリームは、上下方向および左右方向に転送できます。いずれのストリーミングの場合も、内蔵のハンドシェイクおよびバック プレッシャー メカニズムを利用します。

注記: マルチキャスト通信アプローチの場合、準備が完了していないレシーバーが 1 つでもあるとブロードキャスト全体が停止し、すべてのレシーバーの準備が完了してから再開されます。

図 11: **AXI4-Stream インターコネクトを使用した AI エンジンから AI エンジンへのデータ通信**

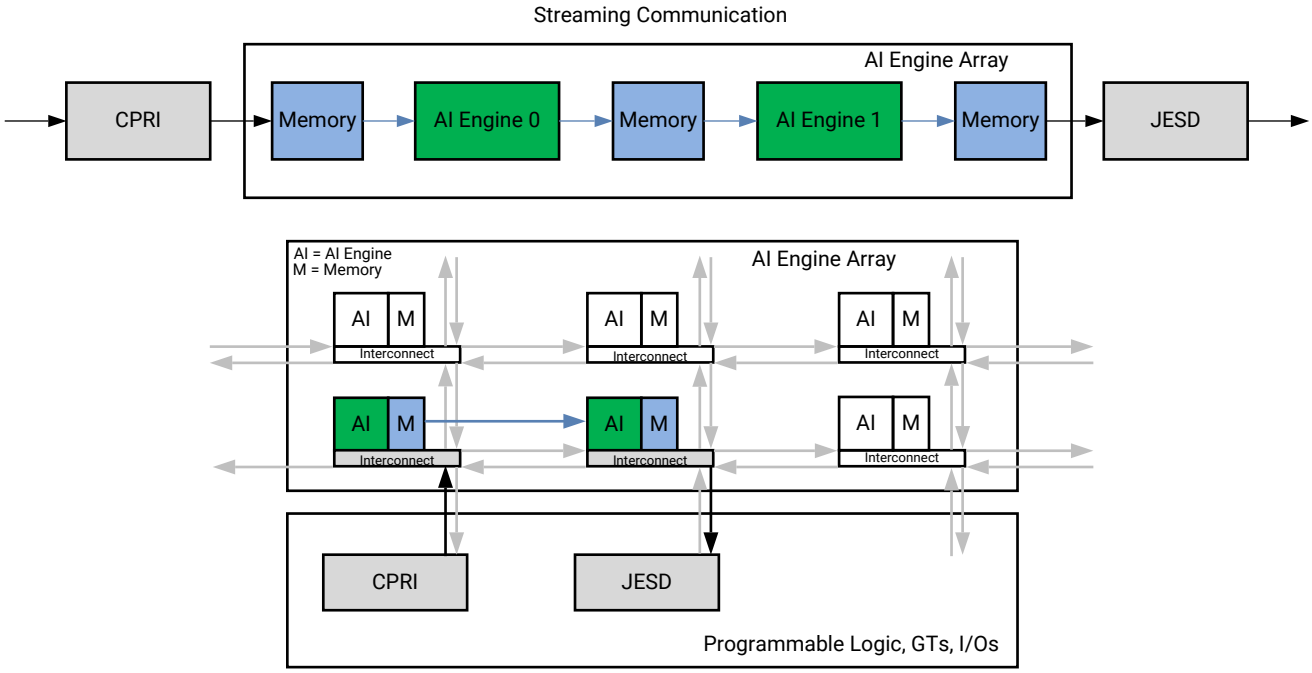


共有メモリを使用した AI エンジンから PL へのデータ通信

一般的には、PL ブロックはストリーム インターフェイス経由のデータを使用します。その後、PL ブロックはデータ ストリームを生成してアレイ インターフェイスへ転送します。アレイ インターフェイスには、PL ストリームを受信して AI エンジン ストリームに変換する FIFO があります。変換された AI エンジン ストリームは、AI エンジンのデスティネーション ファンクションへ転送されます。通信がブロック ベースかストリーム ベースかによって、DMA とピンポン バッファーの使用方法が異なります。

次の図に、PL 内の CPRI™ (Common Public Radio Interface) と JESD® 間のデータ通信の例 (ユース ケース) を示します。AI エンジンと PL は、AI エンジン タイル内の DMA を使用して通信できます。DMA は、このストリームをそれを使用する AI エンジンに隣接するメモリ ブロックへ移動します。この図の上側は論理表現で、下側は物理表現です。

図 12: **共有メモリを使用した AI エンジンから PL へのデータ通信**



X21175-102618

AI エンジンのデバッグ

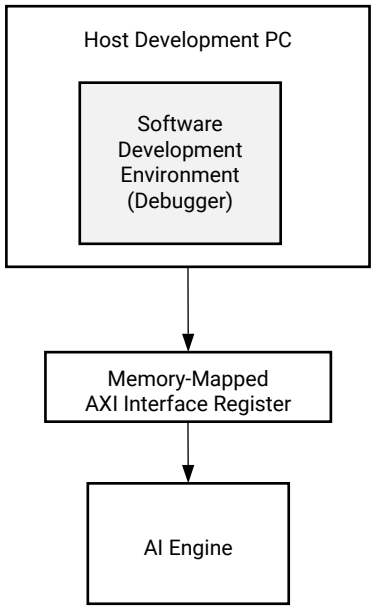
AI エンジンのデバッグには、メモリ マップド AXI4 インターフェイスを使用します。AI エンジン アレイの主要なコンポーネントはすべてメモリにマップされています。

- プログラム メモリ
- データ レジスタ
- AI エンジン レジスタ
- DMA レジスタ
- ロック モジュール レジスタ
- ストリーム スイッチ レジスタ
- AI エンジン ブレークポイント レジスタ

- イベント/パフォーマンス カウンター レジスタ

これらのメモリ マップド レジスタは、メモリ マップド AXI4 インターフェイス要求を生成可能な任意のマスター (PS、PL、および PMC) から読み出し/書き込みが可能です。これらの要求は NoC を経由して AI エンジン アレイ インターフェイスへ送信され、そこからアレイ内のターゲット タイルに転送されます。次の図に、デバッガーを統合したソフトウェア開発環境をホスト開発システム上で実行した代表的なデバッグ環境を示します。

図 13: **AI エンジン デバッグ インターフェイスの概要**



デバッガーは、JTAG 接続またはザイリンクス高速デバッグ ポート (HSDP) 接続を使用して AI エンジンを内蔵する Versal デバイスのプラットフォーム管理コントローラー (PMC) に接続します。

AI エンジンのトレースとプロファイリング

AI エンジン タイルはトレースおよびプロファイリング用の機能を備えています。また、トレースおよびプロファイリング ハードウェアを制御するコンフィギュレーション レジスタも備えています。

トレース

各 AI エンジン タイルからは 2 つのトレース ストリームが出力されます。1 つは AI エンジンから、もう 1 つはメモリ モジュールから出力されます。いずれのストリームも、タイルのストリーム スイッチに接続されます。AI エンジン タイルの各 AI エンジン モジュールとメモリ モジュール、および AI エンジン PL インターフェイス タイル (アレイ インターフェイス タイルの種類 参照) の AI エンジン プログラマブル ロジック (PL) モジュールには、トレース ユニットがあります。これらのトレース ユニットは、次のモードで動作できます。

- AI エンジン モード
  - イベント タイム
  - イベント PC

    - 実行トレース
- AI エンジン メモリ モジュール モード
    - イベント タイム
- AI エンジン PL モジュール モード
    - イベント タイム

トレースは、トレース ユニットから AXI4-Stream 経由で AI エンジン パケット交換ストリーム パケットとして出力されます。パケット サイズは 8x32 ビットで、1 ワードのヘッダーと 7 ワードのデータを含みます。アレイの AXI4-Stream スイッチは、パケット ヘッダーに含まれる情報を使用して、転送可能な任意の AI エンジン デスティネーションへパケットを転送します。これには、AI エンジン タイル DMA 経由で AI エンジン ローカル データ メモリへ、AI エンジン アレイ インターフェイス DMA 経由で 外部 DDR メモリへ、AI エンジンから PL への AXI4-Stream 経由でブロック RAM または URAM へ、などの転送が含まれます。

イベント タイム モードは、独立した番号付きの最大 8 つのイベントをサイクルごとに追跡します。追跡したイベントのステートの変化を記録するために、トレース フレームが作成されます。これらのフレームが出力バッファー内で収集され、AI エンジン パケット交換ストリーム パケットが作成されます。複数のフレームを 1 つの 32 ビット ストリーム ワードにパックできますが、32 ビット境界をまたぐことはできません (32 ビット アライメントのためにフィラー フレームが使用される)。

イベント PC モードでは、監視対象の 8 つのイベントのうち 1 つ以上がアサートされたサイクルで、トレース フレームが生成されます。トレース フレームは、AI エンジンの現在のプログラム カウンター (PC) 値、および監視対象の 8 つのイベントの現在の値を記録します。これらのフレームが出力バッファー内で収集され、AI エンジン パケット交換ストリーム パケットが作成されます。

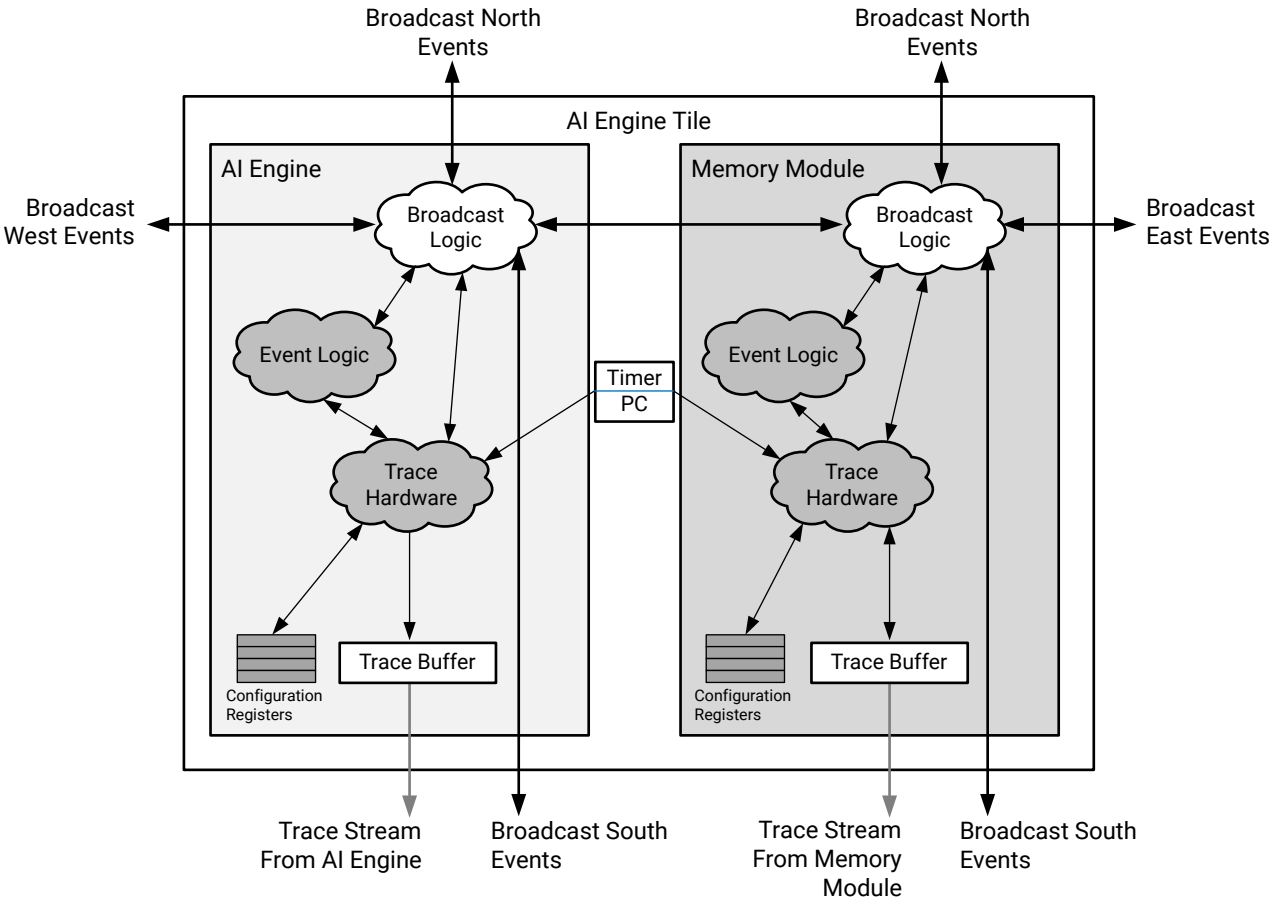
AI エンジンのトレース ユニットは、実行トレース モードで動作できます。トレース ユニットは、オフライン デバッガーがプログラム実行フローを再構築するのに必要な最小限の情報を AXI4-Stream 経由でリアルタイムに送信します。これは、オフライン デバッガーが ELF にアクセスできることを前提条件としています。この情報には、次のものが含まれます。

- 直接分岐 (条件分岐および無条件分岐)
- すべての間接分岐
- ゼロ オーバーヘッド ループ LC

AI エンジンはパケット ベースの実行トレースを生成します。これは、32 ビット幅の実行トレース インターフェイスを介して送信できます。次の図に、AI エンジン タイルのトレース ハードウェアの論理構成を示します。タイルから出力される 2 つのトレース ストリームは、内部でイベント ロジック、コンフィギュレーション レジスタ、ブロードキャスト イベント、およびトレース バッファーに接続されます。

注記: これら 2 つのモジュールの動作モードの違いは、図には表示していません。

図 14: **AI エンジン トレース ハードウェアの論理構成**

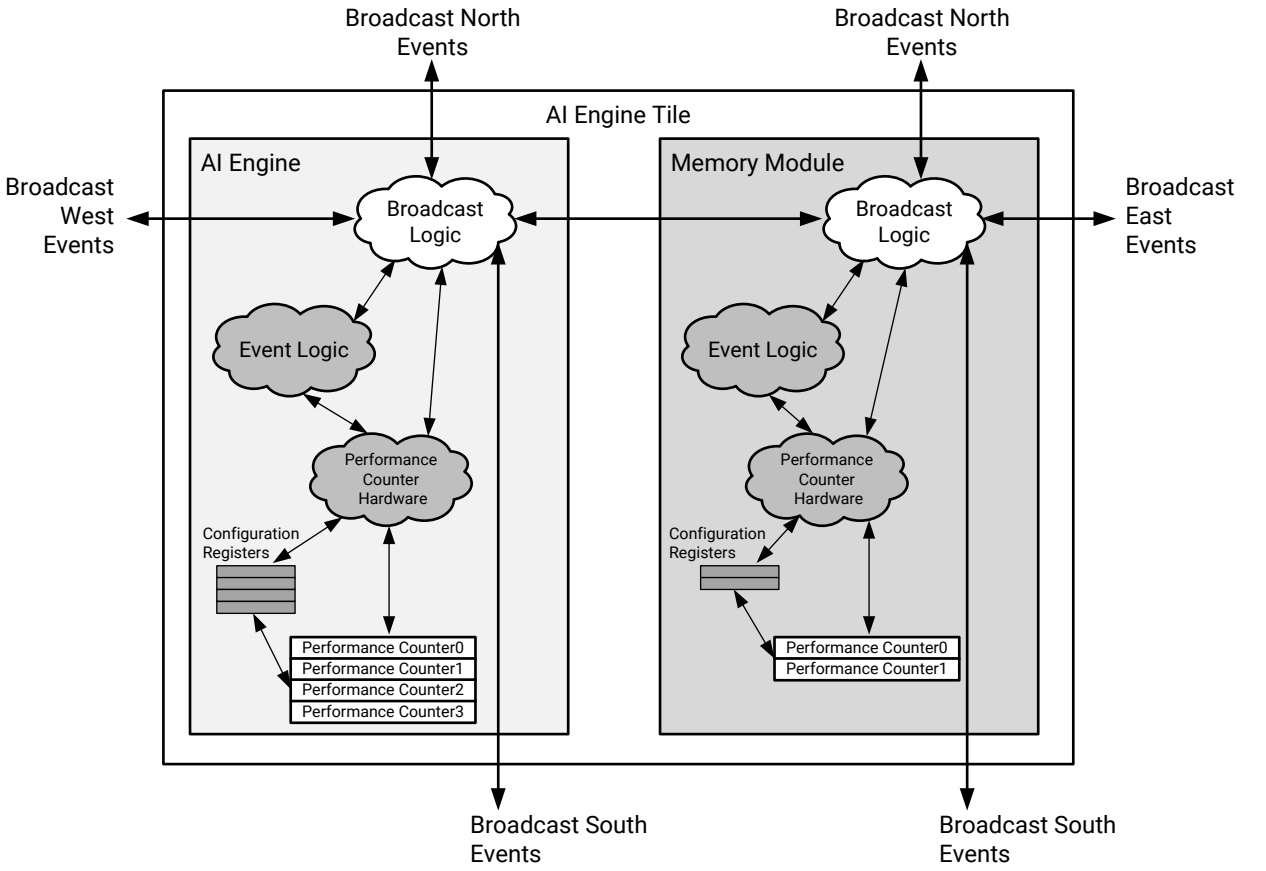


イベント トレース用のトレース ストリームを制御するために、トレースを開始および停止する 32 ビット `trace_control0/1` レジスタがあります。また、トレースに追加される内部イベント番号をプログラムするための `trace_event0/1` レジスタもあります。各レジスタの詳細は、『Versal ACAP AI エンジン レジスタ リファレンス』 (AM015) を参照してください。

プロファイリング (パフォーマンス カウンター)

AI エンジン アレイは、プロファイリングに使用できるパフォーマンス カウンターを備えています。AI エンジンには 4 つのパフォーマンス カウンターがあり、任意の内部イベントをカウントするように設定できます。このカウンターは、イベントの発生回数をカウントすることも、2 つの定義済みイベント間のクロック サイクル数をカウントすることもできます。PL および NoC アレイ インターフェイス タイル内のメモリ モジュールと PL モジュールにはそれぞれ 2 つのパフォーマンス カウンターがあり、これらも同様の機能を実行するように設定できます。次の図に、AI エンジン タイル内のプロファイリング ハードウェアの概略的な論理構成を示します。パフォーマンス制御レジスタとパフォーマンス カウンター レジスタについては、『Versal ACAP AI エンジン レジスタ リファレンス』 (AM015) で説明されています。

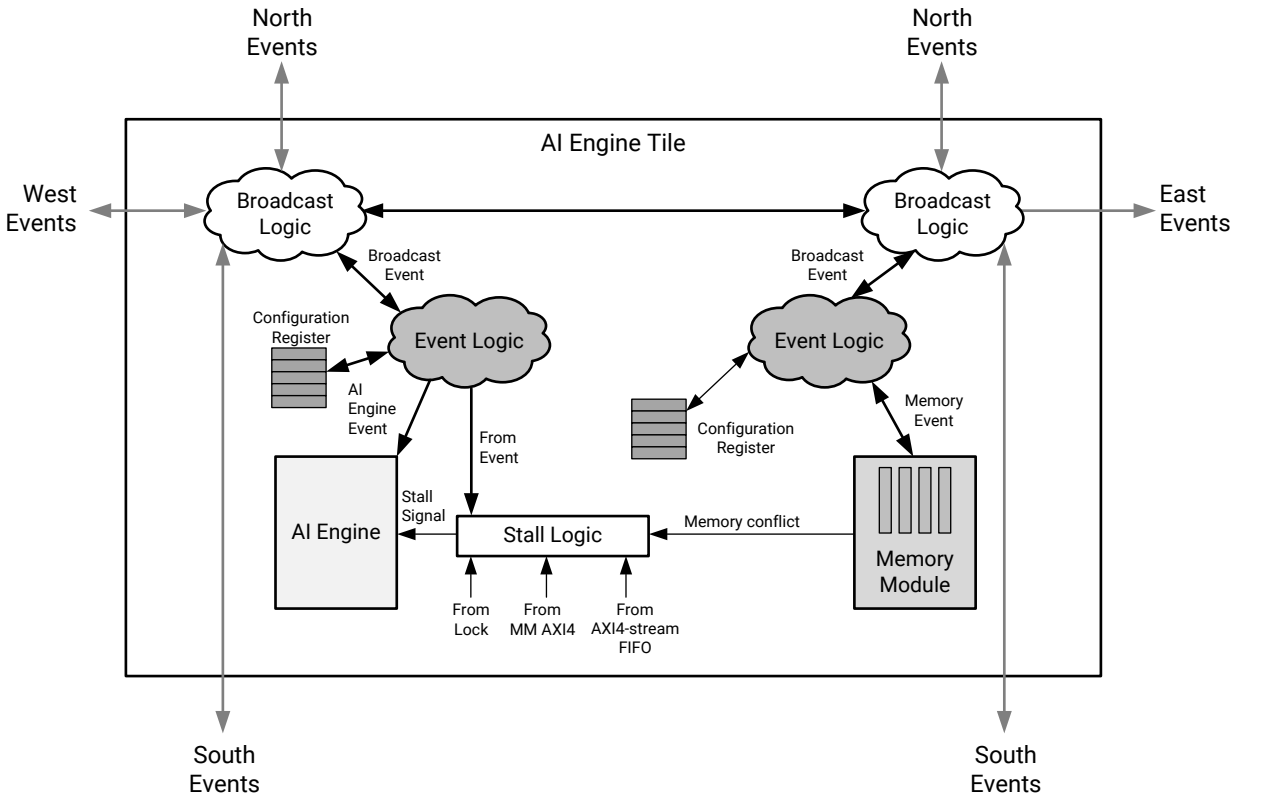
図 15: **AI エンジン プロファイリングの論理構成**



AI エンジンのイベント

AI エンジンとメモリ モジュールには、それぞれイベント ロジック ユニットがあります。各ユニットには、いくつかのローカル イベントが定義されています。次の図に、AI エンジン タイル内のイベントの概略的な論理構成を示します。イベント ロジックは、メモリ マップド AXI4 インターフェイスを介してプログラムできる一連のアクション レジスタを使用して設定する必要があります。AI エンジンおよびメモリ モジュールのみと連携動作するイベント ロジック ハードウェアがあります。特定イベントの発生時にタスクを実行するように、イベント アクションを設定できます。イベント ハードウェアには、コンフィギュレーション レジスタ セットが 2 つあります。また、イベント信号を隣接するモジュールに送信するブロードキャスト ロジックも 2 つあります。

図 16: **AI エンジン タイル内のイベント**



イベント アクション

イベント自体にアクションはありませんが、イベントを使用してアクションを作成できます。イベント ブロードキャストとイベント トレースを設定して、イベントを監視できます。イベント アクションの例には、次のものがあります。

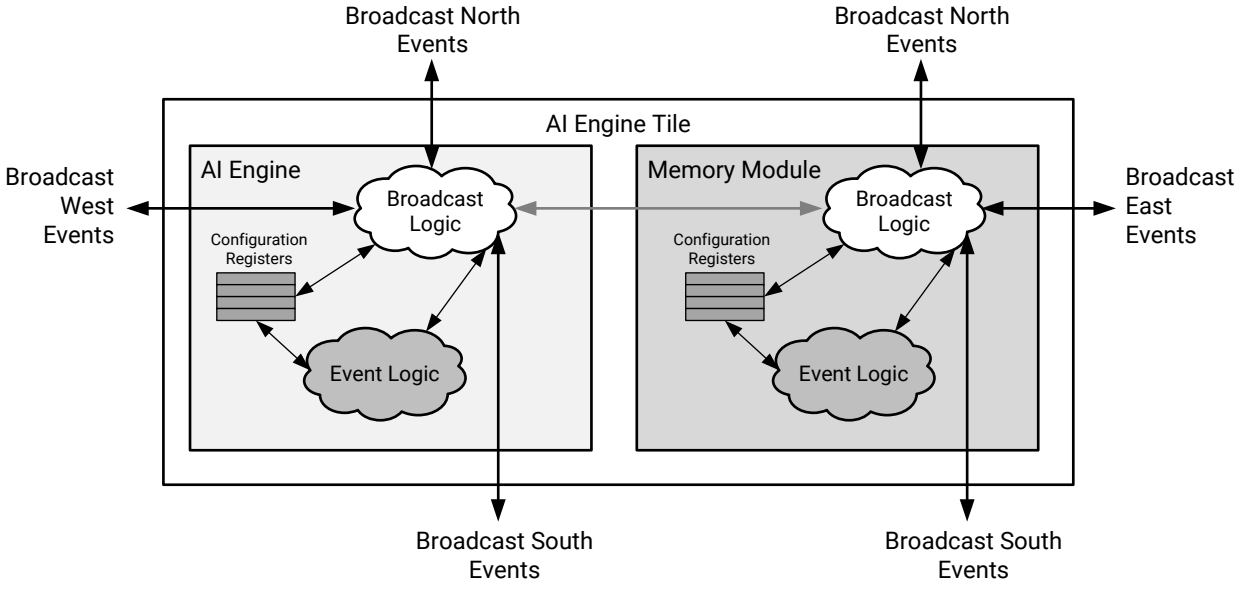
- AI エンジンのイネーブル、ディスエーブル、またはリセット
- AI エンジンのデバッグ停止、再開、またはシングル ステップ実行
- AI エンジンのエラー停止
- タイマーの再同期
- パフォーマンス カウンターの開始および停止
- トレース ストリームの開始および停止
- ブロードキャスト イベントの生成
- コンボ イベントの駆動
- ECC スクラビング イベント

これらのイベント アクションにはいずれも関連するレジスタがあり、そこで設定した 7 ビットのイベント番号を使用して、特定のイベントでアクションをトリガーするように設定します。AI エンジンおよびメモリ モジュールのすべてのイベント アクション コンフィギュレーション レジスタのリストは、『Versal ACAP AI エンジン レジスタ リファレンス』 (AM015) に記載されています。

イベント ブロードキャスト

ブロードキャスト イベントは、設定済みのイベントがアサートされたときにトリガーされるため、イベントでもありイベント アクションでもあります。次に、AI エンジン タイル内のブロードキャスト ロジックの論理構成を示します。AI エンジンおよびメモリ モジュール内にあるブロードキャスト ロジック内のユニットは、4 方向すべてから入力を受信し、4 方向すべてに信号を送信します。ブロードキャスト ロジックはイベント ロジックに接続され、イベント ロジックがすべてのイベントを生成します。送信されるイベントを選択するコンフィギュレーション レジスタと、イベントが AI エンジン タイルから送信されるのを阻止するマスク レジスタがあります。

図 17: **AI エンジン ブロードキャスト イベント**



各モジュールには、ブロードキャスト イベント信号をほかの方向にブロードキャストするように指定する内部レジスタがあります。ブロードキャスト ループを防ぐために、受信したイベント信号と内部イベントを OR 演算し、次のリストに従って、送信されるイベント信号を駆動します。

- 内部、右、上、下 → 左
- 内部、左、上、下 → 右
- 内部、下 → 上
- 内部、上 → 下

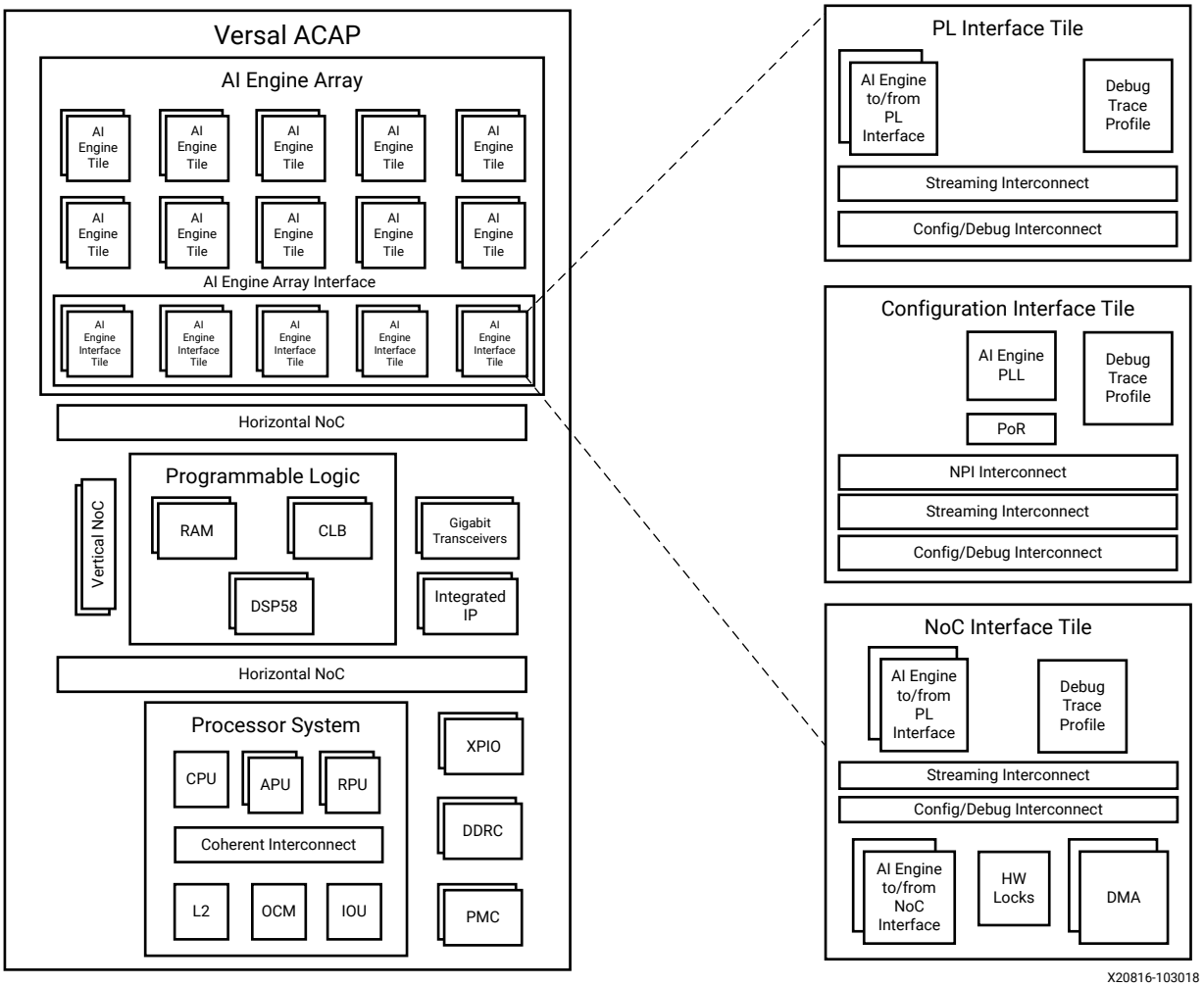
ヒント: AI エンジン モジュールの右側のブロードキャスト イベント インターフェイスは、メモリ モジュールの左側のブロードキャスト イベント インターフェイスに内部で接続され、AI エンジン タイルからは出力されません。AI エンジン モジュールには、上、下、および左方向にそれぞれ 16 のブロードキャスト イベントがあります。メモリ モジュールには、上、下、および右方向にそれぞれ 16 のブロードキャスト イベントがあります。ブロードキャスト イベントに関して、プログラマの観点からは、奇数行と偶数行のインターフェイスに違いはありません。

第 3 章

AI エンジン アレイ インターフェイスのアーキテクチャ

AI エンジンは、次の図に示すように 2 次元アレイとして構成されています。AI エンジン アレイ インターフェイスは、デバイスのその他のブロックとの接続に必要な機能を提供します。AI エンジン アレイ インターフェイスには、3 種類の AI エンジン インターフェイス タイルがあります。これらのインターフェイス タイルは、AI エンジン アレイの各列に 1 対 1 で対応します。インターフェイス タイルは 1 つの行を構成し、メモリ マップド AXI4 および AXI4-Stream データを水平方向 (左右) に移動し、AI エンジン タイルの列 (垂直) 方向にも移動します。AI エンジン インターフェイス タイルはモジュール アーキテクチャに基づいていますが、最終的な構成はデバイスごとに異なります。次の図に、AI エンジン アレイ内の AI エンジン アレイ インターフェイスの内部階層を示します。

図 18: **AI エンジン アレイ インターフェイスの階層**



このセクションでは、アレイ インターフェイス タイルの種類、および各タイルに含まれるモジュールについて説明します。

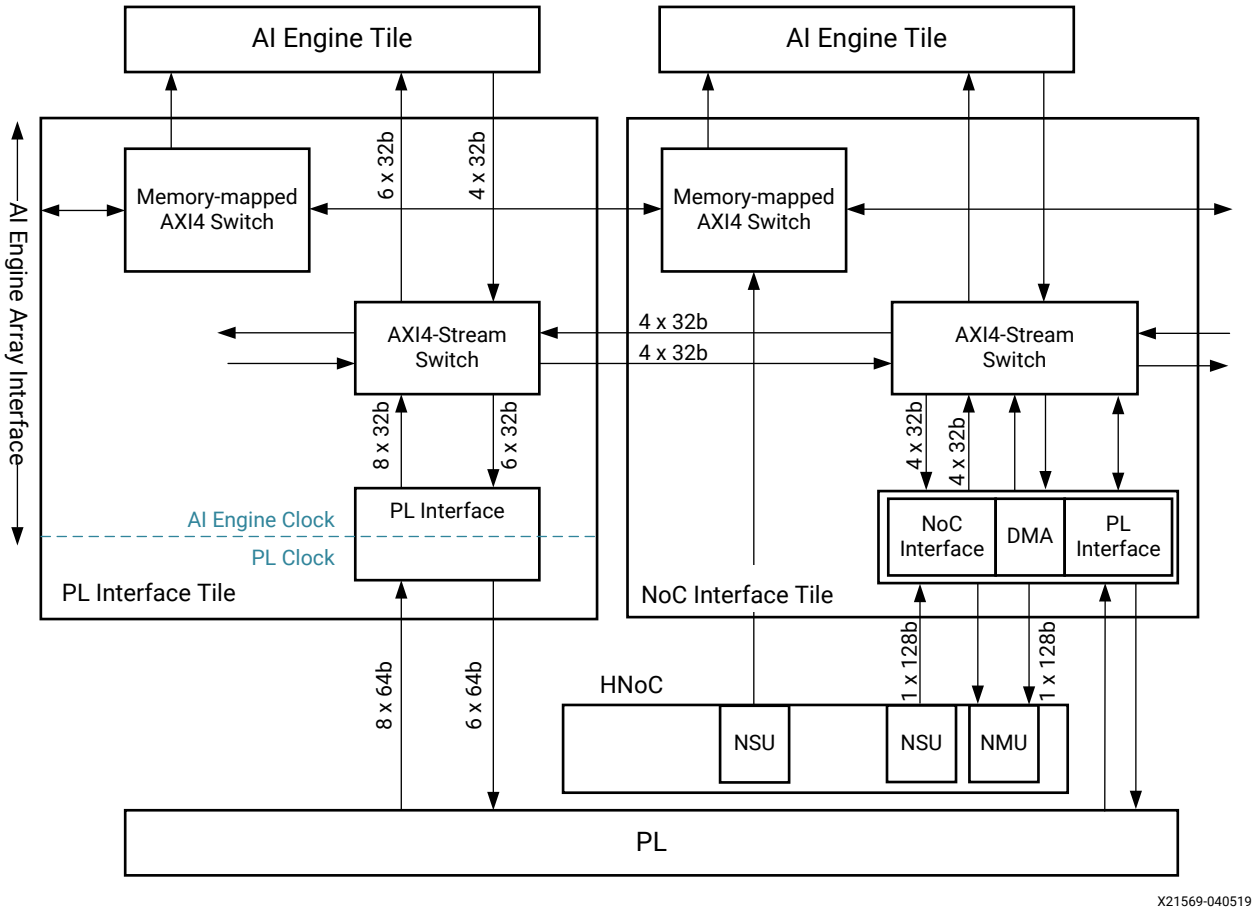
- AI エンジン PL インターフェイス タイル
  - PL モジュール
    - AXI4-Stream スイッチ
    - メモリ マップド AXI4 スイッチ
    - AI エンジンから PL へのストリーム インターフェイス
    - 制御、デバッグ、およびトレース ユニット
- AI エンジン コンフィギュレーション インターフェイス タイル (各 AI エンジン アレイに 1 インスタンス)
  - AI エンジン クロック生成用 PLL
  - パワーオン リセット (POR) ユニット
  - 割り込み生成ユニット
  - Dynamic Function eXchange (DFX) ロジック
  - NoC ペリフェラル インターコネクト (NPI) ユニット
  - AI エンジン アレイのグローバル レジスタ (PLL/クロック制御、セキュア/非セキュア動作、割り込みコントローラー、グローバル リセット制御、DFx ロジックなどのグローバル機能を制御)
- AI エンジン NoC インターフェイス タイル
  - PL モジュール: 同上
  - NMU および NSU へのインターフェイスを備える NoC モジュール
    - 双方向 NoC ストリーミング インターフェイス
    - アレイ インターフェイス DMA

AI エンジン アレイ インターフェイス

AI エンジン アレイ インターフェイスは、PL インターフェイス タイルと NoC インターフェイス タイルで構成されます。また、デバイスにコンフィギュレーション インターフェイスが 1 つあります。次の図に、アレイ インターフェイスの接続を示します。AI エンジン アレイは、この接続を使用して Versal アーキテクチャ内のほかのブロックと通信します。また、この図には PL、NoC、または AI エンジン タイルとの接続、および AXI4-Stream スイッチどうしの接続に使用する AXI4-Stream インターコネクトのストリーム数も示しています。

ヒント: PL インターフェイス タイルと NoC インターフェイス タイルの数は、個々のデバイスにより異なります。AI エンジン アレイのサイズは、『Versal アーキテクチャおよび製品データシート: 概要』 (DS950: 英語版、日本語版) を参照してください。

図 19: **AI エンジン アレイ インターフェイスのトポロジ**



注記: AI エンジンの F_{MAX} は、-1L スピード グレード デバイスの場合で 1GHz です。PL クロックは、その半分のスピードの 500MHz に設定する必要があります。また、AI エンジンと NoC のクロック間の NoC インターフェイス タイルでクロック乗せ換えがあります。

PL および NoC へのインターフェイスには、次の種類があります。

- メモリ マップド AXI4 インターフェイス: NSU から AI エンジン (スレーブ) への通信チャネル
- AXI4-Stream インターコネクトには、次の 3 種類のインターフェイスがあります。
  - PL ストリーミング インターフェイスへの双方向接続
  - アレイ インターフェイス DMA への接続 (メモリ マップド AXI4 インターフェイスを使用して NoC へのトラフィックを生成)
  - NoC ストリーミング インターフェイス (NSU および NMU) との直接接続

AI エンジン アレイ インターフェイス タイルは、次の 2 つの高性能インターフェイスを管理します。

- AI エンジン/PL 間
- AI エンジン/NoC 間

次の表に、PL、NoC、および AI エンジン タイルと接続する AI エンジン アレイ インターフェイスの帯域幅をまとめます。ここに示した帯域幅の値は、-1L スピード グレード デバイスの場合の AI エンジン 1 列分で指定されています。PL と AI エンジン インターフェイス間、および AXI4-Stream スイッチと AI エンジン タイル間では、1 列あたりの接続数が少なくなっています。これは、水平方向の配線能力を高めるために、水平方向に接続されたストリーム スイッチをサポートしているためです。これ以外のスピード グレードの各種デバイスの全帯域幅は、『Versal AI コア シリーズ データシート: DC 特性および AC スイッチ特性』 (DS957: 英語版、日本語版) を参照してください。

表 3: **AI エンジン アレイ インターフェイス/PL インターフェイス間の帯域幅**

接続の種類	接続の数	データ幅 (ビット)	クロック ドメイン	接続あたりの帯域幅 (Gb/s)	全帯域幅 (GB/s)
PL から AI エンジン アレイ インターフェイス	8	64	PL (500MHz)	4	32
AI エンジン アレイ インターフェイスから PL	6	64	PL (500MHz)	4	24
AI エンジン アレイ インターフェイスから AXI4-Stream スイッチ	8	32	AI エンジン (1GHz)	4	32
AXI4-Stream スイッチから AI エンジン アレイ インターフェイス	6	32	AI エンジン (1GHz)	4	24
AXI4-Stream スイッチ間の水平インターフェイス [1]	4	32	AI エンジン (1GHz)	4	16

注記:

1. 記載した全帯域幅は左右方向の値です。各 AXI4-Stream スイッチには、入力と出力の 2 方向の接続があります。

表 4: **AI エンジン アレイ インターフェイス/NoC インターフェイス間の帯域幅**

接続の種類	接続の数	データ幅 (ビット)	クロック ドメイン	接続あたりの帯域幅 (Gb/s)	全帯域幅 (GB/s)
AI エンジンから NoC (NoC 側)	1	128	NoC インターフェイス (1GHz)	16	16
AI エンジンから NoC (AI エンジン側)	4	32	AI エンジン (1GHz)	4	16
NoC から AI エンジン (NoC 側)	1	128	NoC インターフェイス (1GHz)	16	16
NoC から AI エンジン (AI エンジン側)	4	32	AI エンジン (1GHz)	4	16

表 5: **AI エンジン アレイ インターフェイス/AI エンジン タイル間の帯域幅**

接続の種類	接続の数	データ幅 (ビット)	クロック ドメイン	接続あたりの帯域幅 (Gb/s)	全帯域幅 (GB/s)
AXI4-Stream スイッチから AI エンジン タイル	6	32	AI エンジン (1GHz)	4	24
AI エンジン タイルから AXI4-Stream スイッチ	4	32	AI エンジン (1GHz)	4	16

AI エンジン アレイ インターフェイスについては、この後のセクションで詳しく説明します。AI エンジン タイルの説明は、第 2 章: AI エンジン タイルのアーキテクチャ の章を参照してください。

AI エンジン アレイ インターフェイスの機能

- メモリ マップド AXI4 インターコネクト: NoC から受信したメモリ マップド AXI4 要求を AI エンジン アレイ内部に転送するための機能を備えています。
- AXI4-Stream インターコネクト: AI エンジン タイルのストリーミング インターコネクト機能を利用します。
- AI エンジンから PL へのインターフェイス: AI エンジンの PL モジュールは、PL と直接通信します。クロック乗せ換え処理のために非同期 FIFO が用意されています。
- AI エンジンから NoC へのインターフェイス: AI エンジンから NoC へのモジュールは、128 ビット NoC ストリームと 32 ビット AI エンジン ストリームを双方向に変換します。NoC コンポーネント (NMU および NSU) へのインターフェイス ロジックを提供します。NMU と NSU は、AI エンジンとは別の電源ドメインに属するため、レベル変換が実行されます。
- ハードウェア ロック: AI エンジン タイル内の対応するユニットを利用し、アレイ インターフェイスを外部メモリとの双方向 DMA 転送に同期させます。このモジュールには、AI エンジン アレイ インターフェイスまたは外部メモリ マップド AXI4 マスターからアクセスできます。
- デバッグ、トレース、およびプロファイリング: AI エンジン タイルのすべての機能を利用して、ローカル イベントのデバッグ、トレース、およびプロファイリングを実行します。

アレイ インターフェイスのメモリ マップド AXI4 インターコネクト

AI エンジン メモリ マップド AXI4 インターコネクトの主な用途は、コンフィギュレーションやデバッグのために内部 AI エンジン タイルのリソース (メモリやレジスタなど) へ外部からアクセスできるようにすることです。AI エンジン アレイとの間で大量のデータを転送する目的には使用しません。メモリ マップド AXI4 インターフェイスは AI エンジン アレイ インターフェイスの行全体で相互に接続されています。このため、アレイ インターフェイス タイルのメモリ マップド AXI4 インターコネクトは、受信したメモリ マップド信号を目的の列まで水平方向へ移動した後、これらをスイッチを経由してその列の一番下に位置する AI エンジン タイルのメモリ マップド AXI4 インターコネクトへ垂直方向へ転送できるようになっています。

各メモリ マップド AXI4 インターフェイスは、32 ビット アドレスと 32 ビット データで構成されます。メモリ マップド AXI4 の最大帯域幅は 1.5GB/s になるように設計されています。

メモリ マップド AXI4 インターフェイスへデータを供給するため、NoC モジュールにはメモリ マップド AXI4 ブリッジがあります。このブリッジは NoC NSU インターフェイスからメモリ マップド AXI4 転送を受信し、内部メモリ マップド AXI4 インターフェイス スイッチに対してメモリ マップド AXI4 マスターとして動作します。

アレイ インターフェイスの AXI4-Stream インターコネクト

AI エンジンの AXI4-Stream スイッチの主な目的は、AI エンジンとプログラマブル ロジックまたは NoC との間で確定的なスループットおよび高速回路またはパケット データフローを達成することです。このため、AI エンジン アレイとの間で大量のデータを移動できるように設計されています。AI エンジン タイルの一番下の行にある AXI4-Stream スイッチは、AI エンジン アレイ インターフェイス内で相互接続された AXI4-Stream スイッチの行と直接接続します。

AI エンジンからプログラマブル ロジックへのインターフェイス

AI エンジンの PL インターフェイス タイルにある AXI4-Stream スイッチは、AXI4-Stream インターフェイスを使用してプログラマブル ロジックと直接通信できます。AI エンジンから PL へは 6 つのストリームがあり、PL から AI エンジンの各列へは 8 つのストリームがあります。各 AXI4-Stream インターフェイスは、次の帯域幅をサポートできます。

- AI エンジンの各列から PL へは 24GB/s
- PL から AI エンジンの各列へは 32GB/s

VC1902 デバイスには 50 列の AI エンジン タイルと AI エンジン アレイ インターフェイス タイルがありますが、PL インターフェイスに利用可能なアレイ インターフェイス タイルは 39 のみです。したがって、PL インターフェイスの全帯域幅は次のようになります。

- AI エンジンから PL へは 1.0TB/s
- PL から AI エンジンへは 1.3TB/s

これらの帯域幅はいずれも、-1L スピード グレード デバイス (V_{CCINT} = 0.70V) で AI エンジン クロックが公称 1GHz の場合で計算しています。その他のデバイスおよびスピード グレードで PL インターフェイスに利用可能なアレイ インターフェイス タイルの数、および AI エンジンの PL インターフェイスの全帯域幅は、『Versal AI コア シリーズ データシート: DC 特性および AC スイッチ特性』 (DS957: 英語版、日本語版) を参照してください。

AI エンジンから NoC へのインターフェイス

AI エンジンの NoC インターフェイス タイルには、AXI4-Stream インターフェイスに加え、水平 NoC (HNoC) に接続するためのパスも含まれます。AI エンジンから見ると、AI エンジンから NoC へは 4 つのストリームがあり、NoC から AI エンジンへも 4 つのストリームがあります。AI エンジンの NoC インターフェイス タイルは HNoC と AXI4-Stream スイッチ間でトラフィックを転送できます。

ヒント: 実際の合計帯域幅は、デバイスで利用可能な水平および垂直チャネルの数と、NoC の帯域幅の制限により制限されます。

割り込み処理

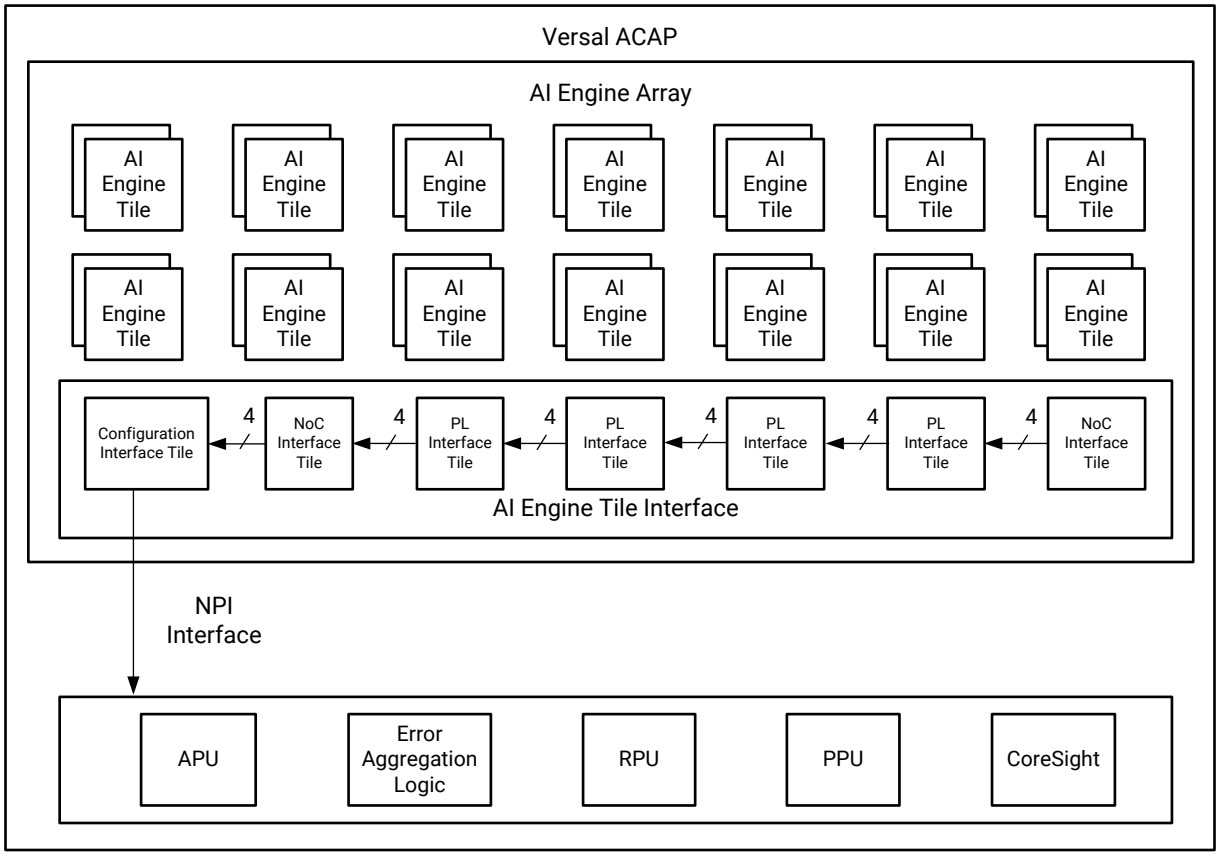
AI エンジン アレイ内のイベントによって、プロセッシング システム (PS) およびプラットフォーム管理コントローラー (PMC) への割り込みをトリガーするように設定できます。このセクションでは、AI エンジン アレイからの各種割り込みの概要について説明します。

AI エンジン アレイは 4 つの割り込みを生成します。これらの割り込みは、AI エンジン アレイから PMC、アプリケーション プロセッシング ユニット (APU)、およびリアルタイム プロセッシング ユニット (RPU) へ送信できます。AI エンジン アレイから生成される割り込みの全体的な階層は次のとおりです。

- イベントは、任意の AI エンジン タイルまたは AI エンジン インターフェイス タイルからトリガーされます。
- 各列には第 1 レベル割り込みハンドラーがあり、生成されたトリガー/イベントをキャプチャして第 2 レベル割り込みハンドラーへ転送できます。第 2 レベル割り込みハンドラーは NoC インターフェイス タイルにのみ存在します。
- 第 2 レベル割り込みハンドラーは、AI エンジン アレイ インターフェイスにある 4 つの割り込みラインのいずれか 1 つを駆動できます。
- これら 4 つの割り込みラインは、最終的に AI エンジンのコンフィギュレーション インターフェイス タイルに接続されます。

次の図は、AI エンジン アレイから Versal ACAP デバイスのほかのブロックへの NPI 割り込みの接続を示した概略ブロック図です。この図に示したアレイ インターフェイス タイルおよび AI エンジン タイルのレイアウト/配置は、実際のものとは異なります。

図 20: **AI エンジン アレイからほかのファンクション ブロックへの割り込みの接続**



上の図では、4 つの割り込みが NoC インターフェイス タイルから生成されています。これらの割り込みは、PL インターフェイス タイルを通過してコンフィギュレーション インターフェイス タイルに到達します。内部エラー (PLL ロック ロスなど) と 4 つの受信割り込みを OR 演算した結果、4 つの割り込みが NPI インターフェイス (32 ビット幅のメモリ マップド AXI4 バス) 上の NPI 割り込み信号に直接接続されます。

デバイス レベルでは、4 つの NPI 割り込みに 4 ～ 7 が割り当てられます。NPI レジスタには 3 つのグループがあります (IMR0...IMR3、IER0...IER3、および IDR0...IDR3)。各ペア (IMR、IER、および IDR) を使用して 4 つの NPI 割り込みを設定できます。IMR レジスタは読み出し専用で、IER および IDR レジスタは書き込み専用です。プログラムできるのは、NPI 割り込み 4 に対応するレジスタのみです。NPI 割り込み 5、6、および 7 に関しては、3 つのレジスタ セットは無視され、NPI レジスタをプログラムしても割り込みをマスクできません。NPI レジスタの構造とアドレスは、『Versal ACAP AI エンジン レジスタ リファレンス』 (AM015) で説明しています。

XILINX

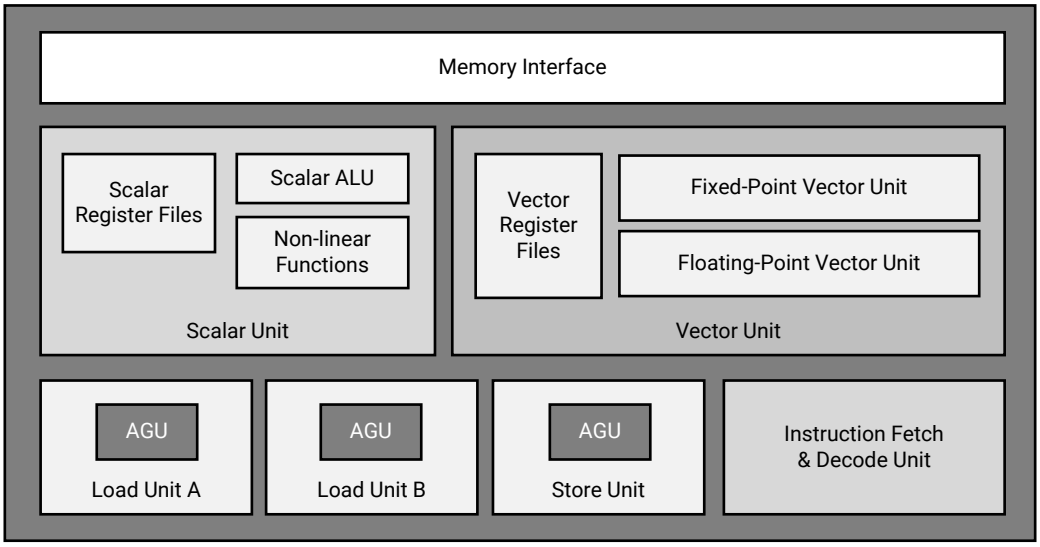
第 4 章

AI エンジンのアーキテクチャ

機能の概要

AI エンジンは SIMD (Single-Instruction Multiple-Data) および VLIW (Very-Long Instruction Word) プロセッサを内蔵した、高度に最適化されたプロセッサで、固定小数点精度と浮動小数点精度の両方をサポートしています。次の図に示すように、AI エンジンはメモリ インターフェイス、スカラー ユニット、ベクター ユニット、2 つのロード ユニット、1 つのストア ユニット、および命令フェッチ/デコード ユニットで構成されます。

図 21: **AI エンジン**



AI エンジンの主な特長は次のとおりです。

- 32 ビット スカラー RISC プロセッサ
  - 汎用ポインターおよびコンフィギュレーション レジスタ ファイル
  - 非線形関数 (sqrt、Sin/Cos、InvSqrt など) をサポート
  - スカラー ALU (32 x 32 ビット スカラー乗算器を含む)
  - スカラー固定小数点とスカラー浮動小数点の間でデータ型変換をサポート
- 3 つのアドレス生成ユニット (AGU)
  - 複数のアドレッシング モードをサポート: 固定、間接、ポスト インクリメント、または循環

- 高速フーリエ変換 (FFT) アドレス生成をサポート
- 2 つのロード ユニット専用の 2 つの AGU
- ストア ユニットに専用の 1 つの AGU

- ベクター固定小数点/整数ユニット
  - 複数のベクター レーンで同時に演算を実行
  - 複素数および実数オペランドに対して複数の精度をサポート (表 6 参照)

    表 6: **ベクター データパスでサポートされる精度**

X オペランド	Z オペランド	出力	MAC 演算回数
8 実数	8 実数	48 実数	128
16 実数	8 実数	48 実数	64
16 実数	16 実数	48 実数	32
16 実数	16 複素数	48 複素数	16
16 複素数	16 実数	48 複素数	16
16 複素数	16 複素数	48 複素数	8
16 実数	32 実数	48/80 実数	16
16 実数	32 複素数	48/80 複素数	8
16 複素数	32 実数	48/80 複素数	8
16 複素数	32 複素数	48/80 複素数	4
32 実数	16 実数	48/80 実数	16
32 実数	16 複素数	48/80 複素数	8
32 複素数	16 実数	48/80 複素数	8
32 複素数	16 複素数	48/80 複素数	4
32 実数	32 実数	80 実数	8
32 実数	32 複素数	80 複素数	4
32 複素数	32 実数	80 複素数	4
32 複素数	32 複素数	80 複素数	2
32 単精度浮動小数点	32 単精度浮動小数点	32 単精度浮動小数点	8

  - 8 つの 16 ビット複素数乗算を実行するように構成可能
  - 32 ビット粒度の完全な並べ替えユニット
  - シフト、丸め、および飽和 (丸めと飽和は複数のモードを利用可能)
  - 2 段階の後置加算 (768 ビットの中間結果)
  - X オペランドは 1024 ビット幅で、Z オペランドは 256 ビット幅です。コンポーネントの使用に関しては、表 6 の最初の行を考慮に入れてください。乗算器オペランドは同じ 1024 ビットおよび 256 ビット入力レジスタから得られますが、一部の値は複数の乗算器にブロードキャストされます。128 個の 8 ビット シングル乗算器があり、結果は後置加算された後で累算されて、16 個または 8 個のアキュムレータ レーン (各 48 ビット) に入ります。

- 単精度浮動小数点 (SPFP) ベクター ユニット
  - 固定小数点数ベクター ユニットと同じ並べ替えを使用

    - 複数のベクター レーンで同時に演算を実行
    - 1 サイクルで 8 回の単精度 MAC (積和演算)
- バランスのとれたパイプライン
    - ファンクション ユニットごとに種類の異なるパイプライン (最大 8 段)
    - ロードおよびストア ユニットがデータ メモリの 5 サイクル レイテンシを管理
- 3 つのデータ メモリ ポート
    - 2 つのロード ポートと 1 つのストア ポート
    - 各ポートは 256 ビット/128 ビット ベクター レジスタ モードまたは 32 ビット/16 ビット/8 ビット スカラー レジスタ モードで動作。8 ビットおよび 16 ビットのストアは Read-Modify-Write 命令として実行
    - 3 つすべてのポートで同時に演算を実行
    - いずれかのポートでバンクが競合するとデータパス全体がストール
- VLIW (Very-Long Instruction Word) ファンクション
    - すべてのファンクション ユニットへ命令を同時発行
    - 複数の命令フォーマットおよび可変長命令をサポート
    - 1 つの VLIW ワードを使用して最大 7 つの演算を並列に発行可能
- ダイレクト ストリーム インターフェイス
    - 2 つの入力ストリームと 2 つの出力ストリーム
    - いずれのストリームも 32 ビットまたは 128 ビット幅に構成可能
    - 1 つのカスケード ストリーム入力、1 つのカスケード ストリーム出力 (384 ビット)
- 次のモジュールへのインターフェイス
    - ロック モジュール
    - ストール モジュール
    - デバッグ/トレース モジュール
- イベント インターフェイスは AI エンジンからの 16 ビット幅出力インターフェイス

レジスタ ファイル

AI エンジンにはいくつかの種類のレジスタがあります。レジスタの種類によって、使用するファンクション ユニットが異なります。このセクションでは、各種レジスタについて説明します。

スカラー レジスタ

スカラー レジスタには、コンフィギュレーション レジスタが含まれます。次に、各レジスタの説明を示します。

表 7: **スカラー レジスタ**

構文	ビット数	説明
r0..r15	32 ビット	汎用レジスタ
m0..m7	20 ビット	修飾子レジスタ
p0..p7	20 ビット	ポインター レジスタ
cl0..cl7	32 ビット	コンフィギュレーション レジスタ
ch0..ch7		
c0..c7	64 ビット	

特殊レジスタ

表 8: **特殊レジスタ**

構文	ビット数	説明
cb0..cb7	20 ビット	循環バッファー開始アドレス
cs0..cs7	20 ビット	循環バッファー サイズ
wcs0..wcs3	40 ビット	ワイド循環バッファー サイズ
s0..s7	8 ビット	シフト制御
sp	20 ビット	スタック ポインター
lr	20 ビット	リンク レジスタ
pc	20 ビット	プログラム カウンター
fc	20 ビット	フェッチ カウンター
mc0..mc1	32 ビット	ステータス レジスタ
md0..md1	32 ビット	モード制御レジスタ
ls	20 ビット	ループ開始
le	20 ビット	ループ終了
lc	32 ビット	ループ カウント
lci	32 ビット	ループ カウント (PCU)
S	8 ビット	シフト制御

ベクター レジスタ

ベクター レジスタは、SIMD 命令で使用する高ビット幅のレジスタです。基本となる最下層のハードウェア レジスタは 128 ビット幅で、接頭辞 V が付きます。V レジスタを 2 つ組み合わせて 256 ビット レジスタを構成できます (接頭辞 W)。WR、WC、および WD レジスタを 2 つ組み合わせて 512 ビット レジスタ (XA、XB、XC、および XD) を構成します。XA と XB で 1024 ビット幅の YA レジスタを構成します。YD を除くすべてのレジスタは、表の一番上が LSB、表の一番下が MSB の順になります。YD は、XD が LSB、SB が MSB を構成します。すなわち、次のようになります。

$$YD = VDL0::VDH0::VDL1::VDH1::VRL2::VRH2::VRL3::VRH3$$

表 9: **ベクター レジスタ**

128 ビット	256 ビット	512 ビット	1024 ビット	
vrl0	wr0	xa	ya	N/A
vrh0				
vrl1	wr1			
vrh1				
vrl2	wr2	xb		yd (MSB)
vrh2				
vrl3	wr3			
vrh3				
vcl0	wc0	xc	N/A	N/A
vch0				
vcl1	wc1			
vch1				
vdl0	wd0	xd	N/A	yd (LSB)
vdh0				
vdl1	wd1			
vdh1				

アキュムレータ レジスタ

アキュムレータ レジスタは、ベクター データパスの結果を格納するために使用します。このレジスタは 384 ビット幅で、48 ビットのベクター レーン x 8 と見なすことができます。このようにして、32 ビットの乗算結果をビット オーバーフローなしに加算できるようにしています。16 ビットのガード ビットにより、最大 2^{16} 回の累算が可能です。アキュムレータ レジスタには、接頭辞 AM が付きます。これらのレジスタを 2 つ組み合わせて 768 ビット レジスタを構成します (接頭辞は BM)。

注記: 動作モードは 2 つあります。1 つは、乗算結果を 8 個のアキュムレータに入力して 16 回の後置加算を実行してから累算するモードです。もう 1 つは、乗算結果を 16 個のアキュムレータに入力して 8 回の後置加算を実行してから累算するモードです。

表 10: **アキュムレータ レジスタ**

384 ビット	768 ビット
aml0	bm0
amh0	
aml1	bm1
amh1	
aml2	bm2
amh2	
aml3	bm3
amh3	

命令フェッチ/デコード ユニット

命令フェッチ/デコード ユニットは、現在のプログラム カウンター (PC) レジスタの値をアドレスとしてプログラム メモリに送信します。プログラム メモリはフェッチした 128 ビット幅の命令値を返します。この命令値をデコードした後、すべての制御信号が AI エンジンのファンクション ユニットへ送信されます。AI エンジンのプログラム メモリ サイズは 16KB のため、128 ビット幅の命令を 1024 個格納できます。

AI エンジンの命令は 128 ビット幅で、複数の命令フォーマットおよび可変長命令をサポートしており、プログラム メモリ サイズを抑えることができます。VLIW スロットをすべて使用する場合は、ほとんどのケースで 128 ビット全体が必要となります。一方、外周ループ、メイン プログラム、制御コード、そして場合によっては内周ループのプリアンブル/ポストアンブルの命令の多くはそれよりも短いフォーマットで十分であり、圧縮した命令を小容量の命令バッファーに格納できます。

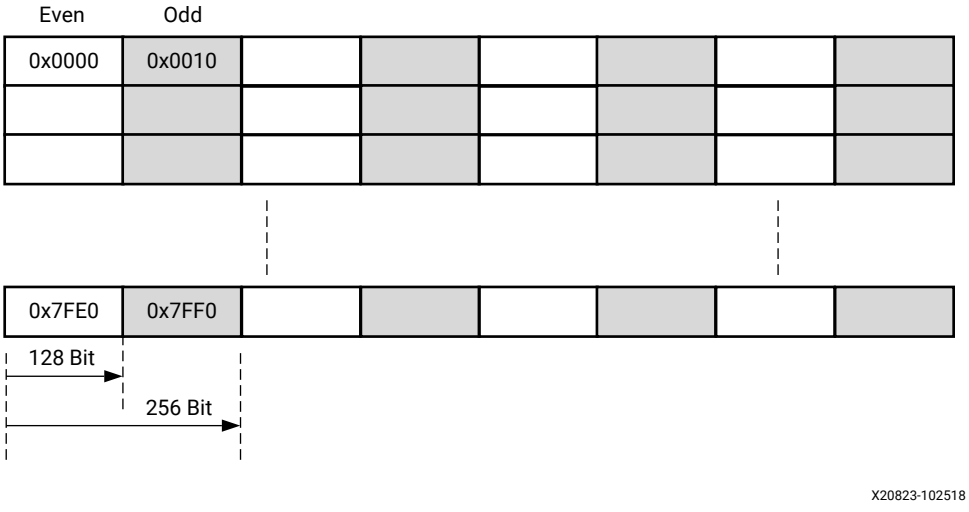
ロードおよびストア ユニット

AI エンジンは、2 つのロード ユニットと 1 つのストア ユニットを使用してデータ メモリにアクセスします。データはデータ メモリにロードまたはストアされます。

各ロードおよびストア ユニットには、アドレス生成ユニット (AGU) があります(ロード ユニットは AGUA と AGUB、ストア ユニットは AGUS)。各 AGU には、P レジスタ ファイルと M レジスタ ファイルからそれぞれ 20 ビットずつが入力されます (レジスタ ファイル のポインター レジスタと修飾子レジスタを参照)。AGU のレイテンシは 1 サイクルです。

個々のデータ メモリ ブロックは 32KB です。AI エンジンは 4 つの 32KB データ メモリ ブロックにアクセスして 128KB ユニットを構成します。これら 4 つのメモリ ブロックは AI エンジンの各サイドにあり、奇数バンクと偶数バンクとして分割され、インターリーブされます (下図参照)。

図 22: **データ メモリのインターリーブ (1 ブロックあたり 32KB)**

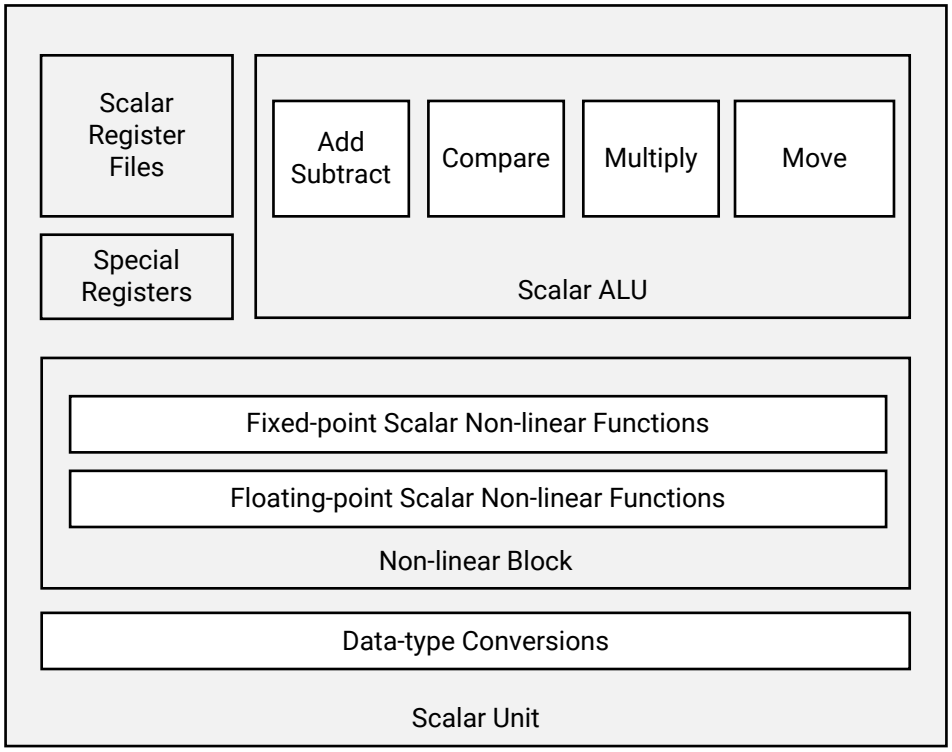


論理表現では、128KB メモリを 1 つの連続した 128KB ブロックとして、または 4 つの 32KB ブロックとして見ることができ、各ブロックを奇数バンクと偶数バンクに分割できます。メモリは 8 つの 16KB バンク (4 つの奇数バンクと 4 つの偶数バンク) として見ることもできます。データ メモリへのアクセス用に、AGU は `0x0000` ～ `0x1FFFF` (128KB) のアドレスを生成します。

スカラー ユニット

スカラー ユニットには、スカラー レジスタ ファイルとスカラー ファンクション ユニットが含まれます。次に、スカラー ユニットのブロック図を示します。

図 23: **AI エンジンのスカラー ユニット**



スカラー ユニットには次のファンクション ブロックが含まれます。

- レジスタ ファイルおよび特殊レジスタ
- 論理演算ユニット (ALU)
- 非線形関数 – 固定小数点および浮動小数点精度
- データ型の変換

整数の加算、減算、比較、およびシフト関数は 1 サイクルで実行されます。整数の乗算には 3 サイクルのレイテンシがあります。非線形関数は、1 または 4 サイクルでスカラー値の結果を出力します。これら演算のスループットはいずれも 1 サイクルです。

ALU、スカラー関数、およびデータ型変換

AI エンジンの論理演算ユニット (ALU) は、次の演算を実行します。いずれの場合も、発行レートは 1 サイクルあたり 1 命令です。

- 整数の加算および減算: 32 ビット。この演算のレイテンシは 1 サイクルです。
- 32 ビット整数に対するビット論理演算 (BAND、BOR、BXOR)。この演算のレイテンシは 1 サイクルです。
- 整数の乗算: 32 x 32 ビット。32 ビットの結果を R レジスタ ファイルに格納。この演算のレイテンシは 3 サイクルです。
- シフト演算: 左シフトと右シフトの両方をサポート。シフト量が正の場合は左シフト、負の場合は右シフトを実行します。シフト量は汎用レジスタを使用して渡します。シフト演算で正と負のどちらのシフトが必要かは、1 ビットのオペランドで示します。この演算のレイテンシは 1 サイクルです。

AI エンジンには、固定小数点と浮動小数点の 2 種類のスカラー初等関数があります。次に、各関数について説明します。

- 固定小数点非線形関数
  - サインおよびコサイン
    - 32 ビット入力の上位 20 ビットを入力
    - 上位 16 ビットのサインと下位 16 ビットのコサインを 1 ワードに連結して出力
    - これら演算のレイテンシは 4 サイクルです。
  - 絶対値 (ABS): 入力値を反転して 1 を加算。この演算のレイテンシは 1 サイクルです。
  - CLZ (Count Leading Zeroes): 32 ビット入力の先行ゼロの数。この演算のレイテンシは 1 サイクルです。
  - 最小値/最大値 (< (LG)/> (GT)): 2 つの入力を比較して最小値または最大値を求める。この演算のレイテンシは 1 サイクルです。
  - 平方根、逆平方根、および逆数: これらの演算は浮動小数点精度で実装されます。固定小数点実装では、入力をまず浮動小数点精度に変換してからこれら非線形関数の入力として渡す必要があります。また、出力は浮動小数点フォーマットであるため、固定小数点整数フォーマットへ戻す必要があります。これら演算のレイテンシは 4 サイクルです。
- 浮動小数点非線形関数
  - 平方根: 入力と出力はいずれも単精度浮動小数点数で、R レジスタ ファイルに対して演算を実行します。この演算のレイテンシは 4 サイクルです。
  - 逆平方根: 入力と出力はいずれも単精度浮動小数点数で、R レジスタ ファイルに対して演算を実行します。この演算のレイテンシは 4 サイクルです。
  - 逆数: 入力と出力はいずれも単精度浮動小数点数で、R レジスタ ファイルに対して演算を実行します。この演算のレイテンシは 4 サイクルです。
  - 絶対値 (ABS): この演算のレイテンシは 1 サイクルです。
  - 最小値/最大値: この演算のレイテンシは 1 サイクルです。

AI エンジンのスカラー ユニットは、入力データを固定小数点と浮動小数点の間で双方向に変換するデータ型変換をサポートしています。`fix2float` 演算と `float2fix` 演算は、32 ビット値の入力に対して可変小数点をサポートしています。32 ビット値と小数点位置の両方を入力として与えます。この演算は、必要に応じて値を拡大または縮小します。いずれの演算もレイテンシは 1 サイクルです。

AI エンジンの浮動小数点は、IEEE 規格に完全に準拠しているわけではなく、一部の機能には制限があります。例外は次のとおりです。

- `float2fix` 関数の引数が非常に大きな正数または負数で、追加の指数インクリメントがゼロより大きい場合、この命令は正しい飽和値である $2^{31}-1$ または -2^{31} ではなく、0 を返します。

  `float2fix` 関数には、入力パラメーターが 2 つあります。

  - n: 変換される浮動小数点の入力値。
  - sft: 固定小数点表記の小数ビット数を表す 6 ビットの符号付き数値 (-32 ～ 31)。

  次に 2 つの例を紹介します。

  - $n*2^{sft} > 2^{129}$ の場合、出力は `0x7FFFFFFF` を返すはずが、代わりに、`0x00000000` を返す。
  - $n*2^{sft} < -2^{129}$ の場合、出力は `0X80000000` を返すはずが、代わりに、`0x00000000` を返す。

  通常、浮動小数点の入力値 `n` は、バグのない範囲 (sft > 0 の場合に $-2^{(129-sft)} < n < 2^{(129-sft)}$) 内にする必要があります。

  対処方法として、次の 2 つがあります。

  - `float2fix_safe`: オプションなしで `float2fix` を指定する場合のデフォルト モードです。この場合、任意の範囲に対して正しい値が返されますが、低速になります。
  - `float2fix_fast`: バグのない範囲のみでの正しい値が返されるので、その範囲が有効であることを確認する必要があります。`floatfix_fast` を選択する場合は、プロジェクト ファイルにプリプロセッサの FLOAT2FIX_FAST を追加する必要があります。

- 固定小数点値の有効範囲は、-2^{31} ～ $2^{32}-1$ です。`float2fix` 関数が -2^{31} を返す場合、この値は範囲内ですが、オーバーフロー例外が正しく設定されません。このオーバーフロー例外への対処法はありません。

ベクター ユニット

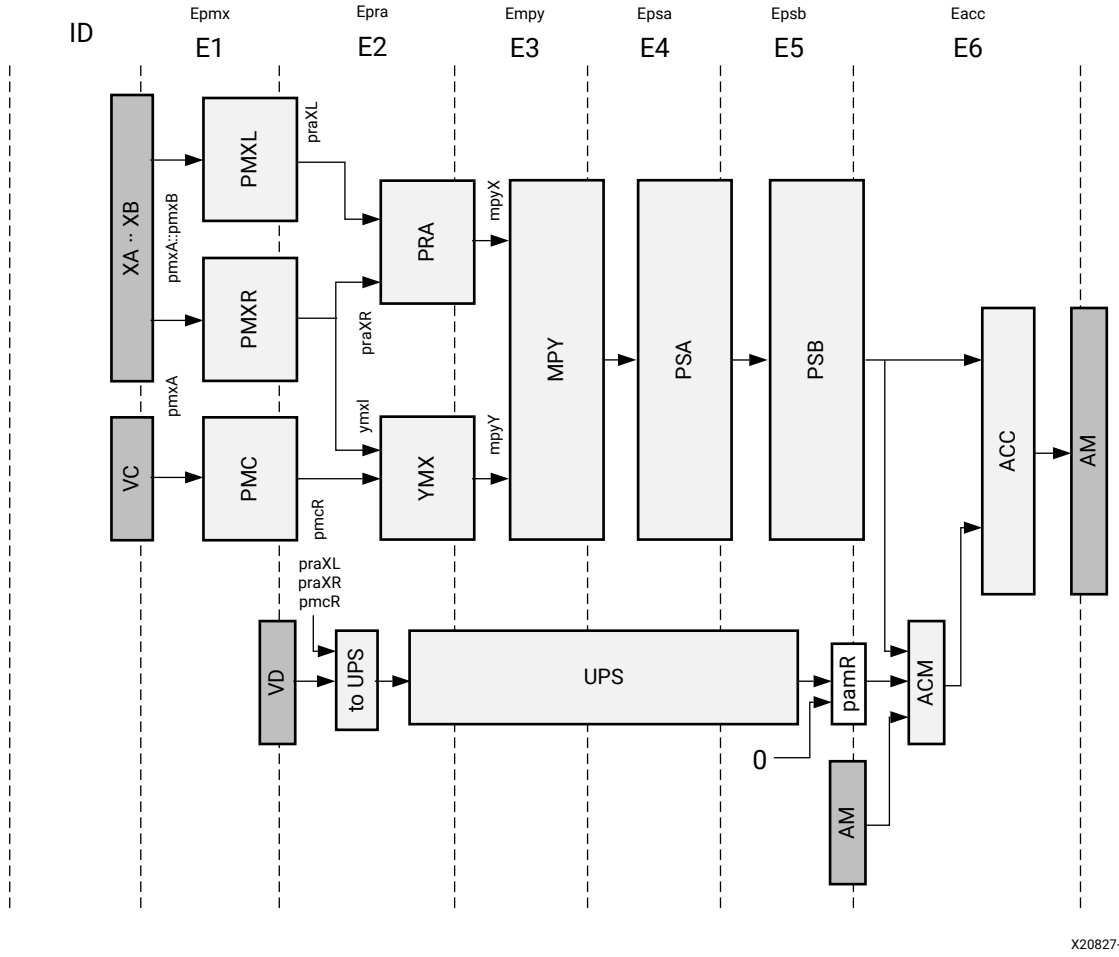
固定小数点数ベクター ユニット

固定小数点数ベクター ユニットには 3 つのデータパスが含まれ、これらはほぼ独立しています。

- 積和演算 (MAC) パス: メインの乗算パスはベクター レジスタから値を読み出し、ユーザー制御可能な方法でこれらを並べ替え、前置加算 (オプション) の後に乗算を実行し、後置加算した値をアキュムレータ レジスタの直前の値に加算します。
- アップシフト パス: このパスは MAC パスと並列に動作します。MAC パスの並べ替えユニットまたはベクター レジスタからデータを読み出し、左シフトを実行してアキュムレータ レジスタへ渡します。
- シフト-丸め-飽和 (SRS) パス: このパスはアキュムレータ レジスタから値を読み出し、ベクター レジスタまたはデータ メモリに値を格納します。このパスが必要なのは、アキュムレータがレーンあたり 48 または 80 ビット幅であるのに対し、ベクター レジスタとデータ メモリは 2 の累乗である 8、16、32、または 64 ビット幅であるためです。したがって、レーンごとにデータを右シフトする必要があります。SRS ユニットは、飽和/丸め制御レジスタ MD のフィールド Q と R を使用して動作を制御し、ステータス レジスタ MC を使用して情報を環境へ戻します。シフト量は、シフト制御レジスタ S で決定します。このユニットは、レジスタ MD のフィールド R の値に応じてさまざまな丸めモードをサポートします。R = 0 の場合、LSb 側の値を切り捨てます。R = 1 の場合は天井関数の動作となり、丸めは実行されません。R = 2 ～ 7 の場合、モードは順に PosInf、NegInf、SymInf、SymZero、ConvEven、ConvOdd となります。

次の図に、メインの乗算およびアップシフト パスのパイプラインを示します。命令デコード ステージ (ID) の後、E1 ～ E6 の 6 つの実行ステージがあります。2 つのステージをまたいだ濃いグレーの四角形は、レジスタを表しています。薄いグレーの四角形はファンクション ユニットを表しており、複数のステージをまたぐことも可能です。白の四角形は、プロセッサ記述内部のハードウェア レジスタを表しています。すべての四角形は矢印で接続されています。これらは純粋な非記憶ワイヤで、nML ではトランジトリと呼ばれます。図に示した要素以外に、実行される命令に応じて異なる接続を可能にするマルチプレクサーもあります。「to UPS」ユニットは、3 つの並べ替えユニットと VD レジスタのいずれかを選択するマルチプレクサーを意味します。入力を読み出し、2 つの値を加算してからデータを UPS ユニットへ出力する内部ユニットがあります。

図 24: **AI エンジンの固定小数点ベクター ユニットの乗算およびアップシフト パスのパイプライン**



次の表に、メインの乗算パスのファンクション ユニットを示します。

表 11: **乗算パスのファンクション ユニット**

ファンクション ユニット	説明
並べ替えユニット	
PMXL	前置加算器 PRA の左側への入力用にベクター レジスタからのデータを並べ替えます。
PMXR	前置加算器 PRA の右側への入力用、または YMX ユニットへの入力用にベクター レジスタからのデータを並べ替えます。
PMC	YMX ユニットへの入力用にベクター レジスタからのデータを並べ替えます。

表 11: **乗算パスのファンクション ユニット** (続き)

ファンクション ユニット	説明
前置加算器ユニット	
PRA	PMXL と PMXR の出力に対して前置加算または減算を実行し、これを乗算器の第 1 引数として使用します。また、PMXL/PMXR ユニットの 32 ビット粒度を補償するため、いくつかの制限された並べ替えが実行されます。
YMX	定数 1 を挿入して前置加算器の結果に 1 を掛けたり、個々のレーンに符号拡張を実行したりする特殊演算。また、PMXR ユニットの 32 ビット粒度を補償するため、いくつかの制限された並べ替えが実行されます。
乗算器ユニット	
MPY	PRA と YMX の出力を掛け合わせます。
後置加算器ユニット	
PSA	MPY の出力 (32 レーン) を 16 レーンに減らす第 1 後置加算ステージ。
PSB	レーン数をさらに 8 まで減らす第 2 後置加算ステージ。または、入力を出力に転送します。
アキュムレータ	
ACM	後置加算器の出力に加算するデータをマルチプレクサーに入力します。アキュムレータの直前の値、アップシフト パスの出力、またはカスケード ストリーム入力のいずれかを選択できます。
ACC	ACM と PSB の出力に対して加算または減算を実行します。

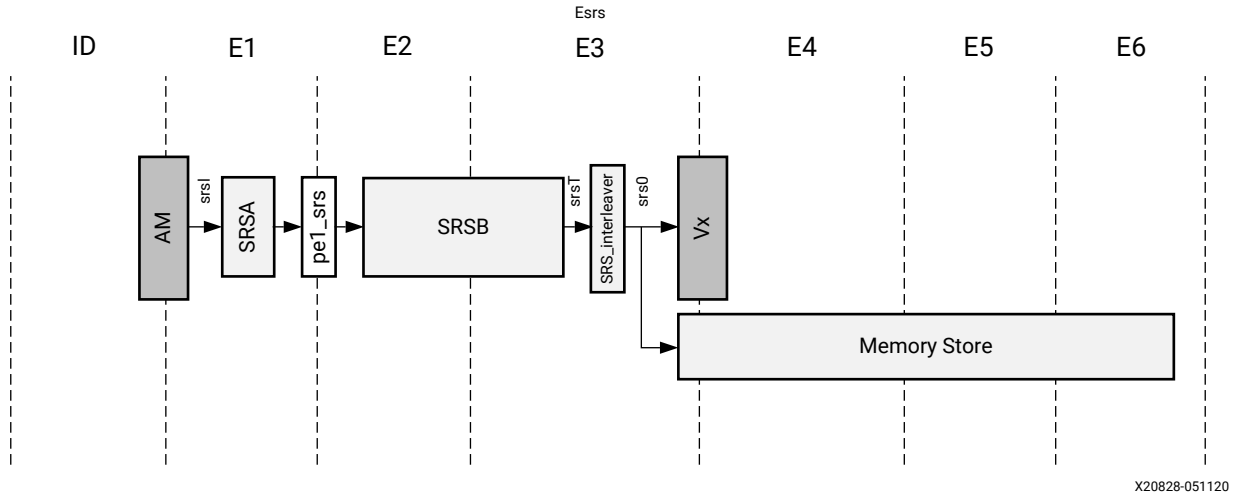
次の表に、アップシフト パスのファンクション ユニットを示します。

表 12: **アップシフト パス**

アップシフト ユニット	説明
UPS へ	ベクター レジスタを読み出し、特定のレーンのみを選択します。
UPS	実際のアップシフトを実行し、メイン データパスの ACM ユニットへ出力します。

次の図に、シフト-丸め-飽和パスのパイプラインを示します。アキュムレータ レジスタを読み出してシフト-丸め-飽和が実行され、その出力がいずれかのベクター レジスタまたはデータ メモリに書き込まれます。この値は E3 ステージでメモリに格納され、E6 ステージでメモリに到達します。

図 25: **AI エンジンのシフト-丸め-飽和データパスのパイプライン**



次の表に、シフト-丸め-飽和パスのファンクション ユニットを示します。

表 13: **シフト-丸め-飽和パス**

シフト-丸め-飽和ユニット	説明
SRSA	80 ビット アキュムレータの 2 つの部分を結合します。48 ビット アキュムレータをシフトする場合は、データをバイパスします。48 ビット x 8 レーンまたは 80 ビット x 4 レーンに対して並列に処理を実行します。この機能は下位と上位の 2 つに分割されており、同じ演算が並列に実行されます。
SRSB	レーンに対して実際にシフトを実行します。この機能は下位と上位の 2 つに分割されており、同じ演算が並列に実行されます。
SRS インターリーバー	アキュムレータのインターリーブが必要な場合 (シフト量レジスタ S の MSb で制御)、SRSB 上位および下位ユニットの出力をインターリーブします。

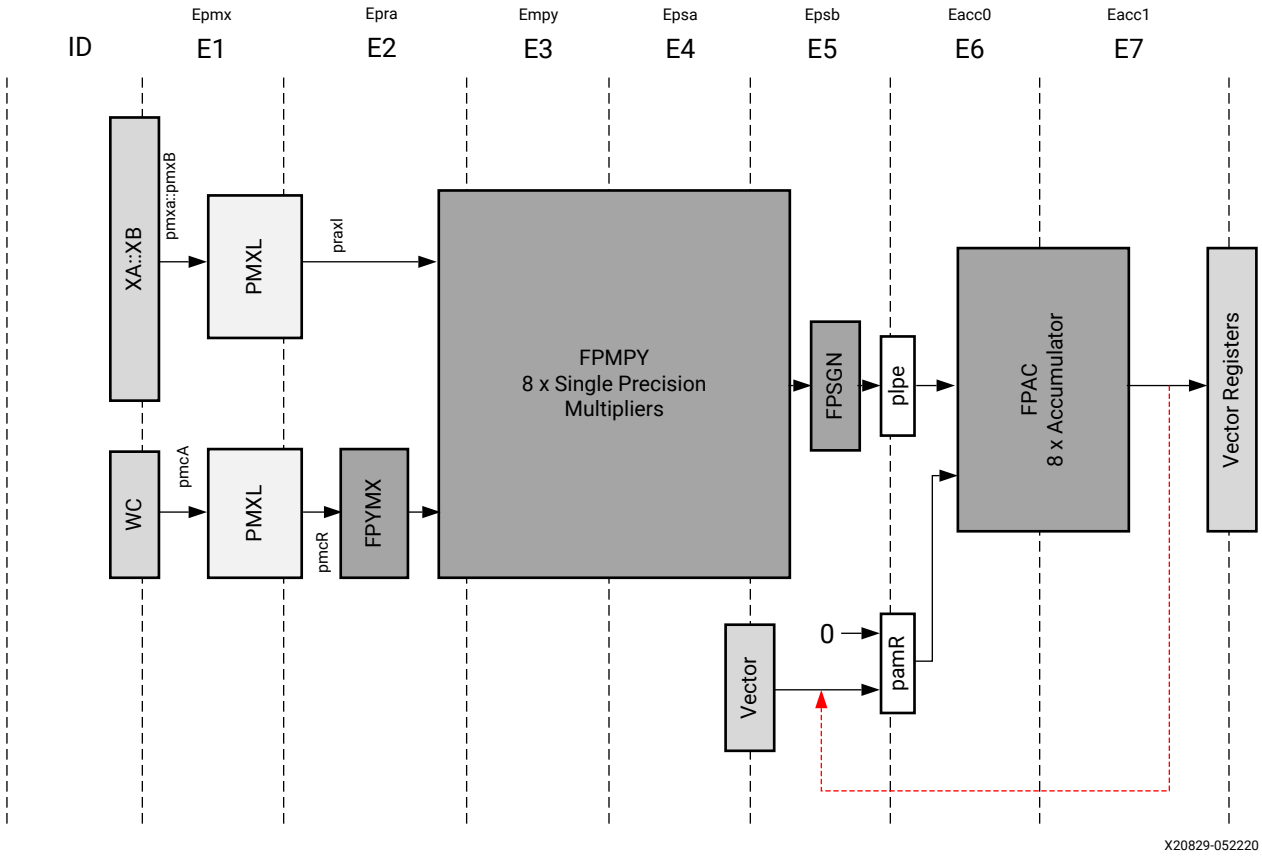
浮動小数点ベクター ユニット

AI エンジンには 8 レーンの単精度浮動小数点積和演算があります。このユニットは、固定小数点データパスと同じベクター レジスタ ファイルおよび並べ替えネットワークを使用します。通常は、固定小数点または浮動小数点全体で、1 サイクルあたり 1 ベクター命令のみを実行できます。

次の図に、単精度浮動小数点データフローのパイプラインを示します。固定小数点ベクター ユニットとは異なり、PMXL および PMC ユニットのみを使用します (PMXR ユニットはなし)。FPYMX は YMX のスタイルで、FPYMX と PMXL の結果が単精度乗算器ユニット (FPMPY) に渡され、8 つの積が並列に計算されます。FPMPY の演算はレイテンシが 3 サイクル、スループットが 1 サイクルです。次に、レーンごとの結果に負の符号を加える FPSGN ユニットがあります。

FPSGN ユニットの次には、FPACC と呼ばれる 2 段のアキュムレータ ユニットがあります。このユニットは、乗算結果を 0 やその他のベクター レジスタから直接取得した値など、さまざまなソースからの値と累算します。ただし、同じベクター内のレーンは直接加算できません。このアキュムレータは FPSGN ユニットによって処理されるため、減算はサポートしません。

図 26: **AI エンジンの浮動小数点ベクター ユニット単精度浮動小数点データパスのパイプライン**



AI エンジンは、浮動小数点フォーマットのベクター初等関数をいくつかサポートしています。これには、ベクター比較、最小値、および最大値が含まれます。浮動小数点データパスは、ベクター固定小数点から単精度浮動小数点への変換および反対方向の浮動小数点から固定小数点への変換をサポートしていますが、スカラー ユニットを経由するため性能は低くなります。この場合、ベクターから抽出されたエレメントを抽出し、スカラー変換を実行し、その結果をベクターに戻すようにします。これを効率よくパイプライン ループに実装すると、1 サイクルあたりほぼ 1 サンプルの性能を達成できます。

次の機能は、AI エンジンの浮動小数点データパスではサポートされません。

- 倍精度演算
- 半精度演算
- カスタム浮動小数点フォーマット (指数部 2 ビット、仮数部 14 ビットの E2:M14 など)
- 乗算前の前置加算
- 乗算とアキュムレータ間の後置加算
- 乗算器とアキュムレータ間での精度向上
- 非正規化した浮動小数点数

レジスタの移動機能

このセクションでは、AI エンジンのレジスタ移動機能について説明します。レジスタの種類と名前の詳細は、レジスタ ファイル を参照してください。

- スカラーからスカラー
  - R、M、P、C、および特殊レジスタの間でスカラー値を移動
  - R、M、P、C、および特殊レジスタへ即値を移動
  - AXI4-Stream との間で双方向にスカラー値を移動
- ベクターからベクター: 1 つの 128 ビット V レジスタを任意の V レジスタへ 1 サイクルで移動。256 ビット W レジスタおよび 512 ビット X レジスタの場合も同様です。ただし、いずれの場合もベクター サイズが同じである必要があります。
- アキュムレータからアキュムレータ: 384 ビット アキュムレータ (AM) レジスタを別の AM レジスタへ 1 サイクルで移動します。
- ベクターからアキュムレータ (次の 2 つの場合)
  - アップシフト パスは 16 または 32 ビット ベクター値を取得し、アキュムレータに書き込みます。
  - 通常の乗算データパスを使用し、それぞれの値に定数 `1` を掛けます。
- アキュムレータからベクター: シフト-丸め-飽和データパスは、アキュムレータをベクター レジスタへ移動します。
- アキュムレータからカスケード ストリーム、およびカスケードからアキュムレータ: カスケード ストリームはアレイにある AI エンジンをチェーン接続し、AI エンジン間でのアキュムレータ レジスタ (384 ビット) 転送を可能にします。入力ストリームと出力ストリームの両方に 384 ビット幅の 2 段 FIFO があるため、AI エンジン間で最大 4 つの値を FIFO に格納できます。
- スカラーからベクター: R レジスタのスカラー値をベクター レジスタへ移動。
- ベクターからスカラー: 128 ビットまたは 256 ビット ベクター レジスタから任意の 8 ビット、16 ビット、または 32 ビット値を抽出し、結果をスカラー R レジスタへ書き込みます。

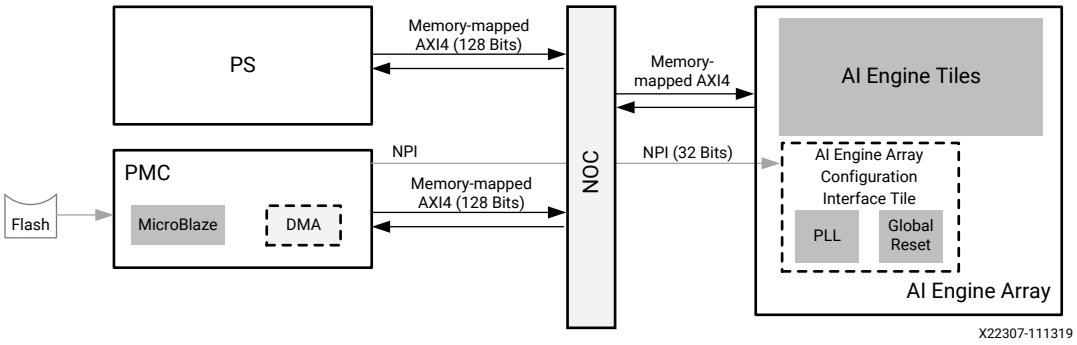
第 5 章

AI エンジンのコンフィギュレーションとブート

AI エンジン アレイのコンフィギュレーション

AI エンジン アレイのコンフィギュレーションには、電源投入時の AI エンジン アレイのコンフィギュレーションと、AI エンジン アレイのパーシャル リコンフィギュレーションの大きく 2 とおりがあります。次に、AI エンジン アレイとコンフィギュレーション インターフェイス、および NoC を経由して PS とプラットフォーム管理コントローラー (PMC) を接続するレジスタの概略図を示します。

図 27: **NoC と NPI を使用した AI エンジン アレイの構成**



PS や PMC など、任意のメモリ マップド AXI4 マスターが、NoC を使用して AI エンジン アレイ内のすべてのメモリ マップド AXI4 レジスタを設定できます。グローバル レジスタは NPI アドレス空間にマップされるため、アレイ コンフィギュレーション インターフェイス タイル内のグローバル レジスタ (PLL コンフィギュレーション、グローバル リセット、セキュリティ ビットなど) は、NPI インターフェイスを使用してプログラムできます。

AI エンジンのブート シーケンス

このセクションでは、AI エンジン アレイのブート プロセスに関連する手順を説明します。

1. 電源投入およびパワーオン リセット (POR) のディアサート: AI エンジン アレイに関連するすべてのモジュール (PLL を含む) に電源が投入されます。電源投入後、PLL はデフォルトの速度で動作します。AI エンジンのブート シーケンスを開始する前に、プラットフォーム管理コントローラー (PMC) と NoC が動作を開始している必要があります。アレイの電源を投入後、PMC は AI エンジン アレイ内の POR 信号をディアサートできます。

2. NPI を使用した AI エンジン アレイのコンフィギュレーション: 電源投入後、PMC は NPI インターフェイスを使用して AI エンジン アレイ内の各種グローバル レジスタ (PLL コンフィギュレーション レジスタなど) をプログラムします。AI エンジン アレイの初期化のために NPI を介して要求される AI エンジン コンフィギュレーション イメージは、フラッシュ デバイスから取得します。
3. PLL の有効化: POR の後に PLL レジスタのコンフィギュレーションが完了したら、PLL イネーブル ビットを有効にして PLL をオンにします。その後、PLL はプログラムされた周波数で安定し、[LOCK] 信号をアサートします。PLL の入力 ([ref_clk]) は、CIPS (Control Interfaces and Processing System) で生成される [hsm_ref_clk] から供給されます。
    - このクロックの生成と分配の詳細は、『Versal ACAP テクニカル リファレンス マニュアル』(AM011) の「PMC および PS クロック」の章で説明しています。
4. リセットのリリース: PLL がロックしたら、ソフトウェアでレジスタをプログラムして AI エンジン アレイのグローバル リセット信号をディアサートします。
5. AI エンジン アレイのプログラム: AI エンジン アレイ インターフェイスは、NoC インターフェイスからメモリ マップド AXI4 を介してコンフィギュレーションする必要があります。これには、すべての AXI4 ストリーム スイッチ、メモリ マップド AXI4 スイッチ、アレイ インターフェイス DMA、イベント、およびトレース コンフィギュレーション レジスタが含まれます。

AI エンジン アレイのリコンフィギュレーション

AI エンジンのコンフィギュレーション プロセスでは、[bootgen] ツールで生成したプログラマブル デバイス イメージ (PDI) が AI エンジン コンフィギュレーション レジスタに書き込まれます。AI エンジンのコンフィギュレーションは、NoC を経由してメモリ マップド AXI4 上で実行されます。AI エンジン アレイのコンフィギュレーションは、NoC に接続した任意のマスターから実行できます。[bootgen] ツールを使用して PDI を生成する方法の詳細は、『Versal ACAP AI エンジン プログラミング環境ユーザー ガイド』 (UG1076) を参照してください。

AI エンジン アレイは、いつでもリコンフィギュレーションが可能です。リコンフィギュレーションは、アプリケーションが駆動します。安全にリコンフィギュレーションを実行するには、次の条件を満たす必要があります。

- トラフィック継続中にリコンフィギュレーションを実行しない。
- リコンフィギュレーションの前に、AI エンジン - PL インターフェイスを無効にする。
- 前のコンフィギュレーションからのデータが残っていると悪影響を及ぼす可能性があるため、リコンフィギュレーションの前にサブ領域のデータをすべて排出する。

AI エンジン アレイのリコンフィギュレーションには、次の 2 つの方法があります。

- 完全なリコンフィギュレーション: AI エンジン アレイに対してグローバル リセットをアサートした後、新しいコンフィギュレーション イメージをダウンロードしてアレイ全体をリコンフィギュレーションします。
- パーシャル リコンフィギュレーション: アレイ内の一部の AI エンジン タイルのみをリコンフィギュレーションし、その他のタイルはカーネルの実行を継続します。リコンフィギュレーションを実行しても、AI エンジン アレイ内で実行中のカーネルは影響を受けません。

AI エンジン アレイの初期化は、PMC と PS が実行します。次の表に、グローバル AI エンジン アレイで利用可能なリセット制御をまとめます。

表 14: **AI エンジンのリセットのカテゴリ**

タイプ	トリガー	範囲
内部パワーオン リセット	ブート シーケンスの一部	AI エンジン アレイ

表 14: **AI エンジンのリセットのカテゴリ** (続き)

タイプ	トリガー	範囲
システム リセット	NPI 入力	AI エンジン アレイ
INITSTATE リセット	PCSR ビット	AI エンジン アレイ
アレイ ソフト リセット	ソフトウェアによる NPI 経由のレジスタ書き込み	AI エンジン アレイ
AI エンジン タイル列リセット	アレイ インターフェイス タイルのメモリマップド AI エンジン レジスタ ビット	AI エンジン タイル列
AI エンジン アレイ インターフェイス リセット	NPI レジスタから	AI エンジン アレイ インターフェイス タイル

列リセットとアレイ インターフェイス タイル リセット (AI エンジン アレイ階層 参照) を組み合わせることで、AI エンジン タイルとアレイ インターフェイス タイルで構成されるサブアレイを、隣接するサブアレイに影響を与えずリセットして再プログラムするパーシャル リコンフィギュレーションのユース ケースがサポートされます。アレイ分割の処理および分離の追加方法は、ユース ケースの種類 (マルチ ユーザー/テナントまたはシングル ユーザー/マルチテナント タスク) により異なります。

付録 A

その他のリソースおよび法的通知

ザイリンクス リソース

アンサー、資料、ダウンロード、フォーラムなどのサポート リソースは、ザイリンクス サポート サイトを参照してください。

Documentation Navigator およびデザイン ハブ

ザイリンクス Documentation Navigator (DocNav) では、ザイリンクスの資料、ビデオ、サポート リソースにアクセスでき、特定の情報を取得するためにフィルター機能や検索機能を利用できます。DocNav を開くには、次のいずれかを実行します。

- Vivado® IDE で [Help]→[Documentation and Tutorials] をクリックします。
- Windows で [スタート]→[すべてのプログラム]→[Xilinx Design Tools]→[DocNav] をクリックします。
- Linux コマンド プロンプトに「`docnav`」と入力します。

ザイリンクス デザイン ハブには、資料やビデオへのリンクがデザイン タスクおよびトピックごとにまとめられており、これらを参照することでキー コンセプトを学び、よくある質問 (FAQ) を参考に問題を解決できます。デザイン ハブにアクセスするには、次のいずれかを実行します。

- DocNav で [Design Hub View] タブをクリックします。
- ザイリンクス ウェブサイトでデザイン ハブ ページを参照します。

注記: DocNav の詳細は、ザイリンクス ウェブサイトの Documentation Navigator ページを参照してください。DocNav からは、日本語版は参照できません。ウェブサイトのデザイン ハブ ページをご利用ください。

参考資料

次の文書は、このユーザー ガイドの補足資料として役立ちます。日本語版のバージョンは、英語版より古い場合があります。

1. Versal ACAP データシート:
   - 『Versal アーキテクチャおよび製品データシート: 概要』 (DS950: 英語版、日本語版)
2. 『Versal ACAP AI エンジン レジスタ リファレンス』 (AM015)

3. 『Versal ACAP AI エンジン プログラミング環境ユーザー ガイド』 (UG1076: 英語版、日本語版)

お読みください: 重要な法的通知

本通知に基づいて貴殿または貴社 (本通知の被通知者が個人の場合には「貴殿」、法人その他の団体の場合には「貴社」。以下同じ) に開示される情報 (以下「本情報」といいます) は、ザイリンクスの製品を選択および使用することのためにのみ提供されます。適用される法律が許容する最大限の範囲で、(1) 本情報は「現状有姿」、およびすべて受領者の責任で (with all faults) という状態で提供され、ザイリンクスは、本通知をもって、明示、黙示、法定を問わず (商品性、非侵害、特定目的適合性の保証を含みますがこれらに限られません)、すべての保証および条件を負わない (否認する) ものとします。また、(2) ザイリンクスは、本情報 (貴殿または貴社による本情報の使用を含む) に関係し、起因し、関連する、いかなる種類・性質の損失または損害についても、責任を負わない (契約上、不法行為上 (過失の場合を含む)、その他のいかなる責任の法理によるかを問わない) ものとし、当該損失または損害には、直接、間接、特別、付随的、結果的な損失または損害 (第三者が起こした行為の結果被った、データ、利益、業務上の信用の損失、その他あらゆる種類の損失や損害を含みます) が含まれるものとし、それは、たとえ当該損害や損失が合理的に予見可能であったり、ザイリンクスがそれらの可能性について助言を受けていた場合であったとしても同様です。ザイリンクスは、本情報に含まれるいかなる誤りも訂正する義務を負わず、本情報または製品仕様のアップデートを貴殿または貴社に知らせる義務も負いません。事前の書面による同意のない限り、貴殿または貴社は本情報を再生産、変更、頒布、または公に展示してはなりません。一定の製品は、ザイリンクスの限定的保証の諸条件に従うこととなるので、https://japan.xilinx.com/legal.htm#tos で見られるザイリンクスの販売条件を参照してください。IP コアは、ザイリンクスが貴殿または貴社に付与したライセンスに含まれる保証と補助的条件に従うことになります。ザイリンクスの製品は、フェイルセーフとして、または、フェイルセーフの動作を要求するアプリケーションに使用するために、設計されたり意図されたりしていません。そのような重大なアプリケーションにザイリンクスの製品を使用する場合のリスクと責任は、貴殿または貴社が単独で負うものです。https://japan.xilinx.com/legal.htm#tos で見られるザイリンクスの販売条件を参照してください。

自動車用のアプリケーションの免責条項

オートモーティブ製品 (製品番号に「XA」が含まれる) は、ISO 26262 自動車用機能安全規格に従った安全コンセプトまたは余剰性の機能 (「セーフティ 設計」) がない限り、エアバッグの展開における使用または車両の制御に影響するアプリケーション (「セーフティ アプリケーション」) における使用は保証されていません。顧客は、製品を組み込むすべてのシステムについて、その使用前または提供前に安全を目的として十分なテストを行うものとします。セーフティ設計なしにセーフティ アプリケーションで製品を使用するリスクはすべて顧客が負い、製品の責任の制限を規定する適用法令および規則にのみ従うものとします。

この文書は暫定的な情報を含むものであり、通知なしに内容が変更されることがあります。この文書に記述される情報は、販売前の製品・サービスに関するもので、情報目的としてのみ提供されており、この文書で参照されている製品・サービスの販売申込みまたは製品の商品化を試みたものとしては意図されておらず、また解釈されるものでもありません。

著作権

この資料に関するフィードバックおよびリンクなどの問題につきましては、jpn_trans_feedback@xilinx.com まで、または各ページの右下にある [フィードバック送信] ボタンをクリックすると表示されるフォームからお知らせください。フィードバックは日本語で入力可能です。いただきましたご意見を参考に早急に対応させていただきます。なお、このメール アドレスへのお問い合わせは受け付けておりません。あらかじめご了承ください。