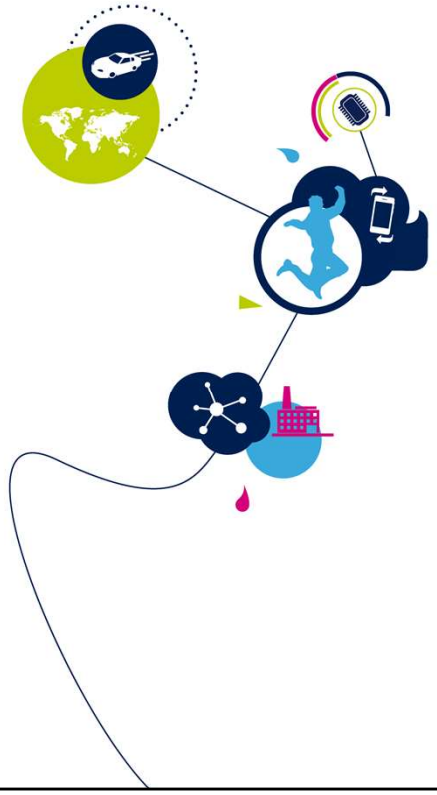
STM32H7- ARM® コア

Arm Cortex®-M7コア

1.0版



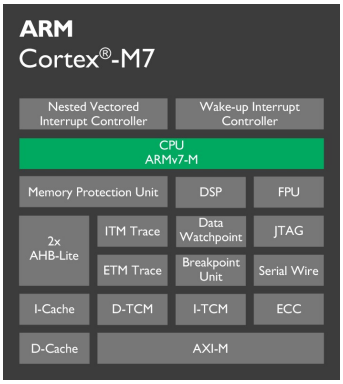
こんにちは、ARM®のプレゼンテーションへようこそCortex®-M7コアは、STM32H7マイクロコントローラファミリのすべての製品に組み込まれています。

Cortex-M7プロセッサの概要

2

- ARMv7E-Mアーキテクチャ
- ハーバード･アーキテクチャ、6-ステージ･パイプライン
- デュアルIssue(発行)のスーパースカラアーキテクチャ!
- DIV12サイクル(最大)、SIMDインストラクション
- メモリ保護ユニット(MPU)
- シングル、ダブル-制度浮動小数点ユニット(FPU)

  ⇒Arm Cortex®-M7に基づきSTM32製品に組み込まれている



DSPに1歩近づく	リアルタイム･プロセッサに1歩近づく
算術演算と並行して読み込みと保存を実行	Tightly Coupledメモリ
オーバーヘッドループゼロ	キャッシュ･メモリを備えたAXI-Mインタフェース

life.augmented

Cortex®-M7コアは、32ビットRISCコアのARM Cortex®-Mグループの一部です。ARMv7E-Mアーキテクチャを実装します。

6段パイプラインと、単精度浮動小数点ユニットとSIMDサポートを備えたデュアルIssueスーパースカラを搭載しています。

Cortex®-M7コアの性能は、Cortex®-M4コアよりもデジタル信号プロセッサの性能にはるかに近いものです。

ループのオーバーヘッドをゼロにして、算術演算を使用して、ロード操作と格納操作を並列実行できます。

Cortex®-M7コアは、Tightly Coupledメモリまたは TCMと直接インタフェースを持ち、割込み遅延が非常に短いため、より決定的な実行が実現されます。

Cortex-Mの互換性

3

- **従来の8/16/32ビットの分類は忘れてください**
  - すべてのアプリケーションに対してシームレスなアーキテクチャ
  - どの製品も超低電力と使いやすさのために最適化

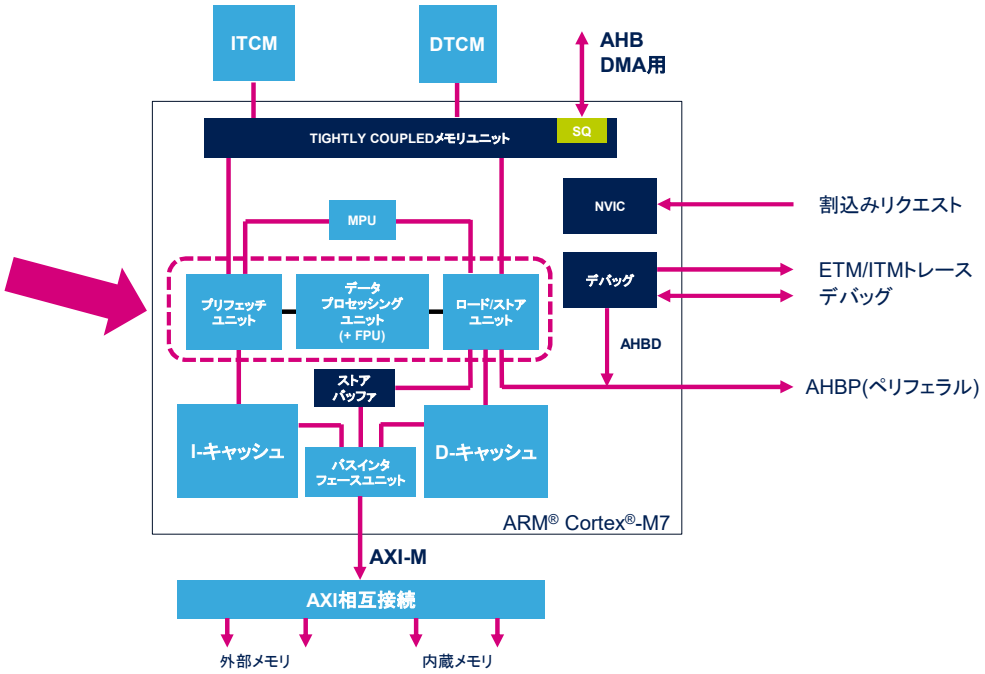
Cortex-M0 & M0+	Cortex-M3	Cortex-M4	Cortex-M7
「8/16ビット」アプリケーション	「16/32ビット」アプリケーション	「32ビット/DSC」アプリケーション	

バイナリ／ツール互換 → ハイパフォーマンス

life.augmented

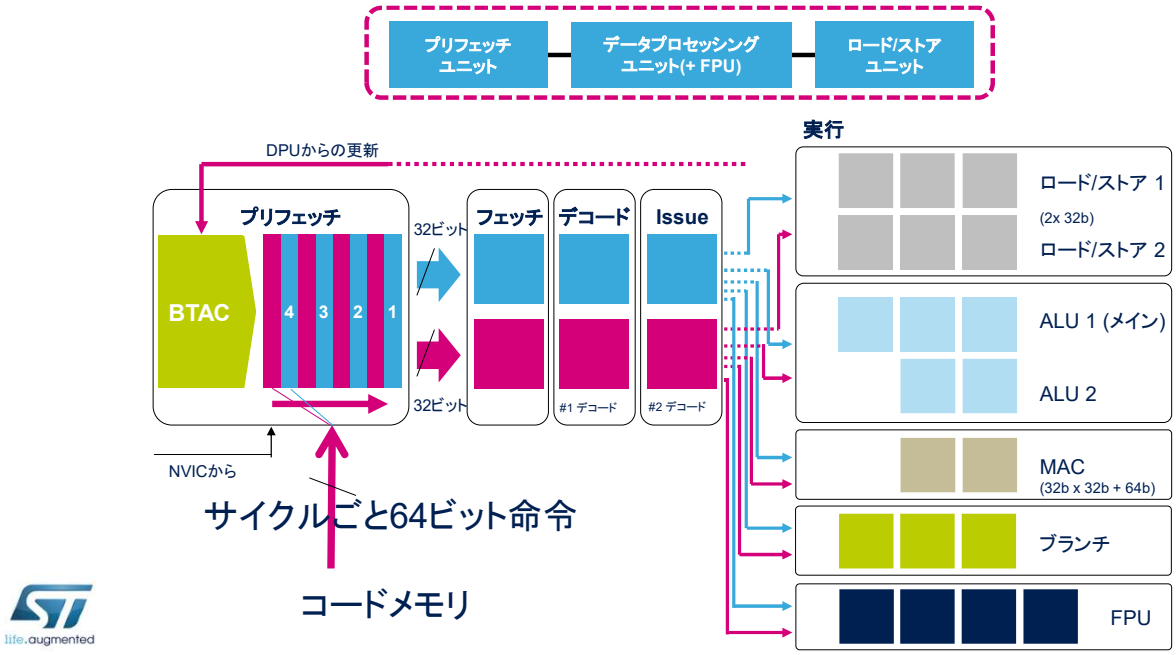
STM32H7マイクロコントローラは、Cortex®-Mプロセッサアーキテクチャのパフォーマンス向上、特に低消費電力モードでの高レベルの性能の恩恵を受けるために、ARM®Cortex®-M7コアを統合します。

コア·アーキテクチャの概要



Cortex®-M7コアは、アーキテクチャの強化により処理能力が向上し、Cortex®-M4コアよりも高いパフォーマンスを発揮します。
このパフォーマンスを向上させるために、プリフェッチユニット(PFU)から命令を実行する3つのユニットを調べ、次にロードストアユニット(LSU)と共にデータ処理単位(DPU)を実行します。

ARM Cortex-M7 → デュアルIssue（発行）



プリフェッチユニット（PFU）は、1サイクルあたり1つの64ビット命令をデータプロセッシングユニット（DPU）に提供します。
以下が含まれます。

- DPUの前にフェッチを可能にするために、それぞれ64ビットの4つのエントリのバッファ
- 単一サイクル分岐予測のためのブランチターゲットアドレスキャッシュ(BTAC)

Cortex®-M7コアは、効率的な動作のために6段のデュアル・Issue(発行)・パイプラインを備えています。一定の条件を満たせば、2つの命令を並行して処理することができます。

データプロセッシングユニット（DPU）は、いくつかのパイプに分かれています。

- 2つのALUを持ち、1つのALUはSIMD演算が可能
- 1サイクルに1つのMACを搭載したシングルMACパイプライン
- 1つの浮動小数点パイプで単精度と倍精度の演算をサポートしています。

命令がIssue段階に到達すると、命令はマイクロオペレーションに分割され、必要なオペレーションと使用されるレジスタに基づいて決定されます。そして、処理パイプのさらに先にある適切なブロックに発行されます。

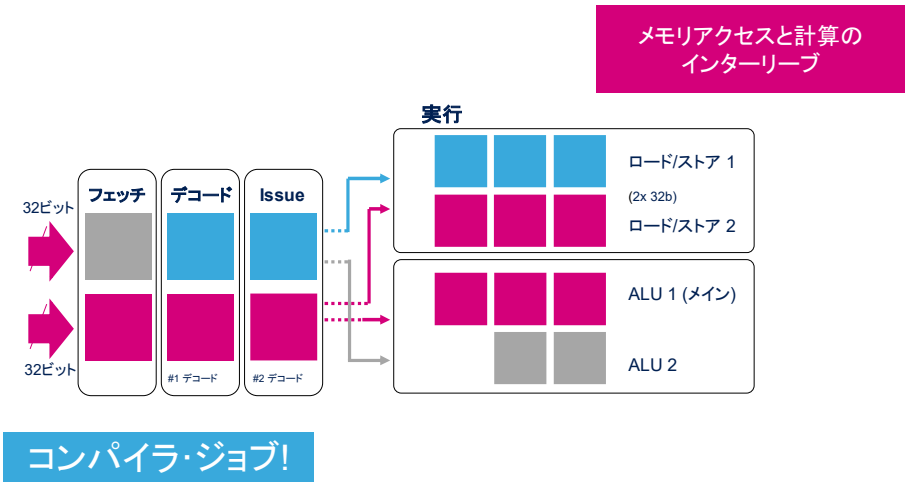
DPUからPFUにフラグを転送することで、デコーダやパイプラインの最初の実行段階での直接分岐を早期に解決することができます。

ロードストアユニット（LSU）は、32ビットのデュアルロードチャネルまたは64ビットのシングルストアチャネルを提供し、ストアのスループットを向上させるためのストアバッファリングを備えています。

コンパイラは、コアのパイプラインの複雑さを隠し、このアーキテクチャを活用するためにコードを最適化します。

ロードとストアを算術演算と並行して実行

- Cortex®-M7コア
  - ペナルティなしで可能なメモリ·アクセス

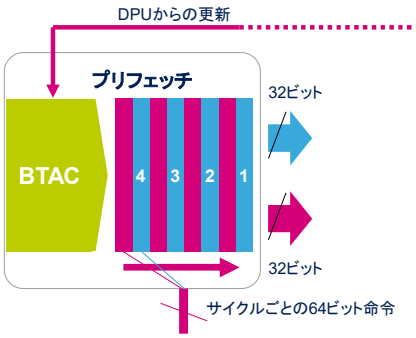


コンパイラ·ジョブ!

Cortex®-M4 コアと比較して、最も重要な利点は、コードが2つの32ビット値(1つの命令で二重負荷)を読み取り、同時にMACパイプ上の前の2つのデータを読み取ることができることです。Cortex®-M7コアは、計算の長いシーケンスでより効率的です。

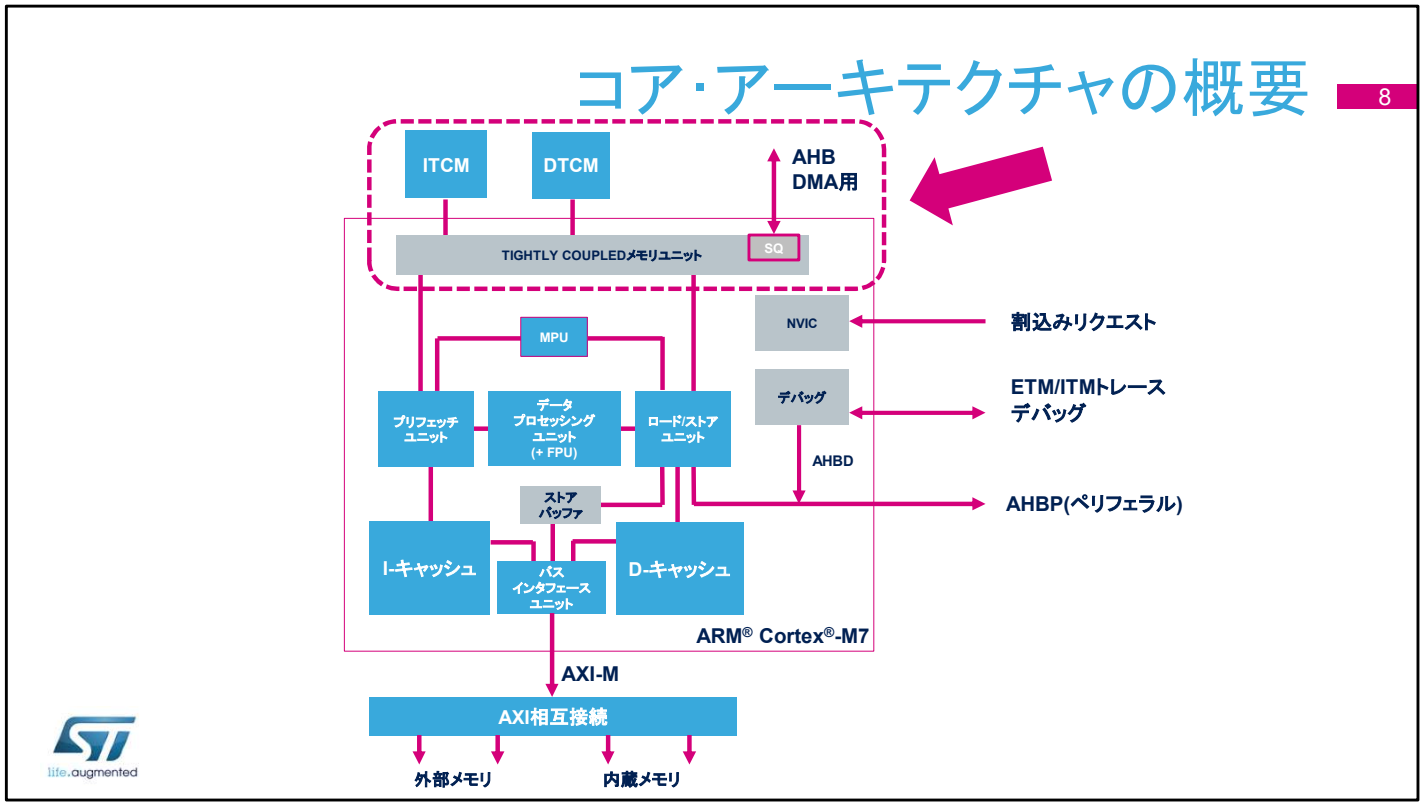
オーバーヘッドループゼロ

- 条件付きループにはインクリメント / デクリメント＋分岐実行が必要
- Cortex-M7の場合：分岐予測とスーパースカラ・デュアル・Issue・アーキテクチャにより、1サイクルが必要



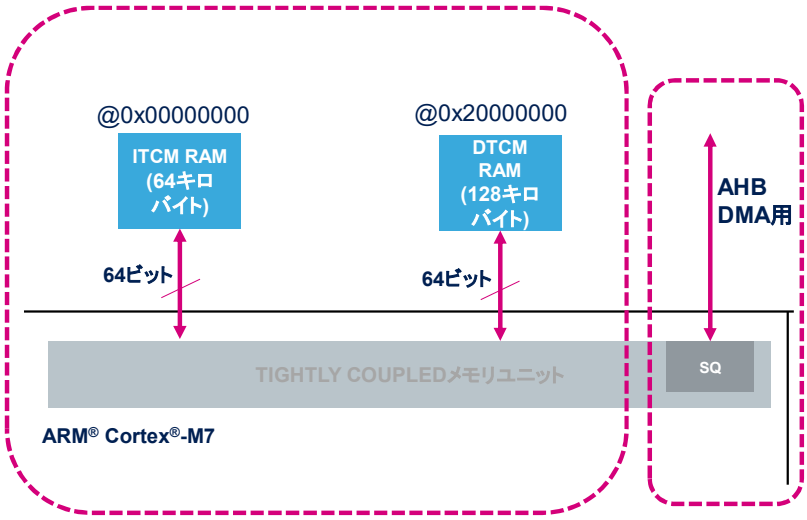
分岐は二重に発行することもできるので、計算と並行して実行することができます。

ブランチターゲットアドレスキャッシュ(BTAC)は、ブランチを取得できるかどうかを予測し、それに応じて反応します。条件を記憶し、処理に基づいて、次のアドレスをフェッチするを予測します。



TCM（Tightly-Coupledメモリ）は、バスを介さずにプロセッサに直接接続された専用メモリで、頻繁に実行されるコードのアービトレーション（調停）や遅延を回避することができます。
命令用（ITCM）とデータ用（DTCM）のTightly Coupledメモリは、これらのインタフェースを介してアクセスされる重要なデータや命令を静的にマッピングすることができます。これは、ベクタテーブル、割込みサービスルーチン、頻繁に実行され低レイテンシで確定的な実行時間を必要とする特定のタイムクリティカルな制御ループなどが該当します。
AHBSは、DMAが緊密に結合されたRAMのいずれかにアクセスする手段を提供します。
STM32H7デバイスは、TCMメモリにECC保護も実装しています。

Tightly coupledメモリ(TCM)



TCMメモリは厳密にゼロウェイト

ITCM RAMは、コアのフェッチバンド幅を満たすために、64ビットのメモリインタフェースを1つ持っています。
STM32H7マイクロコントローラは、ITCMインタフェースを介して64キロバイトのSRAMへのアクセスが可能です。
DTCM RAMには、要求に応じて並列処理を行うために2つの32ビットメモリインタフェースがあります。ソフトウェアは、重要なデータに最大128キロバイトのSRAMを使用できます。
ITCMでは、コードをITCMに、データをDTCMに配置することで、12クロックサイクルの割込みレイテンシを実現しています。
AHBSは、32ビットのAMBA3 AHB-Lite Slaveインタフェースです。AHBSは、ITCMとDTCMへのシステムアクセス(DMAなど)を提供します。AHBSは、システムとプロセッサの同時アクセス要求をサポートしています。

Tightly coupledメモリ(TCM)

ITCM RAM (64KB)

- 重要なコード
- 割込みサービス･ルーチン
- 高い決定性

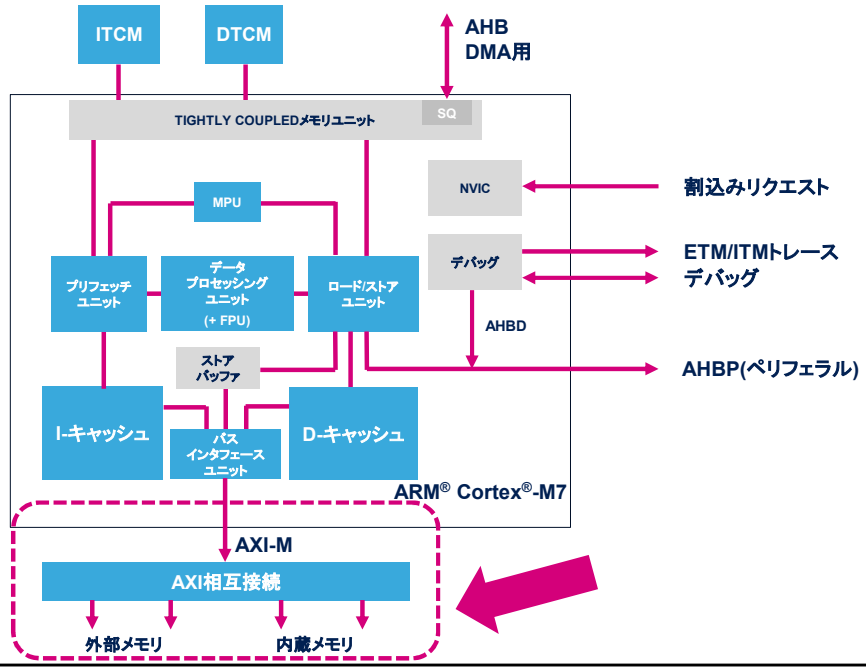
DTCM RAM (128KB)

- 頻繁に使用されるデータ
- スタック / ヒープ
- DSP係数

life.augmented

ここでは、ITCM RAMとDTCM RAMを使用する場合があります。

コア·アーキテクチャの概要

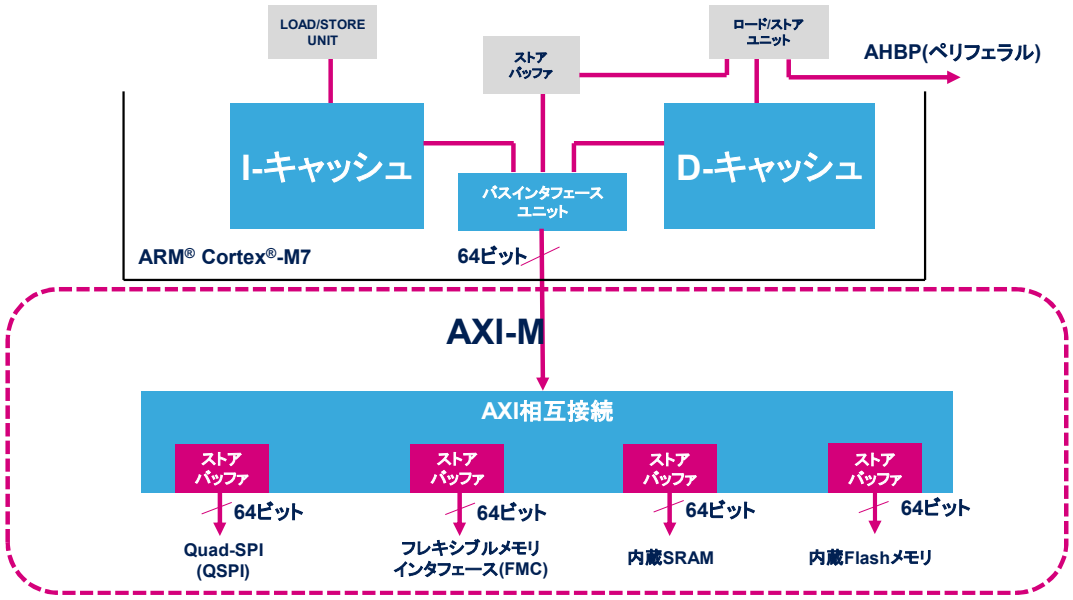


複数のAHBバスがメモリシステムとの並列トランザクションに必要なCortex®-M4コアと比較すると、Cortex®-M7コアは単一のAXIマスタバスを統合します。

AXIマスタ(AXI-M)インタフェースは、バスインタフェースユニット(BIU)の一部です。64ビット幅のAXIインタフェースで、CPUと内部および外部メモリを接続します。次の目的で使用できます。

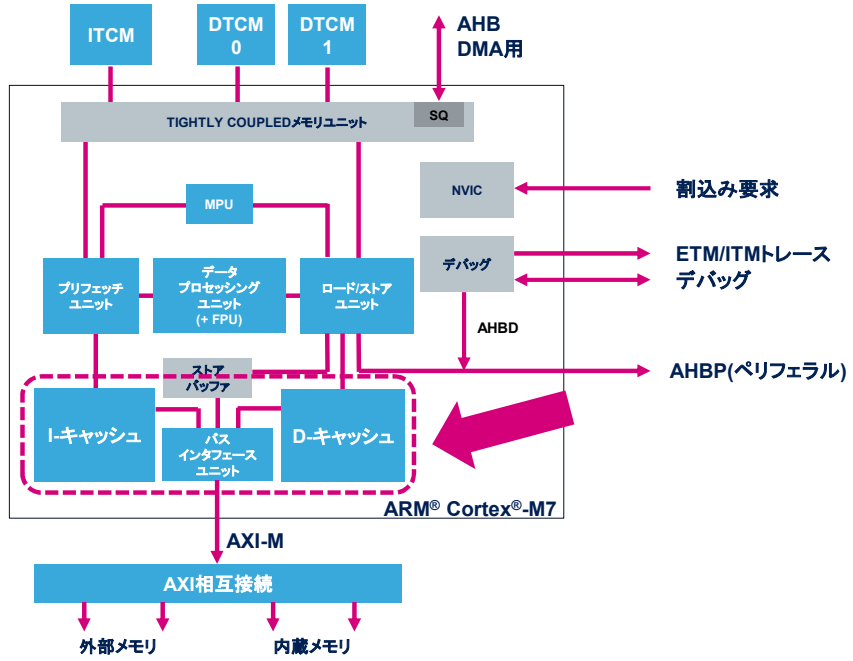
- 命令フェッチ
- データキャッシュラインのフィルと消去
- ノンキャッシャブル、ノーマルタイプのメモリデータアクセス
- デバイスと強順序型のデータアクセス

AXI-Mインタフェース



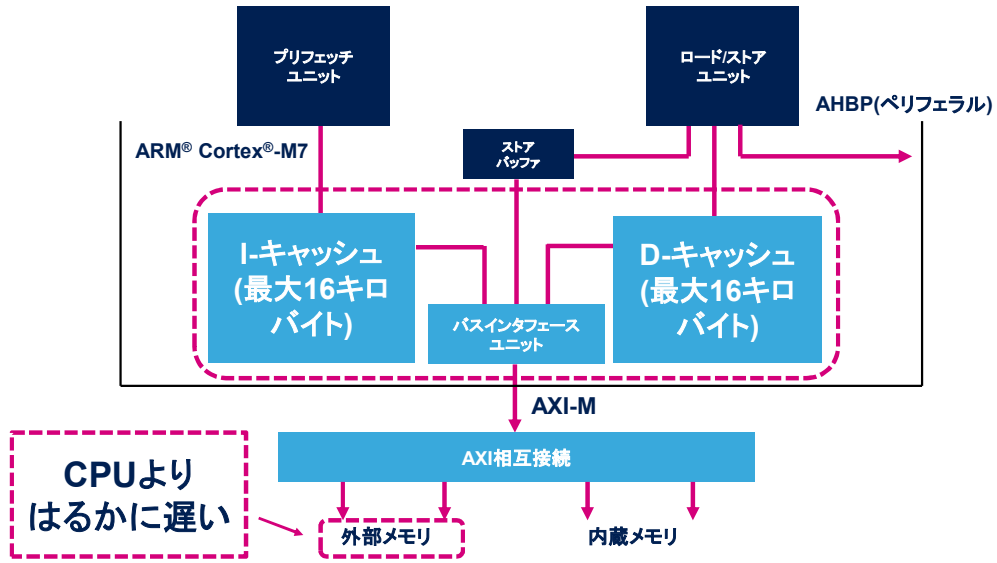
STM32H7マイクロコントローラは、Cortex®-M7のAXIMインタフェースを活用するために、AXIバスマトリックスを統合しています。
AXIでは、アクセスリクエストとデータフェーズとの相関を外しているため、リクエストは対応するデータから独立しています。メモリにレイテンシがある場合、2つのリクエストの間に機能的な関係が存在しなければ、この分離によってバスは新しいリクエストを実行できるようになります（例：命令フェッチとデータフェッチが並行して実行される）。

コア·アーキテクチャの概要



Cortex®-M7コアは、AXIMインターフェースに加えて、オプションの命令キャッシュとデータキャッシュを統合し、効率的なメモリアクセスを実現しています。
キャッシュが有効な場合、TCMやAHBPインタフェイス以外のアクセスは、適切なキャッシュコントローラによって管理されます。キャッシュがヒットした場合、キャッシュ能力の基準が満たされていれば、データはキャッシュRAMに取り込まれたり、書き込まれたりします。
キャッシュが無効になっている場合、キャッシュできない、または共有メモリ属性が設定されている場合、アクセスはAXIMインタフェースを使用してメモリに直接実行されます。

AXI-MのL1キャッシュ・メモリ



STM32H7マイクロコントローラは、キャッシュRAMにECC保護機能を備えた16キロバイトの命令キャッシュおよびデータキャッシュを実装しています。STM32H7のL1キャッシュは、外部メモリなどの低速メモリの次のレベルから、頻繁に使用されるコードやデータへの高速アクセスを可能にします。

どちらのキャッシュも、データ・キャッシュには4ウェイ・セット・アソシアティブ方式、命令キャッシュには2ウェイ・セット・アソシアティブ方式を採用し、256ビット（32バイト）のライン長を使用しています。

セットとは、適切な境界線（2ウェイの場合は64バイト、4ウェイの場合は128バイト）に沿って並べられた連続した行のグループのことです。セットを使用すると、キャッシュされているかどうかを判断するために、アドレスをより速く検索することができます。キャッシュされた命令やデータは、AXIMインタフェースを使って外部メモリからフェッチされます。

ハードウェアの一貫性はサポートされておらず、ソフトウェアは使用前にキャッシュラインを無効にしてクリーニングすることでキャッシュメンテナンスを管理する必要があります。または、データの一貫性を維持する簡単な方法は、領域を「共有」としてマークすることです。これらの領域がD-キャッシュにキャッシュされるのを防ぎますが、すべてのアクセスが次のレベルのメモリに移動するため、パフォーマンスが低下します。

キャッシュRAM ECC

統合されたエラー・チェックと修正

- Cortex-M7キャッシュの場合、ECCはコアによって管理
  - 初期設定ではリセットから有効
  - キャッシュECCは、CM7_CACR.ECCENビットを変更することで無効にすることが可能
  - CM7_CACRは以下の場合にのみ変更する必要がある
    - 両方のキャッシュをオフにする
    - 変更後はキャッシュ全体を無効にする必要がある
- ECCメカニズムは、SECDEDアルゴリズムに基づく

life.augmented

STM32H7デバイスでは、キャッシュRAMのECC保護はSECDEDアルゴリズムを使用しています。キャッシュRAMの保護はコアで管理され、リセット後はデフォルトで有効になります。キャッシュの設定変更は、キャッシュがオフの時にのみ行ってください。ECC保護設定を変更した後は、キャッシュフラッシュを行う必要があります。

キャッシュRAM ECC

統合されたエラー･チェックと修正

- シングルエラーとダブルエラー検出、およびシングルエラー修正をサポート
  - 7ECCビットは命令キャッシュタグRAM、データキャッシュRAM、およびデータキャッシュタグRAMに使用される
  - 命令キャッシュRAMの64ビットワードには、8つのECCビットが使用される
- Cortex-M7プロセッサは、キャッシュ内で検出されたRAMエラーを、クリーン操作と無効化操作によって回復することができる
  - ダーティなデータ･キャッシュラインの場合、データの修正ができなければ、そのエラーは回復出来ない
  - ライトスルーポリシーを使用することで、データ損失を防ぐことができる(L2キャッシュは常にL1キャッシュと一貫性がある)

life.augmented

キャッシュRAMには、命令タグ、データタグ、およびデータに対して7つのECCビットが実装されています。命令キャッシュRAMには8ビットのECCコードが使用されます。
ECC保護機能により、Cortex-M7は、ランタイム中に検出されたRAMエラーから回復することができます。この回復には、キャッシュクリーンとインバリデートのメカニズムを使用します。ダーティ･データ･キャッシュ･ラインの場合、データを訂正できない場合、エラーは回復不能です。ライトスルーポリシーを使用すると、データの損失を回避できます(L2キャッシュは常にL1キャッシュと一貫性があります)。

キャッシュRAMのECC保護の概要

RAM種類	回復可能なエラー	回復不能エラー
データタグRAM	シングルビットエラーとして見られるエラー	マルチビットエラーとして見られるエラー
データキャッシュRAM	シングルビットエラーとして見られるエラー	ダーティラインでマルチビットエラーとして見られるエラー
命令タグRAM	RAMに保存されているタグや有効ビットに、シングルまたはダブルのエラーが発生した場合	無し
命令キャッシュRAM	RAMに格納されているデータに関するエラー	無し

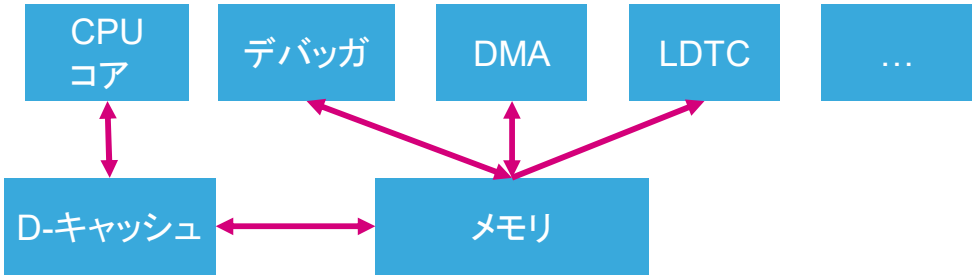
- 命令キャッシュがダーティになることはないので、キャッシュRAMのエラーは、キャッシュを無効にして命令を再試行することで常に回復可能

life.augmented

これは、キャッシュRAM ECC保護と回復可能なエラーの概要です。

データキャッシュ – コヒーレンシ

18



キャッシュ コヒーレンシの問題を解決するために

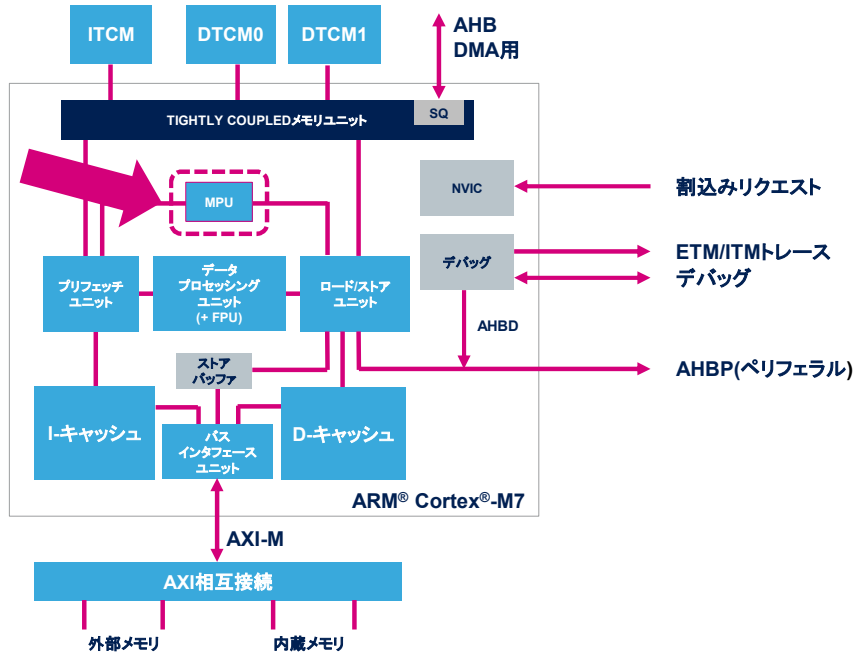
1. キャッシュを一切使用しない（領域を共有またはキャッシュ不可とする）
2. コントロールを別のマスタに渡すときにキャッシュを無効にする
3. 書き込み専用の場合は、ライトスルー･ポリシーを使用可能

life.augmented

異なるマスター（DMAなど）がコアと同じメモリバッファを共有する場合、キャッシュコヒーレンシーの問題を解決するために、以下の3つのソリューションがあります。

- 最初の解決策は、リージョンを「共有」とマークして、これらのリージョンがD-キャッシュにキャッシュされないようにすることです。
- 2つ目の解決策は、ソフトウェアがメモリバッファを渡したり制御したりする際に、キャッシュをクリーンまたは無効にすることです。キャッシュの削除や無効化を行うためのCMSISの機能があります。
- 3つ目の解決策は、CPUがデータ生産者である書き込み専用のメモリバッファにライトスルーポリシーを使用することです。

コア·アーキテクチャの概要



Cortex®-M7コアでは、メモリ保護ユニット(MPU)を使用して、キャッシュコントローラとAXIMインタフェースの動作を構成し、アクセスルールと個別のプロセスを適用します。

メモリ保護ユニットとキャッシュ

- MPUの属性設定がAXI-M/Cacheメモリ･システムの動作に影響を与える
- 特定の属性を持つ16個の独立したメモリ領域
  - キャッシュ･オン / オフ
  - キャッシュ･ポリシー
  - コード実行可能
  - 非特権モード･アクセス

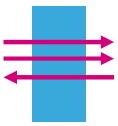
STM32H7マイクロコントローラのメモリ保護ユニット（MPU）は、16の独立したメモリ領域をサポートしており、以下のような独立した属性を設定できます。

- アクセス許可:特権モード/非特権モードでの読取り/書込み権限の場合
- 実行権限:命令のフェッチで禁止されている実行可能領域または領域
- キャッシュポリシー

1. スーパースカラ→1サイクルで2つの命令

2. 1サイクル･ブランチ

3. 遅いメモリを補うキャッシュ･システム



4. あらゆるアプリケーションに対応する
高いシステム帯域幅

life.augmented

STM32H7マイクロコントローラは、Cortex®-M7コアの新機能であるメモリ･インタフェース、低速メモリを補正するキャッシュ･システム、スーパースカラ･アーキテクチャなどの利点を活かして、良好な応答性を維持しながら、あらゆるアプリケーションに高い処理帯域幅を提供します。

参照

- 詳細については、以下のドキュメントをご参照ください
  - STM32F7 Series and STM32H7 Series Cortex®-M7 processor programming manual (PM0253)
  - Managing memory protection unit (MPU) in STM32 MCUs (AN4838)
  - Level 1 cache on STM32F7 series (AN4839)
  - STM32H7x3 system architecture and performance (AN4891)
  - ARM社のウェブサイトは以下のリンクからご覧いただけます。
    - http://www.arm.com/products/processors/cortex-m/cortex-m7-processor.php

life.augmented

詳細につきましては、本アプリケーションノートおよびCortex®-M7プログラミングマニュアル（ www.st.com ウェブサイト）をご参照ください。また、ARM社のWebサイトでは、Cortex®-M7コアの詳細情報をご覧いただけます。