Zynq-7000 All Programmable SoC アーキテクチャ ポーティング クイック スタート ガイド

UG1181 (v1.1.1) 2015 年 10 月 22 日

本資料は表記のバージョンの英語版を翻訳したもので、内容に相違が生じる場合には原文を優先します。資料によっては英語版の更新に対応していないものがあります。日本語版は参考用としてご使用の上、最新情報につきましては、必ず最新英語版をご参照ください。

XILINX
ALL PROGRAMMABLE™

改訂履歴

次の表に、この文書の改訂履歴を示します。

日付	バージョン	内容
2015 年 10 月 22 日	1.1.1	タイトルを更新。
2015 年 8 月 31 日	1.1	「TrustZone」セクションを更新。
2015 年 6 月 25 日	1.0	初版

目次

XILINX
ALL PROGRAMMABLE™

第 1 章

はじめに

この文書は、ザイリンクス Zynq®-7000 All Programmable (AP) SoC のカスタマーがエンベデッド ソフトウェアを非 ARM ベース プロセッサから ARM アーキテクチャに移植するのをサポートします。このガイドは、PowerPC®、Intel®、Renesas-SH、MIPS の各プロセッサから ARM プロセッサへの移植する際の参考資料となります (Zynq-7000 AP SoC は ARM® Cortex®-A9 デュアル コア プロセッサを内蔵)。

ARM Cortex-A9 は、低消費電力 (つまり発熱量に制限があり) かつコスト重視のデバイス向けの汎用プロセッサです。このプロセッサはリリースから数年を経ながら、今なおスマートフォン、デジタル テレビ、そして IoT を実現する一般向けおよび企業向けアプリケーションに広く採用されています。Cortex-A9 プロセッサは ARM がサポートするさまざまな技術を提供します。この Cortex-A9 プロセッサは、サイズと構成における拡張性の高さから、次のような幅広い製品に用いられています。

- 一般的なスマートフォン
- タブレット
- セットトップ ボックス
- ホーム メディア プレーヤー
- オートモーティブ インフォテインメント
- ルーター

Zynq-7000 AP SoC ファミリはザイリンクスの All Programmable SoC アーキテクチャで構成されています。この製品は、豊富な機能を備えたデュアルコア ARM Cortex-A9 MPCore™ ベースのプロセッシング システム (PS) とザイリンクスのプログラマブル ロジック (PL) を、最先端の高性能低消費電力 (HPL)、28nm、high-k メタル ゲート (HKMG) プロセス テクノロジで、1 つのデバイスに統合したものです。PS は ARM Cortex-A9 MPCore マルチコア プロセッサを中核として、オンチップ メモリ、外部メモリ インターフェイス、豊富な I/O ペリフェラルを備えています。

移植上の留意点

図 2-1 に Zynq®-7000 All Programmable (AP) SoC の全体図を示します。この図には、ARM® Cortex™-A9 プロセッサおよびそのコンポーネントが示されています。

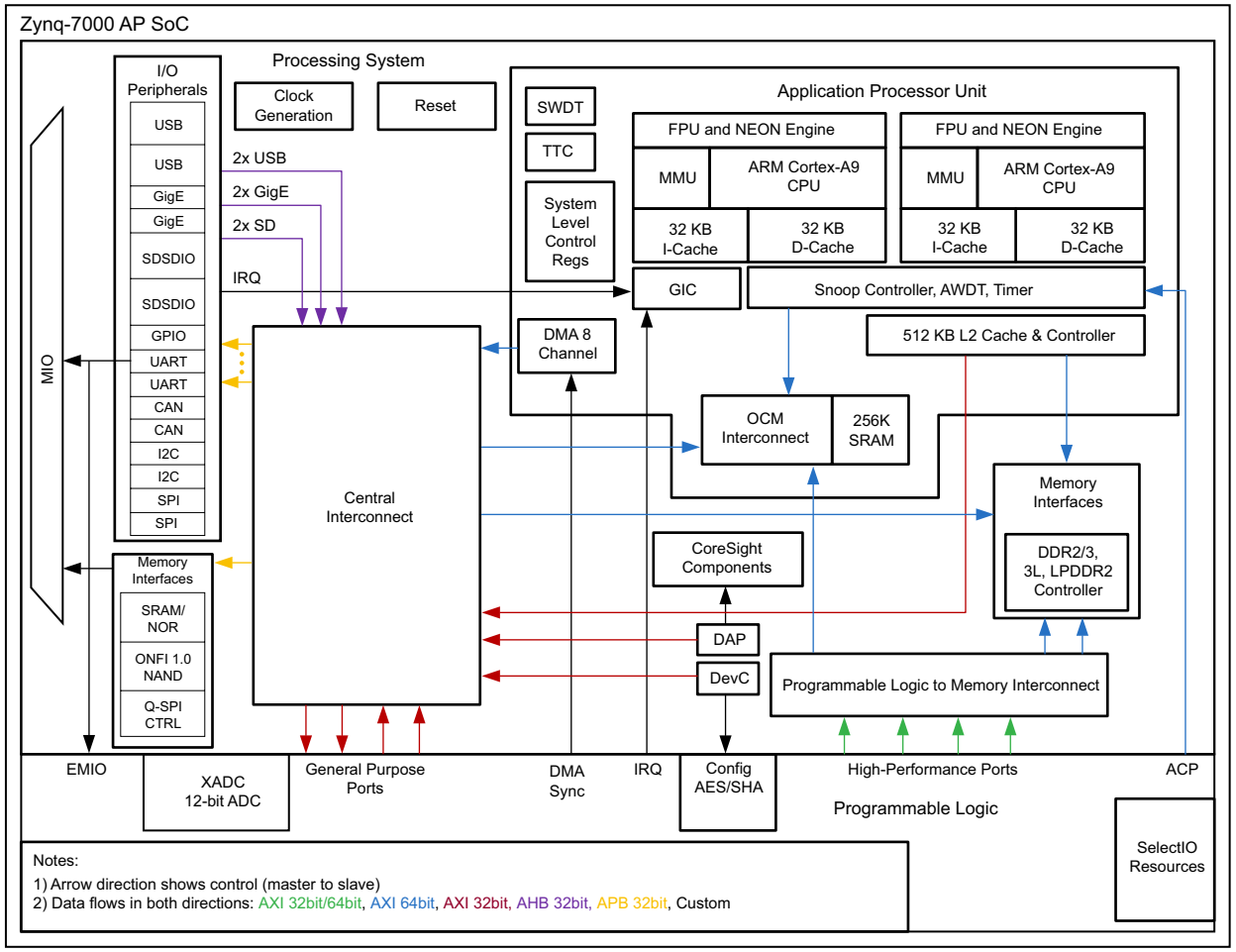

図 2-1 : Zynq-7000 All Programmable SoC の全体図

Zynq-7000 AP SoC が備える ARM Cortex-A9 プロセッサはデュアル コアです。各コアは別々の L1 キャッシュを内蔵しますが、L2 キャッシュは同じものを共有します。

ARM Cortex-A9 の機能

以降のセクションでは、ARM プロセッサの重要な機能について説明します。

CPU モード

CPU は常に 1 つのモードにしか入れませんが、外部イベント (割り込み) またはプログラム制御でそれを切り替えることができます。

- ユーザー モード : 唯一の非特権モード
- FIQ モード : プロセッサが FIQ 割り込みを受け取ると必ず入る特権モード
- IRQ モード : プロセッサが IRQ 割り込みを受け取ると必ず入る特権モード
- スーパーバイザー (svc) モード : CPU がリセットされるか SVC 命令が実行されると必ず入る特権モード
- アボート モード : プリフェッチ アボートまたはデータ アボート例外が発生すると必ず入る特権モード
- 未定義モード : 未定義命令例外が発生すると必ず入る特権モード
- システム モード (ARMv4 以降) : 例外が発生しても入らない唯一の特権モード。このモードには、カレント プログラム ステータス レジスタ (CPSR) のモード ビットに明示的に書き込む命令を実行することによってのみ切り替わります。
- モニター モード (ARMv6 および ARMv7 セキュリティ拡張、ARMv8 EL3) : ARM コアの TrustZone® 拡張をサポートするために用意されています。
- Hyp モード (ARMv7 仮想化拡張、ARMv8 EL2) : セキュアでない CPU 動作の仮想化要件をサポートするハイパーバイザー モードです。

TrustZone

Cortex-A9 アーキテクチャは、TrustZone テクノロジとして提供されているセキュリティ拡張を備えます。これはハードウェア ベースのアクセス制御が可能な 2 個の仮想プロセッサを提供するため、SoC に別の専用セキュリティコアを追加する必要がなくなり、セキュリティが低コストで実現します。この機能によって、アプリケーション コアは (機能ドメインを意味する別の名前と混同しないように) ワールドと呼ぶ 2 つの状態を切り換えることで、信頼性が高いワールドから信頼性が低いワールドに情報が漏れるのを防ぎます。一般に、ワールドの切り換えはプロセッサのほかの機能に影響されないため、同じコアを使用していても各ワールドはほかのワールドから独立して動作できます。メモリおよびペリフェラルは、コアの動作ワールドを認識し、これを用いてデバイス上の機密とコードへのアクセスを制御できます。

TrustZone テクノロジの代表的な用途として、信頼性が低いワールドで高機能なオペレーティング システムを実行し、より信頼性が高いワールドで小規模なセキュリティ専用コードを実行します。これによって、ARM ベース デバイスにおけるメディア使用を制御するためのデジタル著作権管理 (DRM) をさらに厳密にでき、デバイスの無許可使用も防止できます。ただし実際は、ザイリンクス以外の多くのシリコン ベンダーは、ARM TrustZone ハードウェアおよび関連ソフトウェアの具体的な実装の詳細を機密情報として扱っています。そのため、そのようなサードパーティのソリューションでどのようなサービスが提供されているかは必ずしも明確ではありません。ソフトウェア スタックの実装を解析することで、セキュア ワールドの実装に関する詳細を得ることが可能です。

一般的なアプリケーションの例

TrustZone に基づく次のような一般的なアプリケーションが ARM の TrustZone のウェブサイト [参照 3] に示されており、実際の TrustZone の使用方法をわかりやすく示しています。

- モバイル ペイメントおよびモバイル バンキングにおいてユーザー認証を強化するためのセキュリティ保護された PIN エントリ
- トロイの木馬、フィッシング詐欺、および APT (Advanced Persistent Threat) 攻撃からの保護
- 高価値なメディアの配信および利用の有効化 (DRM)
- BYOD (私的デバイス活用) デバイスにおける個人とアプリケーションの分離
- ソフトウェア ライセンス管理
- ロイヤルティベース アプリケーション
- クラウドベース ドキュメントのアクセス制御

Thumb-2 命令セット

プロセッサは Thumb ステートで、ARM 命令セットのサブセットに対応するコンパクトな 16 ビット エンコードである Thumb 命令セットを実行します。ほとんどの Thumb 命令は、通常の ARM 命令に直接対応します。その高いコード密度は、命令オペランドのいくつかを明示することと、ARM 命令セット ステートで実行する ARM 命令と比べて数を制限することで実現しています。

Thumb では、16 ビット オペコードの機能に制限があります。たとえば、条件付きは分岐命令のみであり、オペコードの多くは CPU の汎用レジスタ全体の半数にしかアクセスできません。オペコードが短いことで、一部の演算では追加の命令が必要になりますが、全体としてはコード密度が向上します。メモリのポートまたはバスの幅が 32 ビット未満に制限されている場合、32 ビットの ARM コードに比べて短い Thumb オペコードを使うことで、メモリ帯域幅の上限を超えてプロセッサに書き込むべきプログラム コードが少なくて済むため、性能はむしろ向上します。

Thumb-2 は、制限がある Thumb の 16 ビット命令セットに 32 ビット命令を追加することで命令セットを拡張し、可変長命令セットとしています。Thumb-2 の目的は、Thumb と同等のコード密度を維持しながら、32 ビット メモリ上の ARM 命令セットと同等の性能を実現することです。

SIMD サポート (NEON)

アドバンスド SIMD 拡張である NEON MPE (Media Processing Engine) は 64 ビットと 128 ビットを組み合わせた SIMD 命令セットであり、メディアおよび信号処理アプリケーションにおける標準的な高速化手法です。NEON は Zynq-7000 AP SoC が内蔵する Cortex-A9 の機能であり、包括的な命令セット、独立したレジスタ ファイル、独立した実行ハードウェアを特徴としています。NEON は、8/16/32/64 ビット整数および単精度 (32 ビット) 浮動小数点データ、および単一命令複数データ (SIMD) 演算をサポートし、オーディオおよびビデオ処理、さらにはグラフィックスおよびゲーム処理をサポートします。NEON では、SIMD は最大 16 の演算を同時に実行します。NEON ハードウェアは、VFP で使用したものと同じ浮動小数点レジスタを共有します。NEON は 128 ビットを一度に実行できますが、ARM Cortex-A9 プロセッサは 128 ビット ベクターをサポートしているにもかかわらず、一度に実行するのは 64 ビットです。NEON は、10MHz で動作する CPU 上で MP3 オーディオ復号化を実行でき、また GSM の適応マルチ レート (AMR) 音声コーデックを 13MHz 以下で実行できます。

ベクター浮動小数点ユニット (VFPU) - 単精度、倍精度

ARM アーキテクチャを拡張する浮動小数点コプロセッサです。

キャッシュ

- L1 データ キャッシュ 32KB、L1 命令キャッシュ 32KB
  - それぞれ 4 ウェイ セット アソシエイティブ
- L2 512KB キャッシュ (データと命令に共通)
  - 8 ウェイ セット アソシエイティブ

MMU

メモリ管理ユニット (MMU) は L1 および L2 メモリ システムと連携し、仮想アドレスを物理アドレスに変換します。また、外部メモリとのアクセスも制御します。

- 4KB、64KB、1MB、16MB をサポートするページ テーブル エントリ
- 16 個のドメイン
- グローバルおよびアプリケーション固有の識別子を備えるため、コンテキスト スイッチ用の変換ルックアサイド バッファー (TLB) フラッシュの要件を削除
- アクセス許可チェック機能を拡張

OCM

オンチップ メモリ (OCM) モジュールは 256KB RAM を内蔵します。OCM モジュールは、2 個の 64 ビット AXI スレーブ インターフェイス ポートと 1 個の CPU 専用アクセラレータ コヒーレンシ ポート (ACP) によるアクセスをサポートします。このアクセスは、アプリケーション処理装置 (APU)、スヌープ制御装置 (SCU)、プロセッシング システム (PS) とプログラマブル ロジック (PL) 内の各バス マスターが共有する全装置を通じてなされます。bootROM メモリはブート プロセスでのみ使用され、ユーザーには見えません。

OCM に割り当てるアドレス範囲は、アドレス マップの先頭または末尾の 256KB に設定できるため、ARM のロー/ハイ ベクター例外を柔軟に処理できます。さらに、CPU および ACP AXI インターフェイスは、下位 1MB のアドレス範囲のアクセスを、SCU アドレス フィルタリング機能を用いて DDR に転送できます。

割り込み

- ARM のグローバル割り込みコントローラー (GIC) が割り込みを制御
- 3 個のウォッチドッグ タイマー (WDT) — コア 0 用、コア 1 用、システム用に各 1 個
- 各コアは、プライベート ペリフェラル割り込み (PPI) と共有ペリフェラル割り込み (SPI) をそれぞれ複数サポート
- 割り込みの優先度付けが可能
- 割り込みまたはイベント信号の発生を待っている間、CPU は待機ステート (WFI) に入ることが可能

システム制御コプロセッサ (CP15)

システム制御コプロセッサ (CP15) は、プロセッサの機能のステータス情報を制御および提供します。システム制御コプロセッサの主な機能は次の通りです。

- 全体のシステム制御および設定
- MMU の設定および管理
- キャッシュの設定および管理
- システム性能の監視

タイマー

Cortex-A9 プロセッサの各コアは、それぞれ専用の 32 ビット プライベート タイマと 32 ビット ウォッチドッグ タイマーを持っています。また、これら 2 つのプロセッサ コアで 1 個の 64 ビット グローバル タイマーを共有しています。

これらのタイマーはすべて CPU の 1/2 の周波数 (CPU_3x2x) で常にクロッキングされます。

システム レベルでは、1 個の 24 ビット ウォッチドッグ タイマーと 2 個の 16 ビット トリプル タイマー /カウンターを備えます。システム ウォッチドッグ タイマーは、CPU の 1/4 または 1/6 の周波数 (CPU_1x) でクロッキングするか、MIO ピンまたは PL からの外部信号でクロッキングできます。

2 個のトリプル タイマー/カウンターは、常に CPU の 1/4 または 1/6 の周波数 (CPU_1x) でクロッキングされ、MIO ピンまたは PL からの信号のパルス幅を計測します。

命令セット

ARM は RISC (縮小命令セット コンピューター) プロセッサです。その命令セットには次のような特徴があります。

- ロード/ストア アーキテクチャ
- ハーフ ワードとシングル ワードのロード/ストア命令のために非整列アクセスをサポート (アトミック性を保証しないなどの制限あり)
- 均等な 16 × 32 ビット レジスタ ファイル (プログラム カウンター、スタック ポインター、リンク レジスタを含む)
- デコードとパイプライン処理が容易な 32 ビット固定長命令 (コード密度の観点では不利)
- 16 ビット命令でコード密度を高める Thumb 命令セット
- ほとんどの命令を 1 クロック サイクルで実行

簡潔な設計を補うため、インテル アーキテクチャと比べて、次のような設計上の特徴があります。

- ほとんどの命令が備える条件付き実行が、分岐プレディクタに代わって分岐のオーバーヘッドを低減
- 演算命令は必要な場合のみ条件コードを変更
- 32 ビット バレル シフターは、性能を低下させることなく、ほとんどの演算命令とアドレス計算で使用可能
- 強力なインデックス付きアドレッシング モード
- リンク レジスタが高速リーフ関数呼び出しをサポート
- レジスタ バンク切り換えを備え、シンプルかつ高速で 2 つの優先度レベルを持つ割り込みサブシステム

レジスタ セット

レジスタ セットを次の表に示します。

各 CPU モードとレジスタ						
usr	sys	svc	abt	und	irq	fiq
R0						
R1						
R2						
R3						
R4						
R5						
R6						
R7						
R8						R8_fiq
R9						R9_fiq
R10						R10_fiq
R11						R11_fiq
R12						R12_fiq
R13		R13_svc	R13_abt	R13_und	R13_irq	R13_fiq
R14		R14_svc	R14_abt	R14_und	R14_irq	R14_fiq
R15						
CPSR						
		SPSR_svc	SPSR_abt	SPSR_und	SPSR_irq	SPSR_fiq

レジスタ R0 ～ R7 は CPU モードに関わらず同じで、バンク化されることはありません。

R13 と R14 は、システム モードを除くすべての特権 CPU モードでバンク化されます。つまり、例外が原因で入るモードは、それぞれ専用の R13 と R14 があります。一般にこれらのレジスタは、それぞれスタック ポインターと関数呼び出しからの戻りアドレスを持ちます。

エイリアス

- R13 は SP (スタック ポインター) とも呼ぶ
- R14 は LR (リンク レジスタ) とも呼ぶ
- R15 は PC (プログラム カウンター) とも呼ぶ

カレント プログラム ステータス レジスタ (CPSR) は次の 32 ビットからなります。

- M (ビット 0 ～ 4) はプロセッサ モード ビット
- T (ビット 5) は Thumb ステート ビット
- F (ビット 6) は FIQ ディスエーブル ビット
- I (ビット 7) は IRQ ディスエーブル ビット
- A (ビット 8) は不正確データ アボート ディスエーブル ビット
- E (ビット 9) はデータ エンディアン ビット
- IT (ビット 10 ～ 15、25、26) は if-then ステート ビット
- GE (ビット 16 ～ 19) は greater-than-or-equal-to ビット
- DNM (ビット 20 ～ 23) は書き換え禁止ビット
- J (ビット 24) は Java ステート ビット
- Q (ビット 27) はスティッキー オーバーフロー ビット
- V (ビット 28) はオーバーフロー ビット
- C (ビット 29) はキャリー /ボロー /拡張ビット
- Z (ビット 30) はゼロ ビット
- N (ビット 31) は negative/less than ビット

ほかのアーキテクチャから ARM にソフトウェアを移植する場合、必須事項と任意の留意点があります。次の章ではこれらについて説明します。

アーキテクチャ間の機能比較

アーキテクチャの比較

次の表で各アーキテクチャを比較します。

機能	ARMv7	PowerPC	MIPS	Renesas-SH	x86
エンディアン	ビッグ リトル	ビッグ リトル	ビッグ リトル	ビッグ リトル	リトル
ビット	32	64 (32→64) 32 ビット サブセットを備える 64 ビット アーキテクチャ	64 (32→64) 32 ビット サブセットを備える 64 ビット アーキテクチャ	32	16、32、64
データ ビット	3	3	1、2、3	2	2 (整数) 3 (AVX-512)
オペランドの移動	レジスタ - レジスタ	レジスタ - レジスタ	レジスタ - レジスタ	レジスタ - レジスタ レジスタ - メモリ	レジスタ - メモリ
方式	RISC	RISC	RISC	RISC	CISC
レジスタ	16 (PC、SP を含む)	32	32 x 4 バンク (ゼロを含む)	16	6 (16 ビット) 8 (32 ビット) 16 (64 ビット)
命令ビット	ARM : 32 ビット Thumb : 16 ビット Thumb2 : 16 ～ 32 ビット	32 ビット	32 ビット	16 ～ 32 ビット	可変
拡張	NEON VFP TrustZone Jazelle LPAE	AltiVec、APU、VSX、Cell	MDMX MIPS-3D	なし	x87、IA-32、MMX、3DNow!、SSE、SSE2、PAE、x86-64、SSE3、SSE4、SSE5、AVX、AES、FMA

関数呼び出しの表記規則

通常、ソース コードは C 言語などの高級言語で記述されています。このため、ツール チェーンを適切に変更することで別のアーキテクチャへの移植が可能です。移植先のアーキテクチャに適して機械語を生成は、コンパイラが担います。

ただし、場合によっては各種の最適化のためにコードを微調整が必要です。この場合、コードの生成方法、レジスタの用途、呼び出しシーケンスなどの詳細を開発者が理解している必要があります。

次の表で、各アーキテクチャの関数呼び出しの表記規則を比較します。

表記規則	ARM	PowerPC	MIPS	Renesas-SH	x86
関数の パラメーター	R0 ～ R3	R3 ～ R10 後続の引数 - スタック	先頭の 4 つの引数 - レジスタ ($a0 ～ $a3) 後続の引数 - スタック	R4 ～ R7	スタック (ESS、ESP)
パラメーターの 順序	左から右	左から右	左から右	左から右	右から左
標準戻り値	レジスタ (R0)	R3	1 個のレジスタ ($v0)	R0	EIP
ロング (複素数) 戻り値	レジスタ (R0 ～ R4)	R3 ～ R4	2 個のレジスタ ($v0、$v1)	R0	EIP
スタック ポインター	R13	R1	レジスタ ($sp)	R15	ESS、ESP
戻りアドレス	R14 (リンク レジスタ)	LR (スタックに保存)	レジスタ ($ra)	レジスタ (PR)	EIP
ローカル変数の 配置方法	スタック	スタック	スタック	スタック	スタック
予約されていない レジスタ	R4 ～ R8	R0、R2 ～ R10、R12 FPR0 ～ FPR13 LR、CTR、XER、 CR0 ～ CR7	$0 ～ $15 $24、$25	R1 ～ R7	すべてのレジスタ
揮発性レジスタ	R4 ～ R11	R3 ～ R12	$8 ～ $15 $24、$25	R1 ～ R3	すべてのレジスタ

関数呼び出し後の設定とクリーン アップ

RISC プロセッサ (ARM、MIPS、PowerPC、Renesas-SH) は ABI/EABI[1] 規格の関数呼び出しシーケンスに従いますが、わずかな相違があります。表 3-1 に、各アーキテクチャの呼び出しプロシージャを示します。

表 3-1 : 各アーキテクチャの関数呼び出しプロシージャ

アーキテクチャ	プロローグ (関数を呼び出す準備)	エピローグ (関数呼び出しを終える準備)
ARM	r4 ～ r11 をスタックにプッシュする。 r14 の戻りアドレスをスタックにプッシュする。 引数 (r0 ～ r3) をローカルのスクラッチ レジスタ (r4 ～ r11) にコピーする。 その他のローカル変数を残りのローカル スクラッチ レジスタ (r4 ～ r11) に割り当てる。 必要に応じて、BL を使ってその他のサブルーチンを呼び出す。	結果を r0 に入れる。 r4 ～ r11 をスタックからプルする。 プログラム カウンター r15 への戻りアドレスをプルする。
MIPS	スタック フレームの領域を予約する。スタック フレームは、次に示す最大 5 つのセクションを持つことができる。 • 引数セクション • 保存済みレジスタ セクション • 戻りアドレス セクション • パディング セクション • ローカル データ ストレージ セクション 仮想フレーム ポインターを設定する。仮想フレーム ポインタは、フレーム サイズに追加される sp($29) である。 保存済みの各汎用レジスタのビットマスクにビットを設定する。 保存の必要があるレジスタを保存する。 命令ポインターを関数の先頭に設定する。 関数の実行を開始する。	結果を $v0 に配置する。 プロローグで保存したレジスタごとに復元命令を発行する。 プロシージャから戻る。
PowerPC	呼び出された関数自身が、スタック フレームを割り当て、スタック内の 16 バイト アラインメントを保つ必要がある。 新規スタック フレームの位置を示すためにスタック ポインターをデクリメントし、スタック ポインターの前の値を自身のリンケージ エリアに書き込み、呼び出しから戻った時にスタックが必ず元の状態に戻るようにする。 非揮発性の汎用浮動小数点レジスタをすべて保存済みレジスタ エリアに保存する。 リンク レジスタと条件レジスタの値を、必要に応じてコーラーのリンケージ エリアに保存する。 スタック フレームには次の順で 4 つのセクションがある。 • パラメーター エリア • リンケージ エリア • 保存済みレジスタ • ローカル変数 関数コードを実行する。	スタック フレームに保存した非揮発性の汎用浮動小数点レジスタを復元する。 非揮発性レジスタは、スタック ポインターが更新される前に、新規スタック フレームに保存される。これは、新規スタック フレームが正常に割り当てられる、レッド ゾーンとも呼ばれるスタック ポインターの下の領域に非揮発性レジスタがフィットする場合のみ。レッド ゾーンは、非揮発性の汎用浮動小数点レジスタをすべて保存するのに十分な大きさを持つように定義されている。ただし、非揮発性ベクター レジスタは含まない。 リンケージ エリアに保存した、条件レジスタとリンク レジスタの値を復元する。 スタック ポインターを直前の値に復元する。 リンク レジスタに保存されたアドレスを用いて、呼び出しルーチンに制御を戻す。

1. アプリケーション バイナリ インターフェイス/エンベデッド アプリケーション バイナリ インターフェイス

表 3-1 : 各アーキテクチャの関数呼び出しプロシージャ (続き)

アーキテクチャ	プロローグ (関数を呼び出す準備)	エピローグ (関数呼び出しを終える準備)
x86	古いベース ポインターをスタックの上にプッシュする。 新しいベース ポインターの値を得る。この値は次のステップで設定され常にこの位置を指す。 新規スタック フレームが直前のスタック フレームの上に作成される (すなわち、前のスタック フレームの先頭が新規スタック フレームの末尾になる) ように、スタック ポインター値 (保存されたベース ポインターと直前のスタック フレームの先頭を示す) をベース ポインターに割り当てる。 スタック ポインターを下げて、変数 (関数のローカル変数) を入れる場所を開ける。 関数のコードを実行する。	スタック ポインターを、プロローグの前の値に復元できるように、現在のベース (またはフレーム) ポインター値で置き換える。 ベース ポインターを、プロローグの前の値に復元できるように、スタックからポップする。 前のフレームのプログラム カウンターをスタックからポップし、ジャンプすることで、関数呼び出しに戻る。
Renesas-SH	入力引数値を、R4 ～ R7 と FR4 ～ FR11 から引数のホーム ロケーションに保存する命令 (0 個以上) のシーケンス。 保存するすべてのパーマネント レジスタと戻りアドレス (PR) をプッシュする命令 (0 個以上) のシーケンス。 フレーム ポインターを設定する命令 (1 個以上) のシーケンス。 4 バイト境界に位置合わせしたオフセットを R15 から減算することで、ローカル変数、コンパイラ生成一時ファイル、引数ビルドのエリアに、残りのスタック フレーム領域を割り当てる命令 (0 個以上) のシーケンス。 関数コードを実行する。	フレーム ポインターをインクリメントする 1 個の追加命令。 命令のデスティネーション オペランドまたはインクリメント後のメモリ アドレス オペランドの中を参照し、R15 を変更する複数の命令のシーケンス。

次のセクションでは、各アーキテクチャの留意点を説明します。

割り込みモデル

これらのアーキテクチャの割り込みソースは、特定の規格には従っているわけではありません。利用可能なリソースと対象とするアプリケーションに基づいてそれぞれ例外を定義しています。

ただし、例外は大きく 2 つのグループ、つまりソフトウェア例外と外部イベントに分けられます。次の表に、アーキテクチャ別にこれら 2 つのグループの割り込みを示します。

例外のタイプ	ARM	MIPS	PowerPC	x86	Renesas-SH
命令による例外	SWI 未定義命令 プリフェッチ アボート データ アボート	SYS OV TR	クリティカル入力 マシン チェック システム呼び出し	除算エラー ブレーク ポイント 不正オペコード セグメント不在 スタック セグメント違反 一般保護違反 ページ違反 マシン チェック	TRAP、 UBRKAFTER
外部ワールドによる例外	リセット IRQ FIQ	リセット NMI 割り込み AdEL AdES TLBL TLBS ICache エラー DCache エラー	データ ストレージ 命令ストレージ 外部アラインメント プログラム 浮動小数点 PITimer FITimer WDTimer TLB ミス デバッグ	NMI ユーザー定義割り込み	リセット : POWERON、 MANRESET、 HUDIReset、 ITLBMULTIHIT、 OLTBMULTIHIT 全般 : UBRKBEFORE、 IADDERR、 ITLBMISS、 EXECPROT、 RESINST、 ILLSLOT、 FPUDIS、 SLOTFPUDIS、 RADDERR、 WADDERR、 READPROT、 FPUEXC、 FIRSTWRITE 割り込み : NMI、IRLINT、 PERIPHINT

アドレス マップ

Zynq-7000 AP SoC 内の ARM Cortex-A9 のアドレス マップ

表 3-2 にシステム レベルのアドレス マップを示します。ARM システムのアドレス マップの詳細は、『Zynq-7000 All Programmable SoC テクニカル リファレンス マニュアル』(UG585) [参照 1] を参照してください。

表 3-2 : システム レベルのアドレス マップ

アドレス範囲	CPU と ACP	AXI_HP	その他のバス マスター	説明
0000_0000 ～ 0003_FFFF	OCM	OCM	OCM	アドレスは SCU でフィルターされず、OCM は下位に割り当てる
	DDR	OCM	OCM	アドレスは SCU でフィルターされ、OCM は下位に割り当てる
	DDR			アドレスは SCU でフィルターされ、OCM は下位に割り当てない
				アドレスは SCU でフィルターされず、OCM は下位に割り当てない
0004_0000 ～ 0007_FFFF	DDR			アドレスは SCU でフィルターされる
				アドレスは SCU でフィルターされない
0008_0000 ～ 000F_FFFF	DDR	DDR	DDR	アドレスは SCU でフィルターされる
		DDR	DDR	アドレスは SCU でフィルターされない
0010_0000 ～ 3FFF_FFFF	DDR	DDR	DDR	全インターコネクト マスターにアクセス可
4000_0000 ～ 7FFF_FFFF	PL		PL	PL への汎用ポート #0、M_AXI_GP0
8000_0000 ～ BFFF_FFFF	PL		PL	PL への汎用ポート #1、M_AXI_GP1
E000_0000 ～ E02F_FFFF	IOP		IOP	I/O ペリフェラル レジスタ
E100_0000 ～ E5FF_FFFF	SMC		SMC	SMC メモリ
F800_0000 ～ F800_0BFF	SLCR		SLCR	SLCR レジスタ
F800_1000 ～ F880_FFFF	PS		PS	PS システム レジスタ
F890_0000 ～ F8F0_2FFF	CPU			CPU プライベート レジスタ
FC00_0000 ～ FDFF_FFFF	クワッド SPI		クワッド SPI	リニア モード用クワッド SPI リニア アドレス
FFFC_0000 ～ FFFF_FFFF	OCM	OCM	OCM	OCM は上位に割り当てる
				OCM は上位に割り当てない

MIPS

図 3-1 に MIPS のメモリ マップを示します。

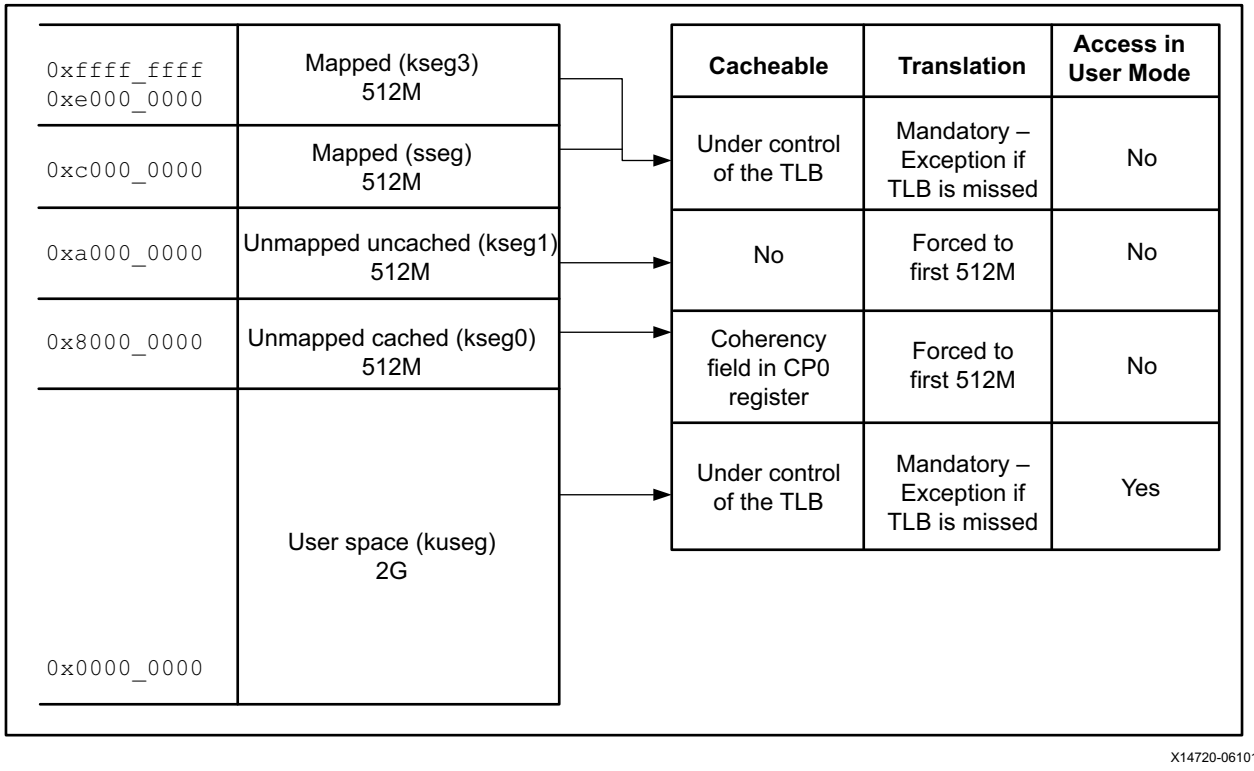

図 3-1 : MIPS のメモリ マップ

PowerPC

図 3-2 に MMU の概略ブロック図を示します。

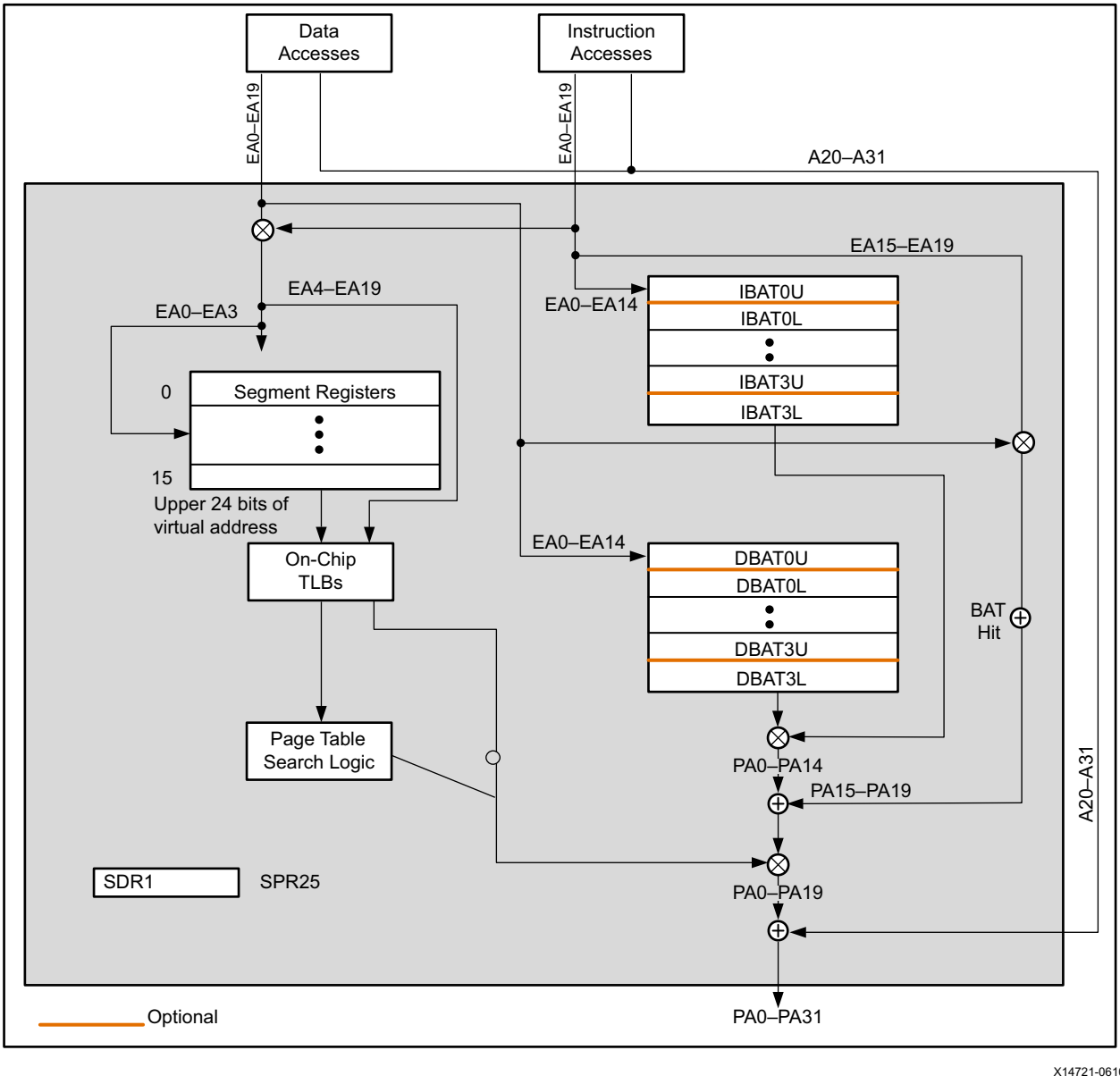

図 3-2 : MMU の概略ブロック図

x86

図 3-3 に x86 のアドレス機構を示します。

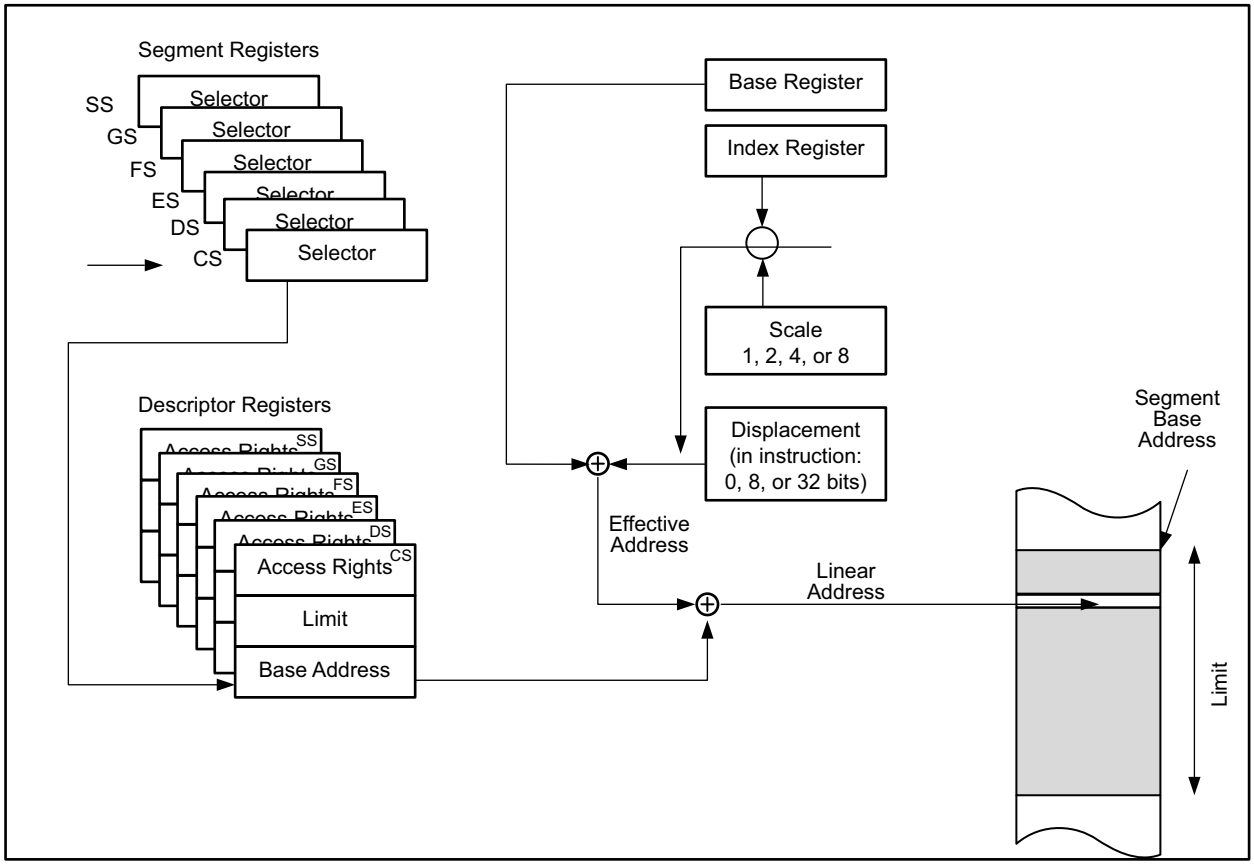

図 3-3 : x86 のアドレス機構

Renesas-SH

図 3-4 に Renesas-SH のメモリ マップを示します。

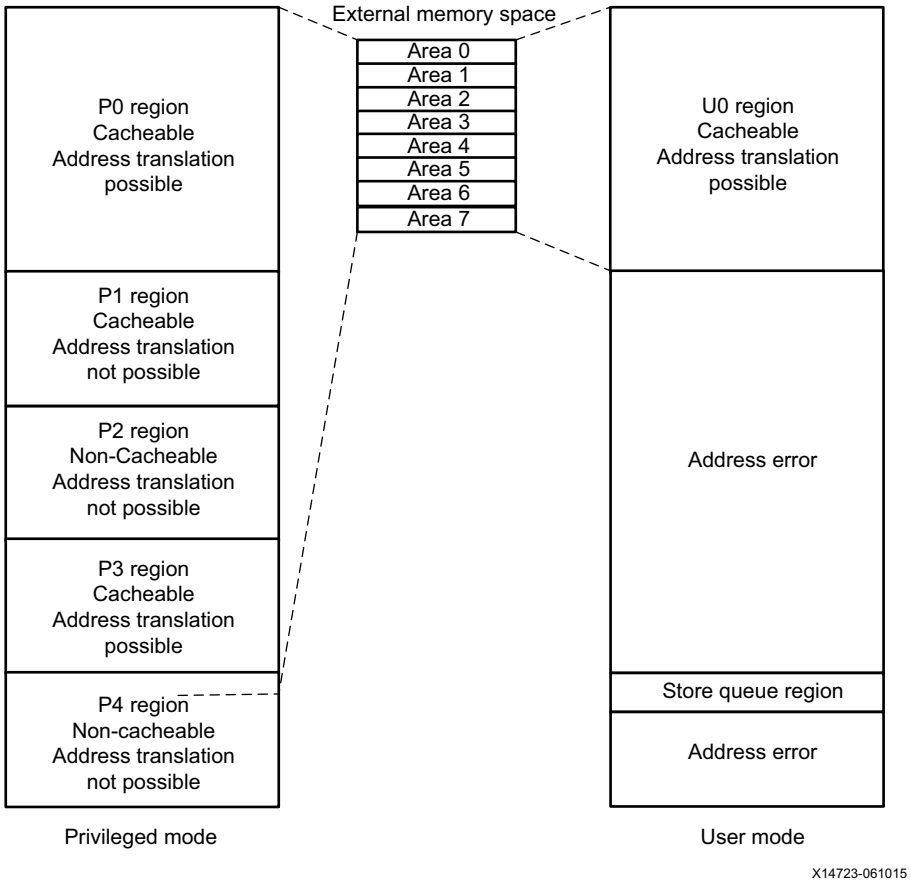

図 3-4 : Renesas-SH のメモリ マップ

レジスタ セット

次の表に、各プロセッサ (ARM、MIPS、PowerPC、x86、Renesas-SH) のレジスタの主な機能を示します。

レジスタ タイプ	ARM	MIPS	PowerPC	x86	Renesas-SH
汎用レジスタ	R0 ～ R12	$8 ～ $15、$24、$25 : 一時的 (保存されない) $16 ～ $23 : 一時的 (保存される)	ユーザー モデル (UISA) GPR0 ～ 31 FPR0 ～ 31 CR FPSCR XER LR CTR TBU/TBL	汎用 : EAX、EBX、ECX、EDX セグメント : CS、DS、ES、FS、GS、SS インデックス : ESI、EDI、EBP、EIP、ESP インジケーター : EFLAGS	汎用、バンク化される : R0 ～ R7 汎用、バンク化されない : R8 ～ R15 浮動、バンク化される : FR0-15、XF0-15
特定用途レジスタ	R13 : スタック ポインター R14 : リンク レジスタ R15 : プログラム カウンター	$0 : ゼロ固定 $1 : アセンブラー用に予約済み $2、$3 : 関数の戻り値 $4 ～ $7 : 関数の引数 $26、$27 : OS 用に予約済み $28 : グローバル ポインター $29 : スタック ポインター $30 : フレーム ポインター (保存される) $31 : リンク レジスタ	なし	なし	制御レジスタ : SR、GBR、SSR、SPC、SGR、DBR、VBR システム レジスタ : MACH、MACL、PR、FUPL、PC、FPSCR
特定用途レジスタ	システム コプロセッサ (CP15)	$SR、$PC、GBR、VBR、SGR、DBR、MACL、MACH、PR、PC、FPUL、FPSCR、TRA、EXPEVT、INTEVT、PTEH、PTEL、TTB、TEA、MMUCR、PASCR、IRMCR、CCR、QACR0/1、RAMCR、LSA0/1、LDA0/1、CPUPOM、PVR	スーパバイザー モード (OEA) MSR PVR SDR1 ASR DAR DSISR SRR0 ～ 1 SPRG0 ～ 3 FPECR DABR DEC EAR PIR	制御 : CR0 ～ 4 デバッグ : DR0 ～ 7 テスト : TR0 ～ 7 GDTR、IDTR、LDTR、TR	なし

詳細なポーティング ガイド : MIPS、PowerPC、Intel、Renesas

ARM Cortex-A9 ベースの SoC であるザイリンクス Zynq-7000 への移行については、『UltraFast エンベデッド デザイン設計手法ガイド』(UG1046) [参照 2] を参照してください。この文書は次について詳細に説明しています。

- システム レベルの留意点
- ハードウェア デザインの留意点
- ソフトウェア デザインの留意点
- ハードウェア デザイン フロー
- ソフトウェア デザイン フロー
- デバッグ手法
- SDSoC 環境

この設計手法ガイドは、ARM 以外のプラットフォームから ARM プラットフォームへのユーザー アプリケーションの移行をサポートします。

MIPS から ARM への移行

詳細説明は、ARM のサイトの『MIPS から ARM への移行』を参照してください。

PowerPC から ARM への移行

詳細説明は、ARM のサイトの『Power アーキテクチャから ARM への移行』を参照してください。

IA-32 から ARM への移行

詳細説明は、ARM のサイトの『IA-32 から ARM への移行』を参照してください。

Renesas-SH から ARM への移行

詳細説明は、ARM のサイトの『SH-4A から Cortex-A への移行』(AN314) を参照してください。

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。

ソリューション センター

デバイス、ツール、IP のサポートについては、ザイリンクス ソリューション センターを参照してください。デザイン アシスタント、デザイン アドバイザリ、トラブルシュートのヒントなどが含まれます。

参考資料

注記 : 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. 『Zynq-7000 All Programmable SoC テクニカル リファレンス マニュアル』(UG585 : 英語版、日本語版)
2. 『UltraFast エンベデッド デザイン設計手法ガイド』(UG1046 : 英語版、日本語版)
3. ARM TrustZone Technology (www.arm.com/products/processors/technologies/trustzone/index.php)

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」、および全て受領者の責任で (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)、全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、http://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下さい。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。http://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下さい。

自動車用のアプリケーションの免責条項

ザイリンクスの製品は、フェイルセーフとして設計されたり意図されてはおらず、また、フェイルセーフの動作を要求するアプリケーション (具体的には、(I) エアバッグの展開、(II) 車のコントロール (フェイルセーフまたは余剰性の機能 (余剰性を実行するためのザイリンクスの装置にソフトウェアを使用することは含まれません) および操作者がミスをした際の警告信号がある場合を除きます)、(III) 死亡や身体傷害を導く使用、に関するアプリケーション) を使用するために設計されたり意図されたりもしていません。顧客は、そのよう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を単独で負います。

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。