ソフトウェアパーティショニング設計の実践アプローチ

（大切な安全機構を守るための設計技術）

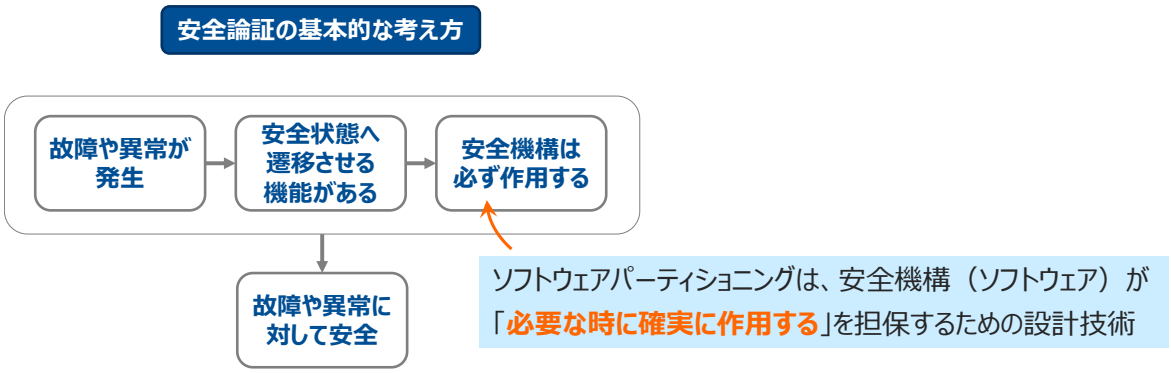
混載が常態化した車載ソフトウェア

自動車の E/E アーキテクチャのドメイン統合やセントラル ECU 化が進んだ結果、車載ソフトウェアの設計にも大きな影響が出てきています。例えば EV、ADAS、IVI、OTA サービス といった多様なアプリが統合された結果、異なるドメインのソフトウェアが 1 チップにひしめき合う構成として量産化されています。これらのアプリ統合によるソフトウェア規模の伸びは演算リソースの増加を上回り、異なる ASIL を持つソフトウェア同士が同一コアでキャッシュやバスを取り合う状況に至っています。この"混載前提"のソフトウェア環境でも、安全機構を確実に作動させ、自動車の安全確保に貢献する設計手法がソフトウェアパーティショニングです。

ソフトウェアパーティショニングは、自動車業界で機能安全対応が始まった時から話題になっていた設計手法の 1 つですが、上記の背景から、近年再び注目される機会が増えています。異なる ASIL 群のソフトウェアの混載環境を分離するためのニーズに加え、ソフトウェア同士の非干渉性を活用した、肥大化するテスト設計に対する効率化策の重要な論拠としても活用されています。今回は、このようなソフトウェアパーティショニング（以降、一部を「パーティショニング」と省略）を適切に設計するための基本の考え方を紹介します。はじめて聞く皆様には順を追って理解していただく良い機会かと思います。よくご存じの皆様はご自身のご経験と照らし合わせて、日々の安全設計が適切に行われているかを振り返る機会として頂ければと思います。

なぜソフトウェアパーティショニングが必要なのか？

ソフトウェアパーティショニングは、決して「流行りの技術だから取り入れる」、「規格に載っているから適用する」というものではありません。安全設計を担保するための重要な技術の 1 つです。とくに前章のようにソフトウェアが支配的になる EV や ADAS のシステムでは必須の技術です。このようなシステムの安全論証におけるソフトウェアパーティショニングの役割を見ていきましょう。



＜図 1：安全論証におけるソフトウェアパーティショニングの役割＞

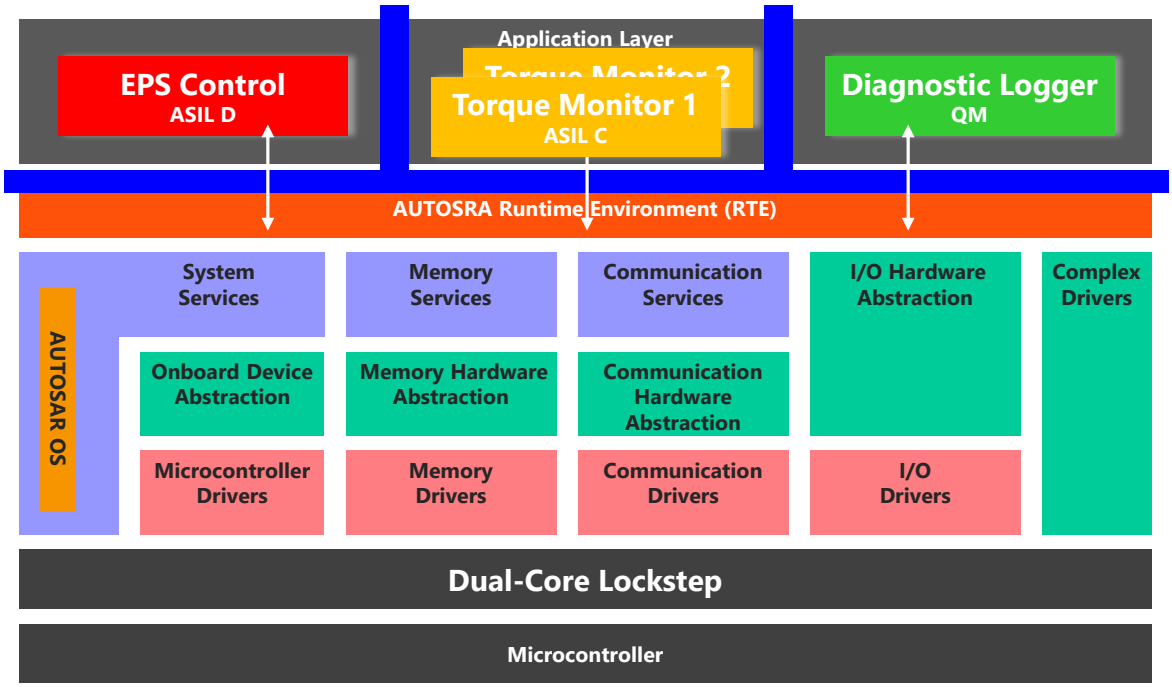
※出典『Biz3 機能安全実装ワークショップ：ソフトウェアのパーティショニング設計』

図 1 は、機能安全の安全論証の基本的な考え方です（論証の裏付けとなる安全分析などは省いています）。この中で特に重要な点が"安全機構は（必要な時に）必ず作用する"です。この点を支える重要な技術がソフトウェアパーティショニングです。ソフトウェアの安全機構を他のソフトウェアから分離し、干渉を防ぐことで、ソフトウェアの安全機構を必要なときに確実に動作させることが可能になります。

例えば、高度運転支援機能の車と人の操作分担を判断、コントロールするシステムでは、フェールセーフも複雑な作用が必要です。このような複雑なフェールセーフの作用は、設計要素の大部分をソフトウェアが担っています。このフェールセーフを確実に作用させるソフトウェアパーティショニングは、最優先の設計事項の 1 つとして扱われています。

非干渉を実現する設計手法とは？

非干渉性を確保するためには、まず守るべき対象と侵害元を明確にする必要があります。ソフトウェアパーティショニングは"何を守り、何から守るか"をアーキテクチャに対する要求として特定することから始まります。図 2 の例に示すような、AUTOSAR 準拠のミドルウェアに搭載されたアプリ機能同士のパーティショニングが典型的な適用事例として業界内でもよく使用されています。この他にも、アプリ機能内のソフトウェアコンポーネント同士のパーティショニング、同じ CAN 通信を共有する制御機能とダイアグ機能とのパーティショニング、のように分離する対象は様々です。重要な点は、ソフトウェアの安全要求を整理し、アーキテクチャを設計する段階から、パーティショニングを適用しやすい機能配置やインタフェースを設計することです。



＜図 2：EPS のパーティショニング設計のイメージ（実際の設計とは異なります）＞

※出典『Biz3 機能安全実装ワークショップ：ソフトウェアのパーティショニング設計』

図 2 は、青色の部分が AUTOSAR によるソフトウェアパーティショニングの機能です。広義には、赤色の「EPS

Control ASIL D」の安全機構の処理やインタフェースを堅牢に設計することもソフトウェアパーティショニングに含まれます。また、緑色の「Diagnostic Logger QM」に発生した異常を他へ影響伝播しないように封じ込めることもパーティショニングの設計手法の 1 つです。ソフトウェアパーティショニングのイメージを持っていただけたかと思います。

次はソフトウェアパーティショニングの設計手法を整理する上で重要な空間・時間・論理・仮想技術の 4 つの観点を説明します。

分離カテゴリ	説明	事例
空間的分離	**ハードウェア機能やゾーン割り当てにより、メモリやI/Oアクセス、バスを互いに隔離する**	**MPU/MMU、セクションマッピング**
時間的分離	**OSやスケジュールの調整により、実行タイミングや優先度の干渉を防ぐ**	**リアルタイムOS(RTOS)、タスクモニタリング**
論理的分離	**ソフトウェア構造・設計のルールにより、データやIF、状態遷移、実行作用を分離する**	**モジュール化、API制限、RTE通信のみ許可**
仮想的分離	**ハードウェア仮想化技術により、異なるOSやアプリを安全に共存させる**	**ハイパーバイザー、仮想ネットワーク**

＜表 1：ソフトウェアパーティショニングの分類＞

※出典『Biz3 機能安全実装ワークショップ：ソフトウェアのパーティショニング設計』

これらの観点は、メモリや演算リソース、通信バスなど、他のアプリソフトウェアやソフトウェアコンポーネントと干渉する懸念がある箇所を整理しています。パーティショニングの技術として、メモリ空間を排他する技術、順序や優先度、経過時間をコントロールする技術などを代表例として載せています。これらの代表的な技術は、いずれも非干渉な状態を構築するためには欠かせない技術です。いずれか 1 つを選択適用するのではなく、守りたい安全機構の処理やインタフェースの特徴を分析して必要な技術を組合せて適用することが重要です。

また、これらの分離を支援する技術に加え、EPS の事例で触れた“安全機構の処理やインタフェースを堅牢に設計”するための、End to End Data Protection、データの 2 度読み出し、処理の冗長化、といった異常検出の技術もパーティショニング手段としてよく使われています。

まとめ

近年のソフトウェアパーティショニングのニーズや用途、設計の考え方といった入口の領域を中心にご覧いただきましたが、いかがでしたでしょうか？自動車のソフトウェア設計が大きく変わる中、このソフトウェアパーティショニングを基本の設計手法の 1 つとして、ぜひ皆様にも安全設計の中で使いこなしていただければと思います。

本メルマガを最後までお読みいただきありがとうございました。具体的なパーティショニング設計はどうするの

か？　安全分析はどうやればいいのか？　といったお声やお悩みなどございましたら、まずは簡単なご相談でも結構ですので、弊社までお気軽にお問い合わせいただけたら幸いです。

2025/6/26　吉川 初芽