ホワイトペーパー

Simulink による ISO 26262 アプリケーション開発のための 11 のベストプラクティス

著者: Jason Moore および John Lee、MathWorks Consulting

MathWorks®

自動車市場でのモデルベース デザインの使用は、過去数十年に渡って拡大し続けています。MATLAB® および Simulink® を使用した自動車の組み込みアプリケーション向けアルゴリズムの開発および展開が、主なユースケースの 1 つとなっています。エンジニアはモデルベース デザインを用いて、エンジン コントローラー、トランスミッション コントローラー、ボディ コントローラー、また最近では自動運転と高度な運転補助システムなどのアプリケーションを開発しています。自動車の組み込みアプリケーションがドライバーからの入力に依存しなくなるにつれ、車両を制御する組み込みシステムにおいて、機能安全要件を満たす必要性が増大しています。

これらのシステムの機能安全要件を満たす確実性を高めるため、開発サイクルのさまざまな観点でエンジニアの指針となる規格が策定されました。自動車の機能安全分野をけん引する規格の一つが、ISO 26262 です。ISO 26262 はトップダウンのワークフローを用いて、システムレベル、ハードウェアレベル、およびソフトウェアレベルのガイドラインなど、開発サイクルのさまざまな観点を分類し、機能安全の目標を達成します。

IEC Certification Kit は、モデルベース デザインを適用できる分野において、ISO 26262 規格への準拠の指針として使用することができます。必要に応じ参照して拡張できる高レベルのワークフローが、キットで提供されています。アルゴリズム構築時に使用するモデリングのスタイルとアーキテクチャは、慎重に検討する必要のある事項です。ISO 26262 を促進するモデリングのスタイルとアーキテクチャを適用することで、無干渉 (FFI) などの規格の重要な項目を満たすために必要な作業を大幅に削減することができます。このホワイトペーパーでは、IEC Certification Kit での参照ワークフローに適したモデリング アーキテクチャの構成に活用できる、幅広いモデリングのベストプラクティスを詳しく説明します。これらのベストプラクティスは、MathWorks 技術コンサルティング サービスが、数々の自動車会社とのコンサルティング業務の中で構築してきたものです。

ISO 26262 を用いる理由

ISO 26262 規格は、電気システムや電子システムにおいて、機能安全の面で開発企業の指針として使用されます。ISO 26262 が取り扱う重要な点には、次が含まれます。

- 機能安全に必要な開発プロセスのステージ
- 自動車安全ライフサイクルでの指針
- システム エンジニアリング、ハードウェア エンジニアリング、およびソフトウェア エンジニアリングについての機能安全の分類
- ASIL (Automotive Safety Integrity Level) 規格を用いて許容可能なリスクレベルの安全要件を定めるための手法
- ASIL に基づいた検証活動での許容基準に関する指針

MATLAB および Simulink などのモデルベース デザイン用開発プラットフォームでは、幅広い組み込みシステムに展開可能なアルゴリズムを作成することができます。また、Simulink を使用すれば、開発サイクルの初期段階でこれらのアルゴリズムを何度も検証することができます。IEC Certification Kit リファレンスワークフローでは、これらの機能を使用して、テスト可能なユニットモデル、統合モデル、およびシステムレベル モデルを作成できる総合的なワークフローを提供します。この高レベルのワークフローは、図 1 と図 2 で示すとおり、2 つのセクションに分けられます。

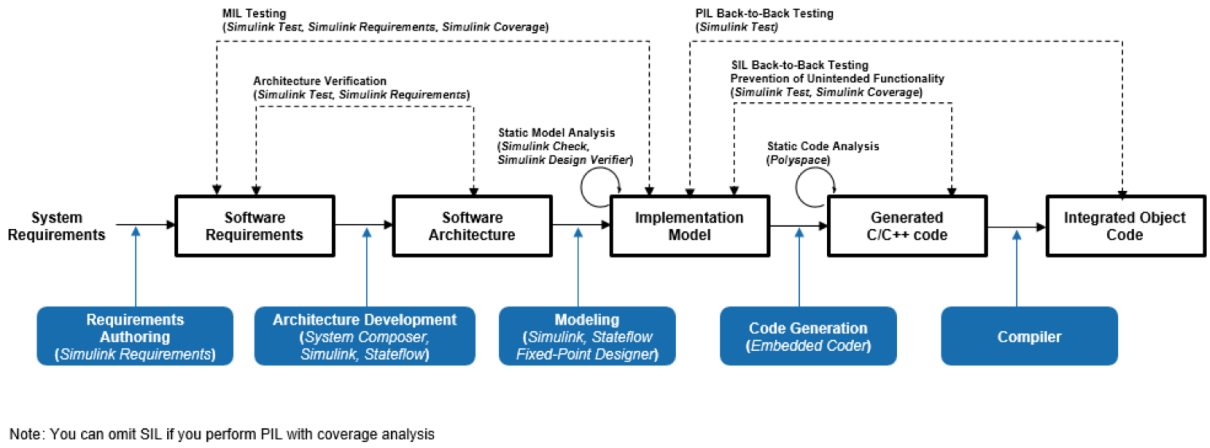

図 1. 開発フェーズ 1: モデルベース デザイン

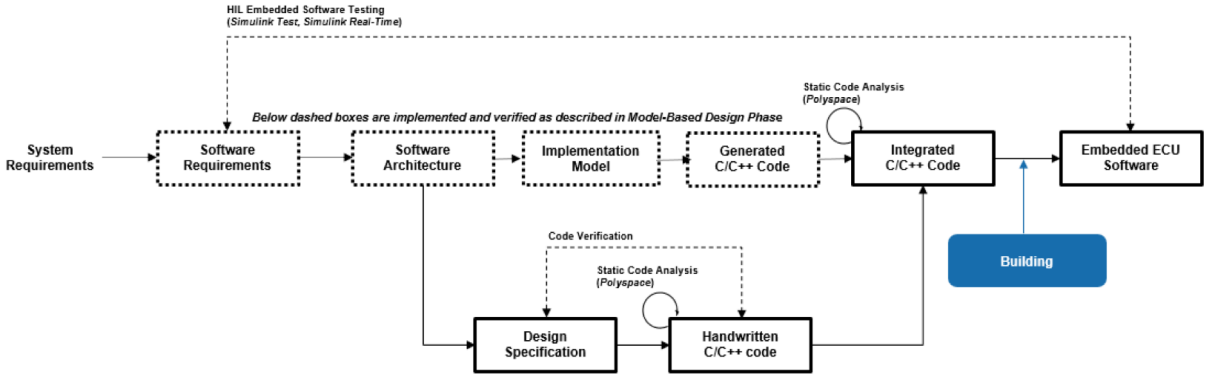

図 2. 開発フェーズ 2: 組み込みソフトウェアのテスト

IEC Certification Kitは、モデル アーキテクチャの推奨構築方法に関する提案は行いません。上記の一般的ワークフローは、アルゴリズムに必要な開発および検証ステージを高レベルで示しています。開発および検証プロセスでアルゴリズムをどう分割するかによって、これらの手順に従う際の作業量と容易さが大きく変化します。アルゴリズム アーキテクチャの設計選択は、開発企業の効率性、ソフトウェアの再利用性、およびソフトウェアのテスト容易性に大きな影響を与えることがあるため、最初にアルゴリズム アーキテクチャの設計選択について熟慮することがとても重要です。

モデルベースデザインを用いた ISO 26262 開発のベストプラクティス

このホワイトペーパーでは、モデルベース デザインを用いて ISO 26262 に準拠しようとする際に、アルゴリズムを分割して検証と開発の手間を減らすために使用することができるモデリング プラクティスを提案します。これらのベストプラクティスは、次のカテゴリーに分類することができます。

- モデル アーキテクチャ:
  1. ユニットレベルのモデルについてのモデル参照の使用
  2. ユニットを機能でグループ化する戦略の選択
  3. モデルのトップレベルにおける ASIL レベルと QM レベルの分割
  4. 統合レベルでのアルゴリズム コンテンツの除去
  5. モデル メトリクスを使用したユニット複雑度のモニタリング
- 信号の経路指定と定義:
  6. バス信号の ASIL、特徴、およびレートによるグループ分け
  7. 必要な信号のみのユニットへの伝達
  8. 信号とパラメーター オブジェクトの配置の最適化
  9. ASIL 間のデータ交換の保護
- コード生成の設定:
  10. コード配置戦略の決定
  11. 共有ユーティリティについての異なる名称トークンの使用

これらのベストプラクティスは、他のベストプラクティスと連携して機能するよう設計されていますのでご注意ください。ベストプラクティスに従わずに全体的な目標を達成することも可能ですが、推奨されません。

モデル アーキテクチャ

Simulink でアルゴリズムを開発するときに最初に決定することの一つに、一般的なモデル アーキテクチャがあります。次の事項に影響するため、この段階での決定が最初の重要なステップとなります。

- ソフトウェアのテスト容易性
- ソフトウェアの再利用性
- ユニットおよび統合テスト手法
- ソフトウェア統合の容易さ
- 無干渉 (FFI) のためのソフトウェア分割

これらのベストプラクティスは、ISO 26262 準拠を目指すエンジニアと共同で作業している MathWorks のコンサルタントが作成したものであり、各項目の最適化に役立ちます。ベストプラクティスは、確認、検証、および文書化フェーズの効率を高め、ISO 26262 準拠を容易にする分野に重点を置いています。

1. ユニットレベルのモデルについてのモデル参照の使用

ISO 26262 第 6 部での主な重点の一つは、ソフトウェア ユニットの開発、確認、および検証のワークフローです。ソフトウェア ユニットは、適切な操作で個別にテストする必要があるアプリケーションにおいて、テスト可能な最小パーツとなることが想定されています。要件がこれら個別のソフトウェア ユニットに対して、またはその内部にマッピングされ、要件ベースのテストケースが構築されて、ソフトウェア ユニットを徹底的にテストします。そのため、ユニット開発に使用されるモデル構造については、次のような特性を考慮する必要があります。

- ユニットのテスト容易性
- ユニットのコード生成
- ユニットの複雑度
- ユニットのテスト ワークフロー
- ユニットのトレーサビリティ
- ユニットのドキュメンテーション

これらの特性では、モデル参照をユニット開発での最初のモデリング パターンとして挙げています。モデル参照には、希望する結果に導く次のような複数の特性があります。

- コード生成 – モデル参照から生成されたソース ファイルへの一対一のマッピングがあります。
- 再利用性 – モデル参照は統合モデル内の複数個所で使用できます。
- テスト容易性 – モデル参照はテスト ハーネスの構築に理想的なものです。
- チームの共同作業 – モデル参照は開発者間で並行した開発を可能にし、全体の設定管理とバージョン管理プロセスを単純化します。

図 3 では、モデル参照を使用して、機能をテスト可能なユニットに分割する方法を示しています。

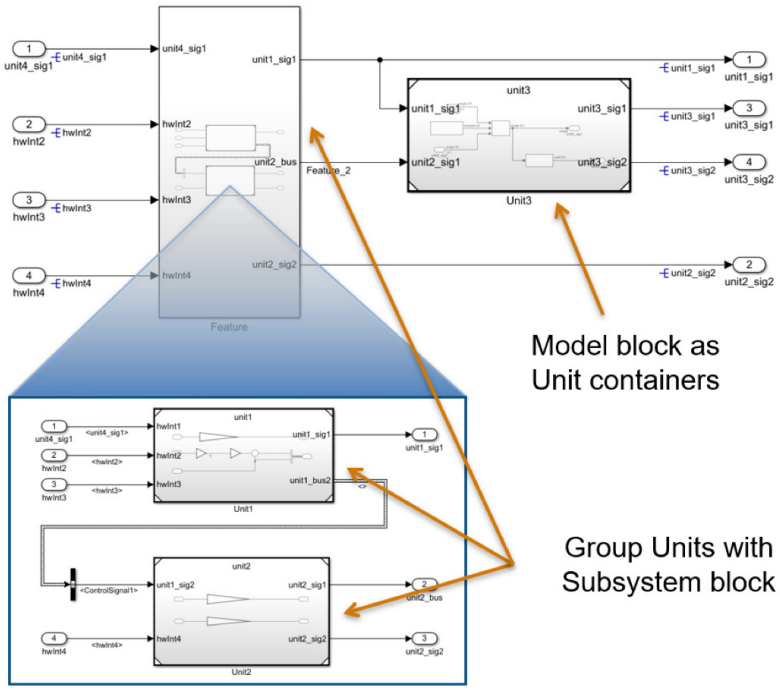

図 3. アルゴリズム ユニットへのモデル参照の使用

ベストプラクティスとしては推奨されていないものの、Simulink ライブラリのアトミック サブシステムをソフトウェア ユニットとすることができます。このアプローチでは、開発者が分割とテストの境界を明確に管理して制御する必要があります。たとえば、インターフェイスは固定する必要があるため、ユーザーはデータ型、サンプル時間、および関数プロトタイプ制御やファイル分割などのコード生成オプションを含む関連するブロック設定を指定する必要があります。アトミック サブシステムをシングル レート、マルチレート、またはマルチタスクなど異なる実行環境に置く場合、追加的な管理と評価が必要となります。Simulink の 2019a のリリースでは、再利用可能なサブシステム ライブラリ向けのライブラリベースのコード生成が導入され、このようなアトミック サブシステムの使用が促進されています。これにより、アトミック サブシステムをスタンドアロンのユニットとして管理、指定、生成、テスト、再利用しやすくする新しい機能が追加されました。

また、サブシステムをコンテナとして使用し、類似ユニットの機能グループ分けをさらにサポートすることもできます。モデルとデータ管理で複雑度が増すため、通常は、ネストされたモデル参照は推奨されません (詳細は次のベストプラクティスを参照)。

2. ユニットを機能でグループ化する戦略の選択

大規模なモデルを構築する場合、さまざまな種類のモデル構造から選択し、次のようなモデル階層を追加することができます。

- 仮想サブシステム
- アトミック サブシステム
- モデル参照
- ライブラリ ブロック

ISO 26262 に対応するため、これらのユニット モデルのそれぞれの ASIL へのグループ分けには柔軟性が保持されています。前のベストプラクティスでは、モデル参照をユニットレベルのアルゴリズムに使用することを要求する「強固な」制約を加えました。ただし、ユニットのグループ化は、サブシステムまたはモデル参照でも行うことができます。考慮すべきトレードオフは、管理が必要なモデル参照数や、機能レベルで希望するモデル境界の厳格さ等です。アーキテクチャがモデルのセグメント方法に中立で、ユーザーが大きなモデルで作業できる場合は、これらのユニットは仮想サブシステムにグループ化されます。機能モデルのセグメント化戦略を決定する際に、次の事項を検討します。

- 生成されたコード ファイルと機能のグループ化
- 複数開発者による並行した機能開発
- モデル参照ファイルに関連するオーバーヘッド
- 設計における機能のグループ化の可視性

ユニットとトップレベルの間のこの中間レベルのモデルでは、モデル構造の選択は、組織のモデリング ガイドラインと設定によって決まります。図 4 では、仮想サブシステムの使用で許容される手法を示します。

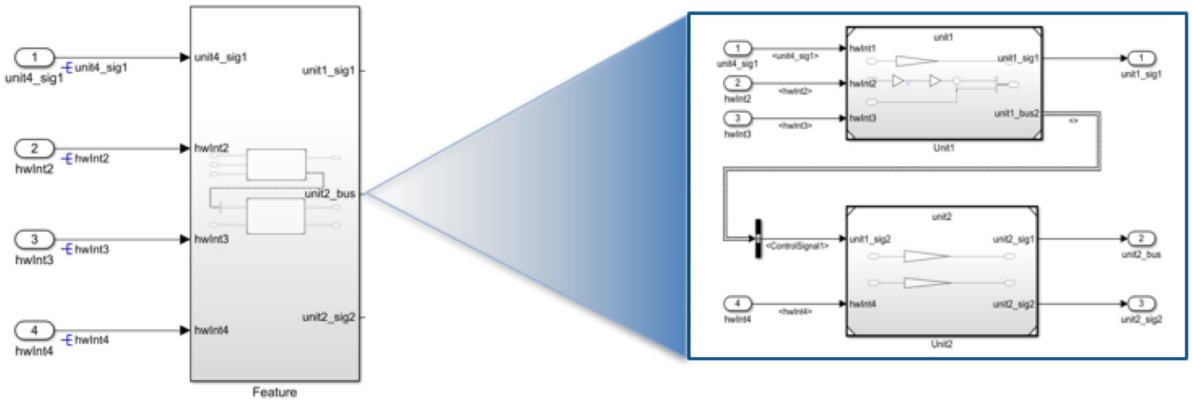

図 4. 複数ユニットの機能でのグループ化

3. モデルのトップレベルにおける ASIL レベルと QM レベルの分割

ISO 26262 の重要な概念の一つは、無干渉 (FFI) を軸としています。ISO 26262 には 5 つの安全レベル (品質管理 [QM]、および ASIL A–D) があり、システムの機能安全に関する観点に基づいて、システムレベルおよびソフトウェアレベルの機能の分類に使用することができます。ISO 26262 に準拠する電気システムおよび電子システムは、異なる ASIL のコンポーネントを持っている場合があります。たとえば、クリティカルではない診断データを出力するアルゴリズムの一部は、QM コンポーネントとして分類される場合があり、また、車両のブレーキ能力に影響を及ぼす可能性のあるアルゴリズムのセクションは、障害時のハザード/傷害リスクが大きいため、高レベル ASIL コンポーネントに分類される場合があります。

複数の ASIL コンポーネントを持つシステムでは、これらのアルゴリズムを異なるコンテナに効率的にセグメント化するアーキテクチャから恩恵を受けます。これは、次の 2 つの理由によります。

- それぞれの ASIL に異なる開発、検証、および確認の要件を設定することができる。
- ASIL を分割してセグメント化することで、無干渉 (FFI) を実現できる。

さまざまな ASIL が分割されるため、モデル構造を選択してセグメント化を補助する必要があります。アルゴリズムを展開する際にそれぞれの ASIL の間にしっかりとした境界ができるよう、モデル参照を使用します。そのため、システムレベル モデルのトップ レベルを、それぞれ分離した ASIL を表すモデル参照から構成される複数モデル参照群に分割する必要があります。このケースでは、コード生成の設定 (詳細は「コード生成の設定」のセクションを参照) のため、システムレベルのモデルはシミュレーション用途のみで、コード生成のみ ASIL に基づいて分割して実行できるという点に注意してください。代わりに、ユニットレベルでコード生成して統合することもできますが、可能なかぎり高いレベルでコードを生成することにより、コード生成時のオーバーヘッドの量が削減されます。

それぞれの ASIL にモデル参照を使用することにより、またはより一般的には無干渉が必要なときはいつでも、分割した関数とソース ファイルがそれぞれのモデル参照用に生成されます。これに加えてコード生成設定のベストプラクティスに従うことで、セグメント化された各部分は、独自のソース ファイル、共有ユーティリティ、およびデータ定義を持つことになり、アルゴリズムの異なるセクション間での無干渉を実現しやすくなります。図 5 では、モデルの ASIL に基づいた階層的な分割を示します。

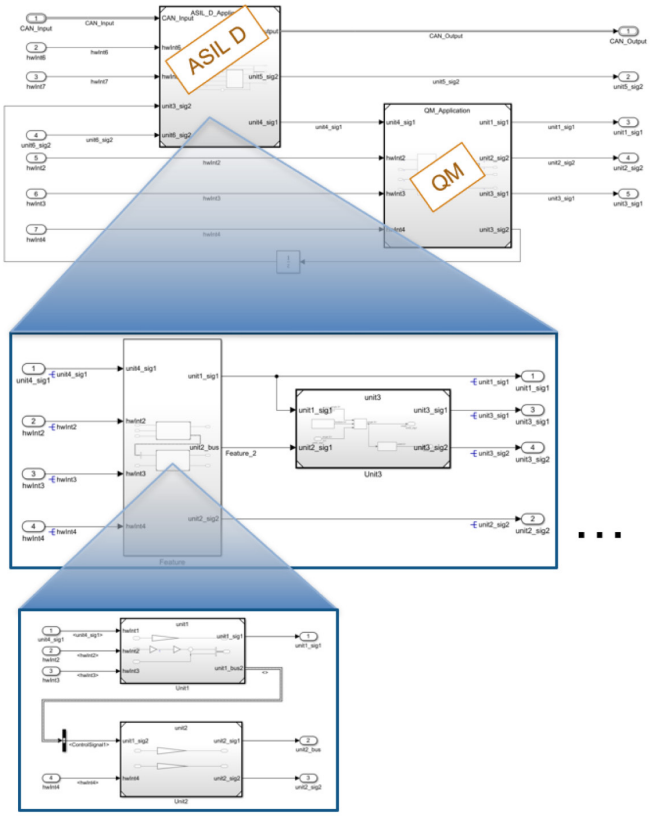

図 5. ASIL に基づいたモデルの階層

4. 統合レベルでのアルゴリズム コンテンツの除去

ISO 26262 には、ユニットレベル、統合レベル、およびシステムレベルなど、それぞれのアーキテクチャでの複数のテスト実行レベルに関する概念があります。通常は、ソフトウェア ユニットに対して、対象の ASIL に基づくさまざまなレベルの厳密なテストを行う必要があります。たとえば、ASIL D では、完全な変更条件/判定カバレッジ (MC/DC) が要求されることがありますが、ASIL A または B では、条件/判定カバレッジで十分な可能性もあります。このため、ユニットが、アルゴリズム機能を実装する唯一の場所となります。アルゴリズム コンテンツがモデルの統合またはシステムレベルで発生する場合、設計のフル カバレッジを達成することがより難しくなります。そのため、ユニットレベル モデルの外部でアルゴリズム コンテンツが発生しないよう確認しておくことが推奨されます (図 6)。

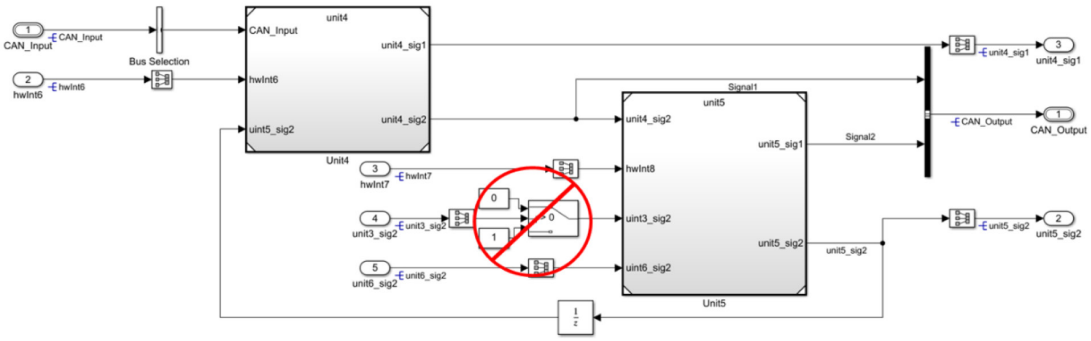

図 6. ユニットの外部でのアルゴリズム コンテンツの回避

このベストプラクティスは、MathWorks Automotive Advisory Board (MAAB) のルール db_0143: 各モデル階層の類似ブロック タイプにもマッピングされます。

5. モデル メトリクスを使用したユニット複雑度の監視

多くの企業が、開発サイクルの後期で、ISO 26262 が推奨するカバレッジのレベルまでのアルゴリズムの検証が難しくなることを理解しています。これは通常、設計時にアーキテクチャを考慮しなかったこと、および開発時にユニットレベルのサイズと複雑度を管理しなかったことにより生じます。この問題を緩和する方法論の一つは、開発企業全体でユニット サイズのメトリクスしきい値を設定することです。これらのしきい値には、次が含まれることがあります。

- ユニットの最大入力/出力数
- 再利用可能なライブラリ
- 循環的複雑度
- 要素数

ユニットの複雑度を管理するには、開発サイクルのモデル レビューのプロセスにこれらのメトリクスのレビューを取り入れます。これにより、複雑度が可視化できるほか、開発プロセスにおけるアルゴリズム設計のスコープも設定することができます。Simulink Check™ のモデル メトリクス ダッシュボード (図 7) でこのプロセスを単純にすることができます。

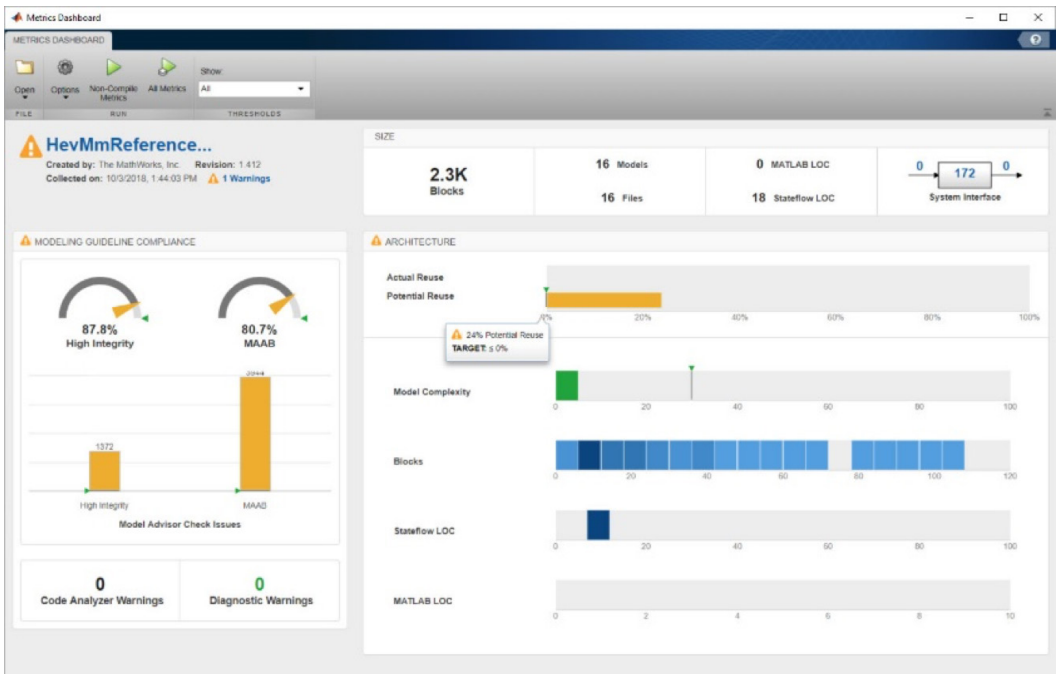

図 7. Simulink Check™ のモデル メトリクス ダッシュボード

モデル メトリクス ダッシュボードでは、モデル作成者とモデルのレビュー担当者が、総ブロック数、MATLAB コード行数 (LOC)、Stateflow® LOC、モデル複雑度、ブロック数、および他のメトリクスを簡単に確認できるようにすることができます。ダッシュボードは、継続的にモデルを監視することで設計ワークフローをサポートしています。これにより、それらが小さすぎるユニットに分割されていないか (管理の複雑度を増大させます)、また大きすぎないか (簡単にテストできず再利用が困難となります) を確認することが可能になります。

しきい値は、検討の対象となることがあります。これはモデルの複雑度はモデルの単一の観点からは計測できず、多くの場合、意味のある決定を行うには複数のメトリクスを解析する必要があるためです。たとえば、ブロック数が 1 つであるにもかかわらず、1 つの Stateflow チャートに非常に高い循環的複雑度がある場合があります。

MathWorks と自動車会社グループ (Delphi Technologies、Bosch、PSA、Renault、および Valeo) から最近発表された論文、*「MATLAB と Simulink を使用した組み込みソフトウェア開発における品質目標のモデル化」*では、メトリクスとしきい値の指針が提供されています。たとえば、モデルの循環的複雑度は 30 以下、またモデルの要素数は 500 未満とされています。

各会社が異なるしきい値を持っており、MathWorks 技術コンサルティング サービスがこの分野をサポートをします。

信号の経路指定とインターフェイスの定義

ISO 26262 第 6 部には、ユニットとコンポーネント間でのインターフェイスの複雑度とデータ交換に対処するための複数の検討事項があります。たとえば、次のようなものがあります。

- 表 3 - 1b: ソフトウェア コンポーネントのサイズと複雑度の制限
- 表 3 - 1c: インターフェイスのサイズ制限
- 表 7 - 1g: データ フロー解析
- 表 7 - 1k: インターフェイスのテスト

これらの目標を達成するには、ユニット、コンポーネント、および異なる ASIL 間でのデータ交換方法に関するアーキテクチャ戦略を決定する必要があります。MathWorks は、さまざまなエンジニアリング チームと共同作業する中で、ユニットレベル モデル、コンポーネント、およびさまざまな ASIL 間でのデータ インターフェイスを解析するために必要な作業を削減する、複数のベストプラクティスを作成してきました。このセクションでは、インターフェイスの複雑度とデータ交換を管理する 4 つのベストプラクティスをご紹介します。

6. バス信号の ASIL、特徴、およびレートによるグループ分け

ISO 26262 第 6 部の表 7 - 1g では、開発チームがデータ フロー解析を行うことが推奨されています。データ フロー解析は、信号がどのようにソフトウェア アルゴリズム中からユニットレベルまで伝達されているかを理解するために必要なものです。この種類の解析では、信号の要件が衝突しているエリア、または提供元の特性よりさまざまな信号を直接使用すべきでない分野を特定することができます。この解析を実行するには、開発企業がバス階層戦略を策定し、開発者が信号の送信元と信号の特性を理解しやすくする必要があります。バス信号を階層的にグループ化する方法を指定しない場合、次の問題が発生することがあります。

- 非効率的なバスのセグメント化とモデリング パターン
- 開発者の間で一貫性のないバスのグループ化
- バスを分割して再作成する際のモデリングの問題
- 非効率的なコード生成

モデル アーキテクチャがトップダウンの設計アプローチを必要とするように、バス階層にも同じことが必要になります。それぞれの信号用 ASIL を管理するため、タスク レートと ASIL に基づいてバス階層内で信号をグループ化します (図 8)。ASIL に基づいてこれらの信号をグループ化することにより、信号の送信元を調べ、信号がより高いレベルの ASIL のユニットで使用されているかどうかを調べることが容易になります。たとえば、信号が QM コンポーネントから送信されているものの ASIL コンポーネントで使用されている場合、この信号の依存関係が解析され構成されているかを確認するため、追加の解析が必要になります。

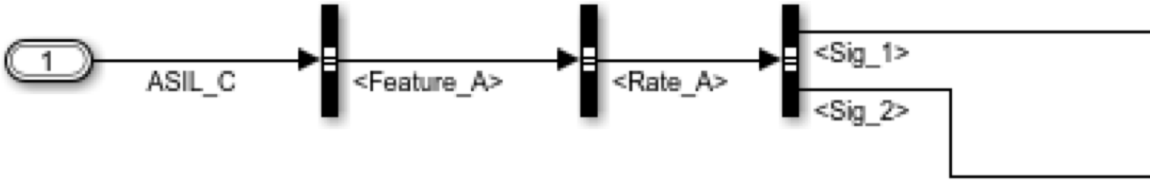

図 8. バス階層内における信号のグループ化

7. 必要な信号のみのユニットへの伝達

ISO 26262 第 6 部の表 3 と表 7 では、ユニットレベルのインターフェイスとデータ交換に関する重要な提案をしています。これらの 2 つの表では、ソフトウェア コンポーネントとインターフェイスのサイズと複雑度は可能な箇所では削減されるべきであることが示されています。また、検証プロセス中のインターフェイスに関するテストも推奨されています。不必要な変数がユニットレベルのモデルまで伝達されている場合、これらの信号がユニットに影響しないことを確認するため、追加のテストが必要になります。この懸念を軽減するには、ユニットレベルへ伝達される入力の削減が役立ちます。これを行うために、ユニットレベルのモデルに入る前のバス信号を、対応するユニットで使用される信号のみに分割することが可能です (図 9)。

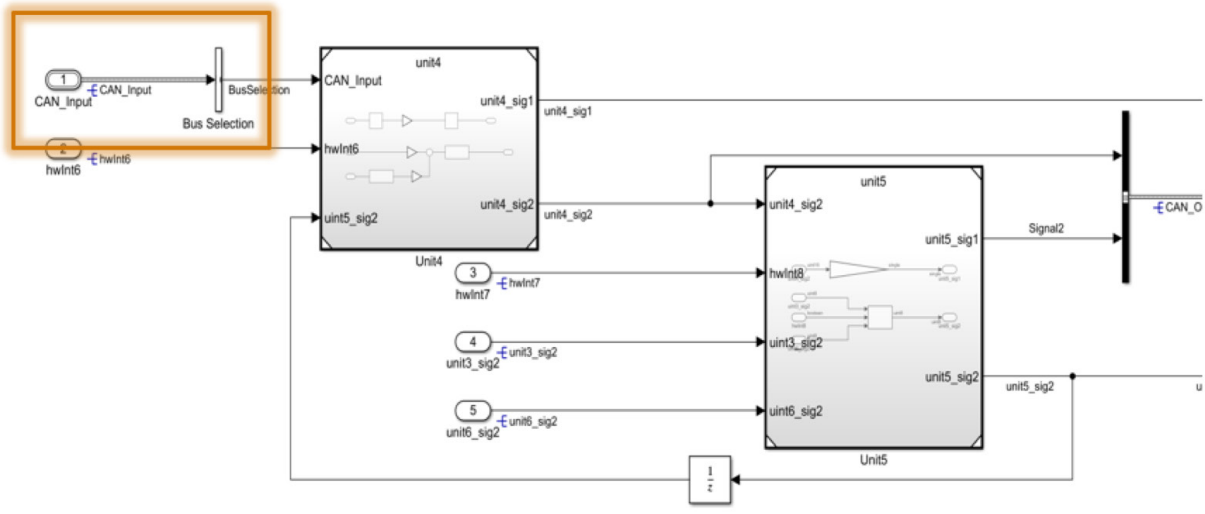

図 9. バス信号をユニットレベルのモデルに入る前に分割

8. 信号とパラメーター オブジェクトの配置の最適化

信号とパラメーターオブジェクトに関する高レベルでのユースケースは、モデルと基盤ソフトウェアの間のインターフェイスを定義することです。このようなユースケースでは、パラメーター オブジェクトは通常キャリブレーション値を指定するために使用され、ゲインとルックアップテーブルのブロックなどのモデル ブロック内に配置されます。信号オブジェクトの用法はより複雑ですが、これは通常、キャリブレーション アクティビティをサポートするため内部信号、またはルート入力/出力ポートと関連づけられています (図 10)。

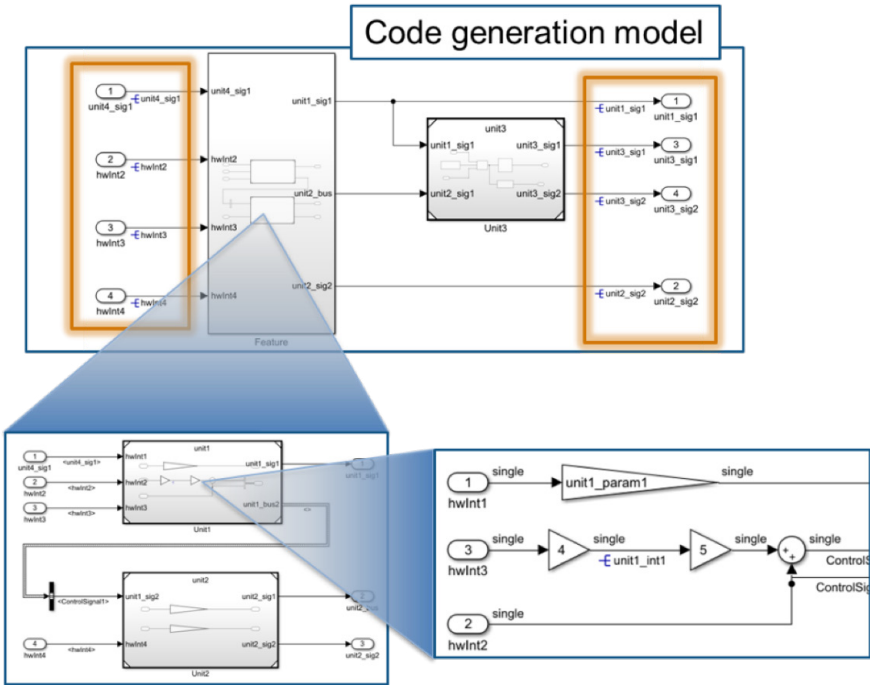

図 10. 信号とパラメーター オブジェクトの配置

モデル参照ブロックへのインターフェイスには、信号オブジェクトが含まれていないことに注意が必要です。これは、ユニットレベルの境界の信号は、内部インターフェイス信号とみなされているためです。また、上記ベストプラクティス 6 に記載された最も高度な ASIL パーティションで生成されたコードも、このことを前提としています。内部インターフェイス信号用に、Embedded Coder® でこれらの信号を内部最適化アルゴリズムに基づいて定義することが最善の手段です。ただし、データ型情報は、図 11 で表されているように、モデル参照境界でのポート ブロック設定の一部として明示的に定義する必要はありません。

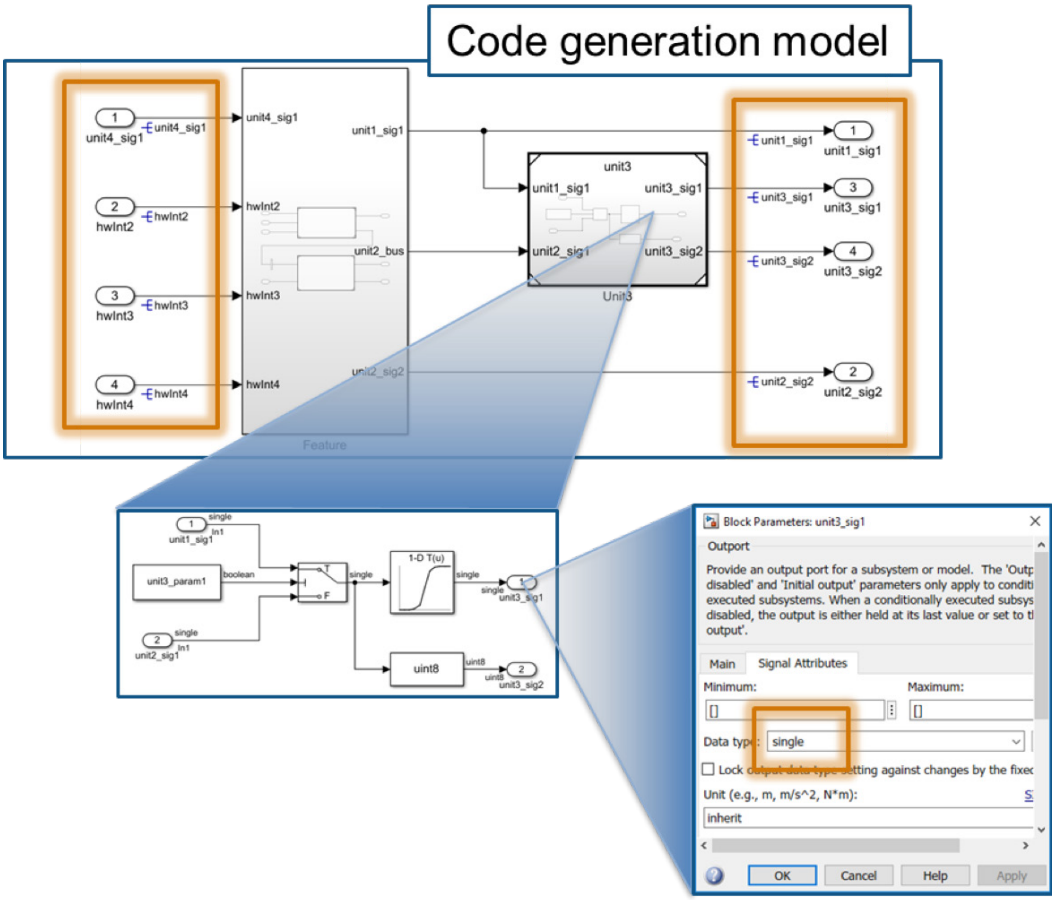

図 11. ポート用のデータ型を定義

このベストプラクティスは、インターフェイス信号オブジェクトの配置をコード生成モデル境界に制限します。また、これにより、それぞれの ASIL にコードが生成される際に、確実にルートレベルの出力ポートが他の ASIL セクションのインターフェイス関数を指すようにすることができます。

信号とパラメーター オブジェクトに使用されるストレージ クラスは、ソフトウェア アーキテクチャとコーディング方法に基づいて設定することができます。ただし、ASIL モデル間のデータ交換に必要な保護、または無干渉に必要となる分割は対象外となります。

9. ASIL 間のデータ交換の保護

それぞれの ASIL 用のコードを別々に生成する際に配慮することの一つは、ASIL 間でのデータ交換に保護手段が必要だということです。複数の戦略が、ASIL セクションのルートレベル入力/出力ポートでストレージ クラスを用いて存在しています。一般的な方法の一つは、データに対する get および set アクセス関数を持つストレージ クラスを使用することです。get および set アクセス関数を用いることで、適切なソフトウェア コンポーネントのみデータにアクセスできるよう、これらのインターフェイスに保護を追加することができます。

図 12 では、GetSet ストレージ クラスをルートレベル入力ポートと対応する生成コードで使用する例を示しています。

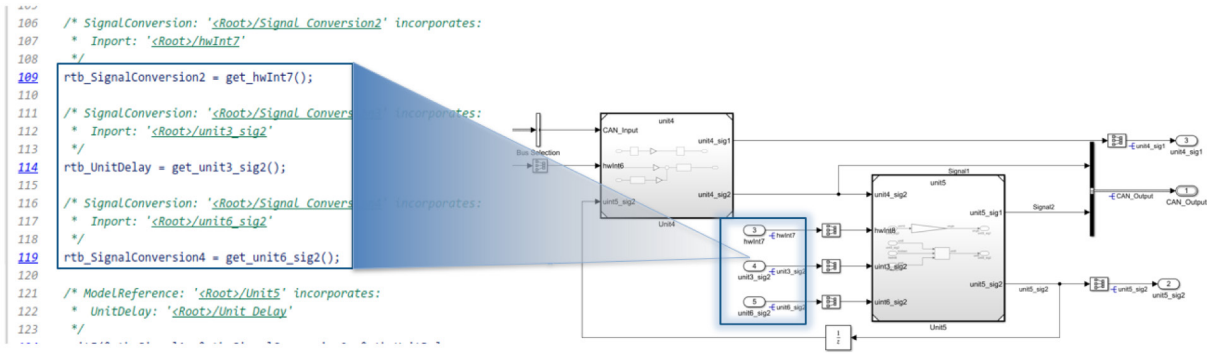

図 12. ASIL コンポーネント間の非 AUTOSAR インターフェイス

図 13 では、ストレージ クラスの一つを設定して、Get および Set API を ASIL 間で合致させる方法を示しています。

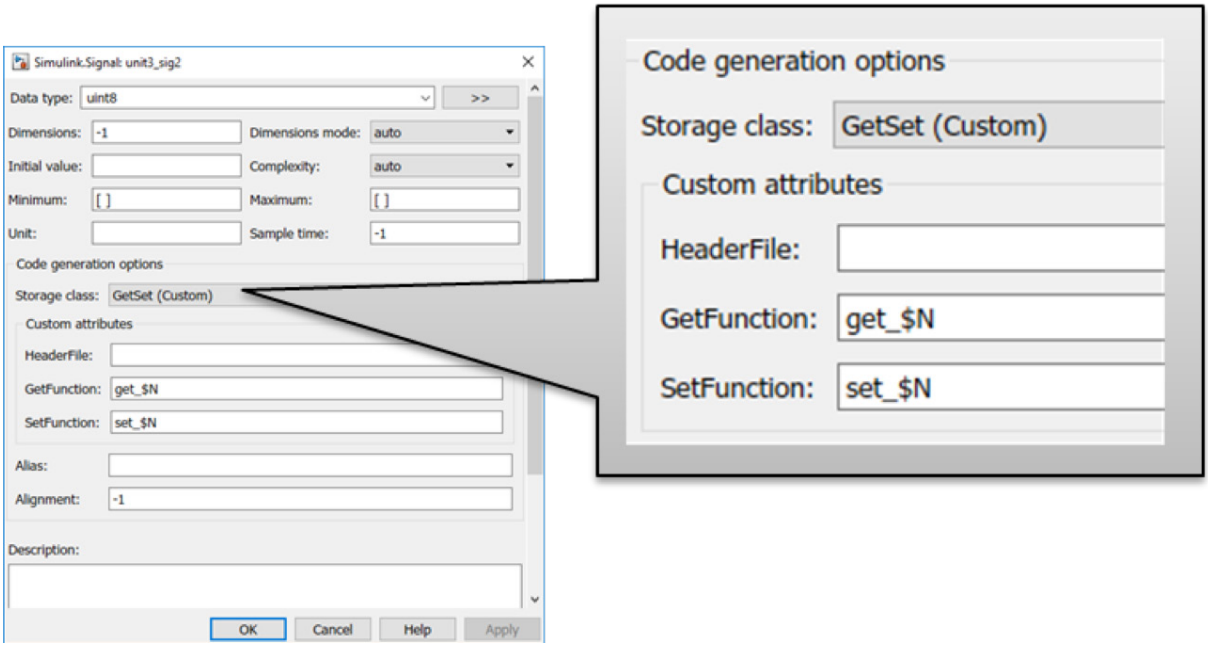

図 13. ルートレベル I/O ストレージ クラスのコード生成オプションの設定

GetSet ストレージ クラス オブジェクトは、インターフェイス保護のエントリ ポイントのみ提供することに注意が必要です。保護の実際の実装は、一般的に手動コーディングにより実装された低レベル ソフトウェア層により行われ、GetSet ストレージ クラスを用いて簡単に統合することができます。

コード生成の設定

上記ベストプラクティスに沿ってモデルを開発した後も、コード生成の設定において、ISO 26262 によって定められた目標を実現するための特別の注意が必要になります。このセクションでは、共有ユーティリティ ファイルでのコードの配置と分割の観点から、コード生成の設定に関するベストプラクティスについて説明します。ここでも、以下に記載された設定は、前のベストプラクティスに従っていることを前提としています。

10. コード配置戦略の決定

ISO 26262 の重要な設計概念の一つは、無干渉 (FFI) です。特定の ASIL にあるシステムの一つのセクションで問題が発生した場合も、分離した ASIL の機能に影響を与えないことを確実にするため、無干渉が必要になります。たとえば、QM または ASIL A コンポーネントで問題または障害が発生した場合も、ASIL D 機能が動作を確実に継続できるよう、設計ではこの機能を ASIL D 機能から離れてセグメント化することが推奨されます。組み込みシステムでは、アプリケーションの一部がメモリーセクション、またはアクセスすべきではない関数へアクセスしている場合が、懸念される欠陥の一種として挙げられます。この懸念に対処する一つの方法は、機能を異なるメモリーセクションに、またはマイクロプロセッサの異なるコアに分割することです。これは、コンパイラに、それぞれの変数、関数、またはファイルが、どのセクションに配置されるべきかを伝えることにより実現できます。図 14 はアプリケーションの分割を示しています。

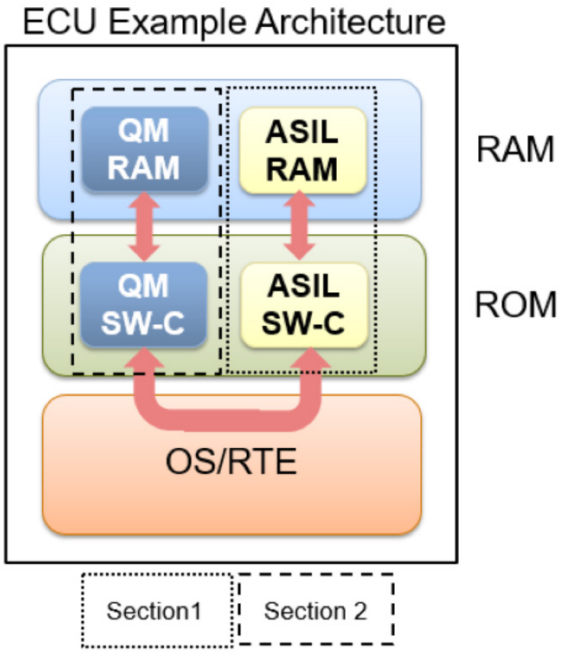

図 14. システム アーキテクチャのセグメント化による無干渉の実現

もう一つの手法は、さまざまな ASIL と QM レベルを異なるメモリー セクションに分割することです。メモリー セクションを分割することにより、さまざまな ASIL が意図せず連携する懸念が抑えられます。たとえば、QMソフトウェアコンポーネント (SW-C) 内の関数による、保護されたASIL RAMへの書き込みなどです。この設定を行うために、図 15 のようにメモリー セクションを設定オプションから選択することができます。この方法を用いる必要はありませんが、これによりワークフロー全体が簡略化されます。

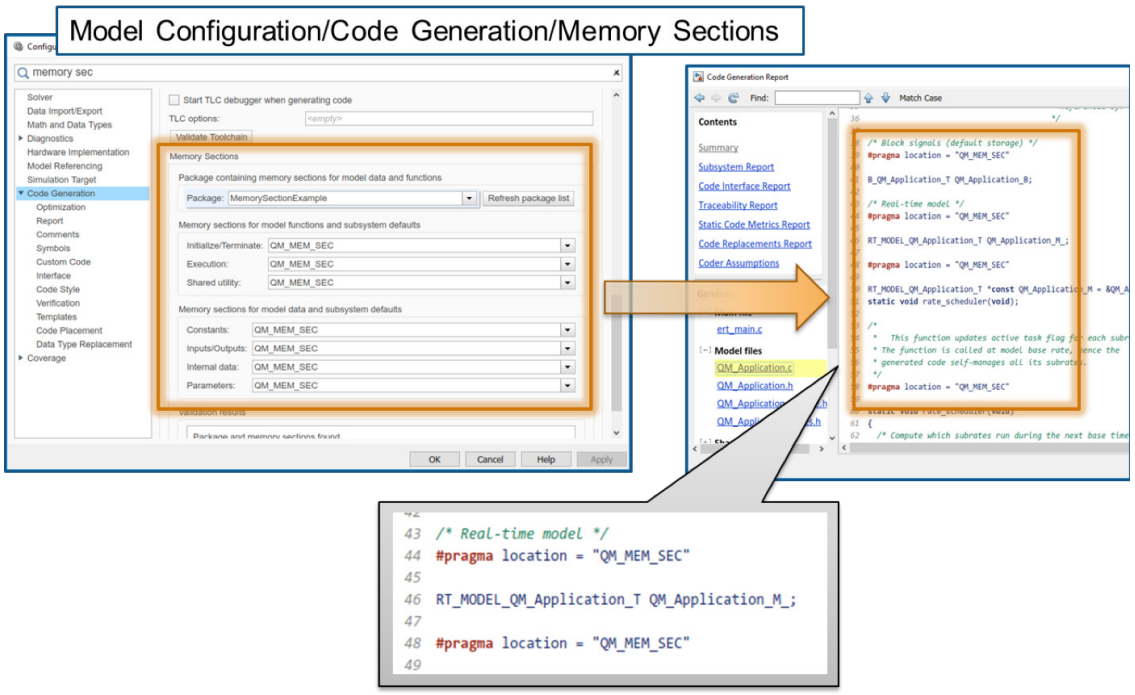

図 15. メモリー セクションのパラメーターの設定

11. 共有ユーティリティについての異なる名称トークンの使用

Embedded Coder は、共有ユーティリティとして知られる共通のユーティリティ ファイルを生成します。これらのファイルは、コード生成モデルにまたがって使用される基本的な関数になります。この方法は、元となる共有ユーティリティ ファイルに区別が無いため、安全関連のアプリケーションに問題をもたらします。セグメント化の概念を持つ信号アプリケーションにコードをコンパイルするには、共有ユーティリティが、その ASIL に基づいて生成され割り当てられる必要があります。これは、図 16 のようにコード生成の設定で行うことができます。

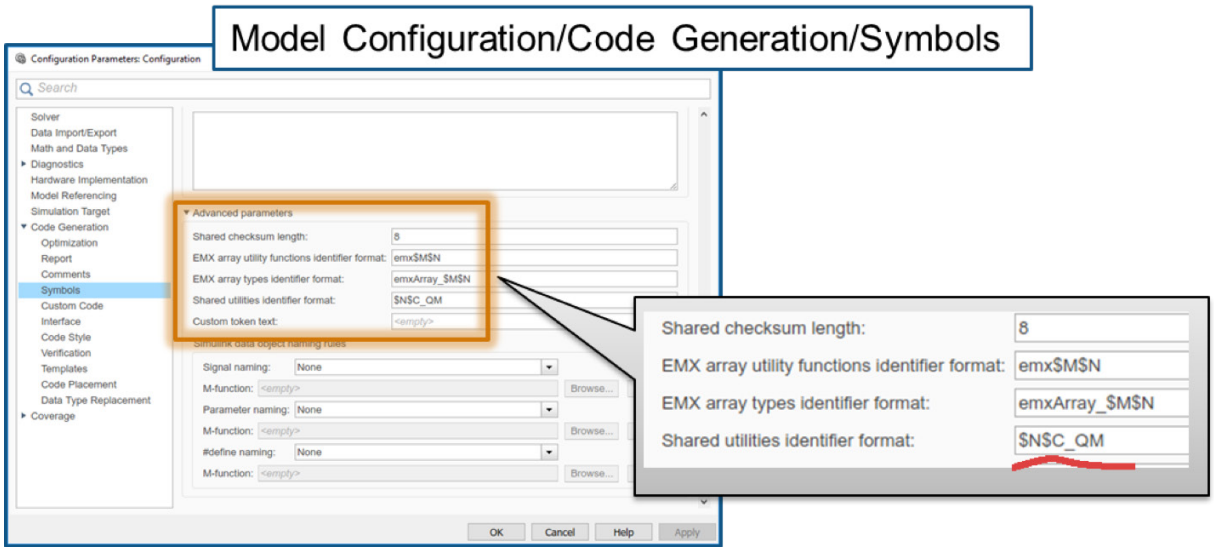

図 16. 共有ユーティリティの設定

上記の設定により、ASIL 関数であるユニークな識別子を持つ生成済み共有ユーティリティを作成することができます。例として、図 17 に、上記 QM 接尾辞に基づいて設定された場合のルックアップテーブル ブロック用の共有ユーティリティを示します。

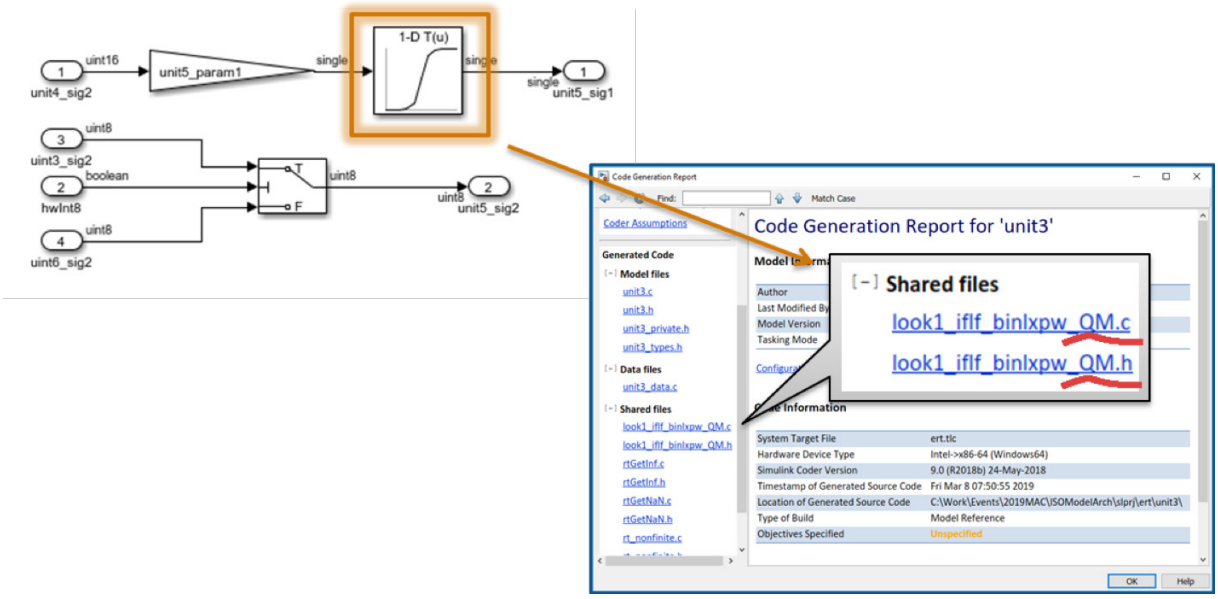

図 17. QM 用に設定された共有ユーティリティ

まとめおよび将来の課題

この文書に記載された発見は、MathWorks がコンサルティングに従事した複数のケースから作成されたベストプラクティスです。これらのベストプラクティスは、ISO 26262 の適用を可能にすることが実証されています。ただし、これらは ISO 26262 の要求事項の一部のみに対応しており、それぞれのアプリケーションには固有のニーズがあるため、これらのベストプラクティスに従うことだけでは ISO 26262 の準拠は保証されません。

これらのベストプラクティスを AUTOSAR 規格に適用する作業が進行中です。*ホワイト ペーパーの初期ドラフトの申し込み*

関連情報

- *MATLAB および Simulink における ISO 26262 のサポート* - 概要
- *ISO 26262 プロセス導入アドバイザリ サービス* - 技術コンサルティング サービス