白皮书

使用 Simulink 开发 ISO 26262 应用的 11 项最佳实践

作者：Jason Moore 和 John Lee，MathWorks Consulting

MathWorks®

在过去几十年里，基于模型设计在汽车领域得到了广泛运用。其中一个主要使用场景是用 MATLAB® 和 Simulink® 为汽车嵌入式应用开发和部署算法。工程师使用基于模型设计开发了多种应用，例如发动机控制器、变速箱控制器、车身控制器，以及最近的自动驾驶 (AD) 和高级驾驶员辅助系统 (ADAS)。随着汽车嵌入式应用不断发展，对驾驶员干预的依赖日益减少，控制汽车的嵌入式系统越来越需要满足功能安全要求。

为了满足这些系统的功能安全要求，相关标准相继制定，围绕开发的各个方面为工程师提供指导。ISO 26262 是一项在汽车功能安全领域广受欢迎的标准。 ISO 26262 采用自上而下的工作流程，分解开发周期的各个方面，包括系统级、硬件级和软件级开发指导，以实现功能安全目标。

MathWorks 的工具箱 IEC Certification Kit 可作为遵循 ISO 26262 标准适用于基于模型的设计指南。此套件提供了详细的工作流程，可按需参考和扩展。构建算法时，您需要认真考虑要使用的建模风格和架构。通过采用符合 ISO 26262 的建模风格和架构，可以大大减少为满足标准关键方面所需的工作，例如软件模块间的互不干扰 (freedom from interference)。本白皮书详细介绍了建模最佳实践，可用于构建模型架构，补充从 IEC Certification Kit 中引用的工作流程。在与各大汽车公司开展咨询项目时，MathWorks 咨询服务部门已公开这些最佳实践。

为何选择 ISO 26262?

ISO 26262 标准可用于指导开发部门完成电子电气系统功能安全方面的开发工作。ISO 26262 标准从以下方面提供关键指导?

- 功能安全所需的开发流程阶段
- 汽车安全生命周期指导
- 系统工程设计、硬件工程设计和软件工程设计的功能安全分解
- 使用汽车安全完整性等级 (ASIL) 标准确定可接受风险级别的安全要求的方法
- 根据 ASIL 开展验证活动的验收标准指导

MATLAB 和 Simulink 等基于模型设计的开发平台使您能够为各种嵌入式系统创建可部署的算法。Simulink 还使您能够在开发周期中尽早和经常地验证这些算法。IEC Certification Kit 参考工作流程使用这些功能提供全面的工作流程，用于创建可测试的单元模型、集成模型和系统级模型。此高层次工作流程分为两部分，如图 1 和 2 所示。

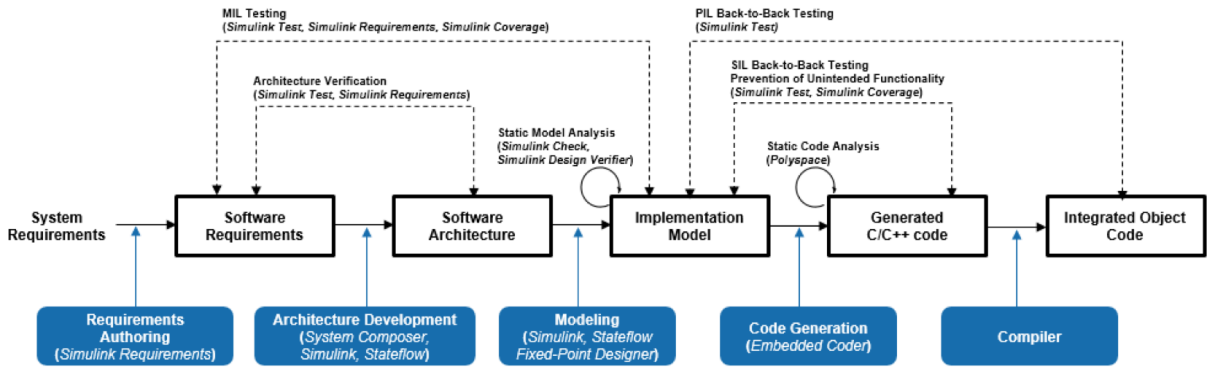

图 1. 开发阶段 1：基于模型设计。

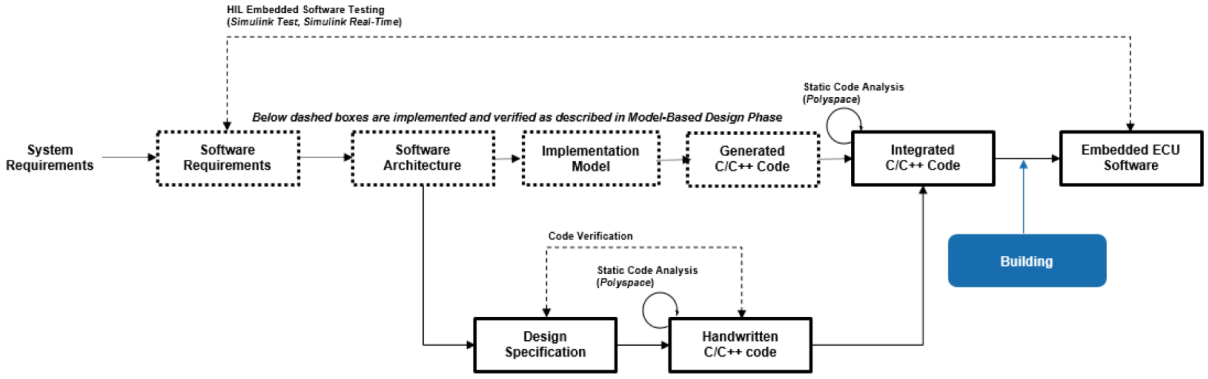

图 2. 开发阶段 2：嵌入式软件测试。

IEC Certification Kit 不针对应当如何构建模型架构提供建议。上面列出的一般工作流程在较高层次上展示了算法所需的开发和验证阶段。根据开发和验证过程中算法的分解方式，工作量和难易程度可能会有很大差异。首先请仔细斟酌这些算法架构的设计，因为它们会对开发组织的效率、软件的可重用性和软件的可测试性产生重大影响。

使用基于模型设计开发 ISO 26262 的最佳实践

本白皮书就建模实践提供了相关建议。在遵守 ISO 26262 和基于模型设计时，可使用这些建模实践对算法进行合理分割，以减少验证和部署工作。这些最佳实践可分为以下几类：

- 模型架构：
  1. 单元级模型设计使用模型引用
  2. 选择策略将单元按特性分组
  3. 在顶层模型中按照 ASIL 等级和 QM 等级进行拆分
  4. 集成级别不包括具体算法
  5. 使用模型指标监控单元复杂度
- 信号路由与定义：
  6. 按 ASIL 等级、特性和速率对总线信号进行分组
  7. 仅向单元传递需要的信号
  8. 优化信号和参数对象的布局
  9. 对不同 ASIL 等级模块之间交换的数据进行保护
- 代码生成配置：
  10. 确定代码布局策略
  11. 共享的代码使用不同的名称标记

请注意，这些最佳实践可配套使用。偏离最佳实践依然可能会实现总体目标，但不推荐。

模型架构

在 Simulink 中开发算法时，第一步是决定采用怎样的通用模型架构。这个第一步非常重要，因为在这个阶段做出的决定会影响：

- 软件可测试性
- 软件可重用性
- 单元和集成测试方法
- 软件集成的难易度
- 软件模块合理划分实现互不干扰 (freedom from interference)

这些最佳实践由 MathWorks 顾问在与需要遵守 ISO 26262 的工程师合作的基础上总结编写而成，旨在帮助您优化上述特性。最佳实践围绕简化 ISO 26262 遵从的相关方面，同时提高验证、确认和文档阶段的效率。

1. 单元级模型设计使用模型引用

ISO 26262 第 6 部分的主要关注点之一是开发、验证和确认软件单元的工作流程。软件单元是应用程序的最小可测试部分。为了确保正确运行，必须单独测试。需求将映射到这些软件单元，或在这些软件单元内，并且将构建基于需求的测试用例，以彻底测试软件单元。因此，选择用于单元开发的建模结构时，需要考虑多个方面，包括：

- 单元可测试性
- 单元代码生成
- 单元复杂度
- 单元测试工作流程
- 单元可追溯性
- 单元说明文档

这些特点均表明需要依赖模型引用作为单元开发的主要建模模式。模型引用具备有助于取得目标成果的多个特性，包括：

- 代码生成——从模型引用到生成的源文件的一对一映射。
- 可重用性——模型引用可在集成模型的多个位置上使用。
- 可测试性——模型引用是构建测试框架的理想选择。
- 团队合作——模型引用可以让多个开发人员同时进行开发工作，并简化总体配置管理和版本控制流程。

图 3 显示了如何使用模型引用将功能拆分为多个可测试单元。

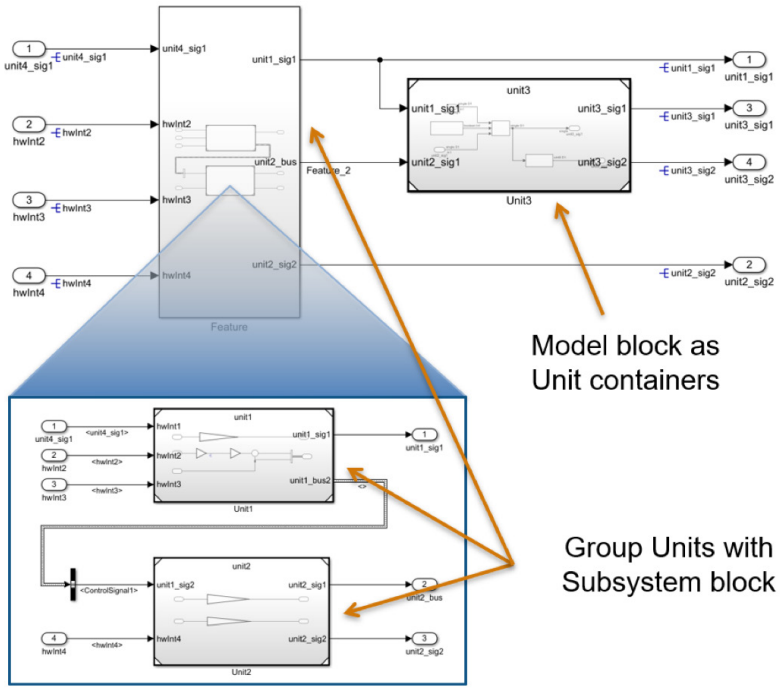

图 3. 将模型引用用于算法单元。

尽管没有将此方法作为最佳实践进行推荐，但 Simulink 库中的原子子系统也可用作软件单元。对于此方法，开发人员需要明确地管理和控制分区和测试边界。例如，接口需要锁定，意味着用户需要指定数据类型、采样时间和相关联的模块设置，包括代码生成选项，例如函数原型控制和文件分区。将原子子系统放置到不同的调度（例如单速率、多速率或多任务）中时，需要额外小心，并且需进行额外用户评估。在 2019a 版本中，Simulink 为可重用子系统引入了基于库的代码生成功能，以方便这类原子子系统的使用。此特性提供了许多新功能，可以更好地将原子子系统作为独立单元进行管理、设定、生成、测试和重用。

您也可以将子系统用作容器，进一步支持对类似单元进行功能分组。一般不推荐嵌套式模型引用，因为它会增加模型和数据管理的复杂度（请参见下一最佳实践了解详情）。

2. 选择策略将单元按特性分组

构建大型模型时，您可以从各种类型的建模结构中选择，以添加模型层次结构，例如：

- 虚拟子系统
- 原子子系统
- 模型引用
- 库模块

对于 ISO 26262，可以将这些单元模型灵活分组到它们各自的 ASIL 等级模块中。上一项最佳实践增加了一个“硬性”限制，即模型引用必须用于单元级算法。但是，单元分组可以是子系统，也可以是模型引用。需要做些权衡考虑，包括需要管理的模型引用的数量，以及您希望在功能级别上确保模型边界的牢固程度（不能频繁修改接口定义）。如果模型划分方式对架构来说无关紧要，而且用户群体接受使用更大的模型，就可以将这些单元归入虚拟子系统组。确定按特性进行模型划分的策略时，需要考虑：

- 生成的代码文件和功能分组
- 多个开发人员同时开发特性
- 模型引用文件相关的开销
- 特性分组在设计中是否轻松可见

在单元层和顶层之间的模型中间层，应当根据组织的建模规范和偏好选择建模结构。图 4 显示了可接受的使用虚拟子系统的方法。

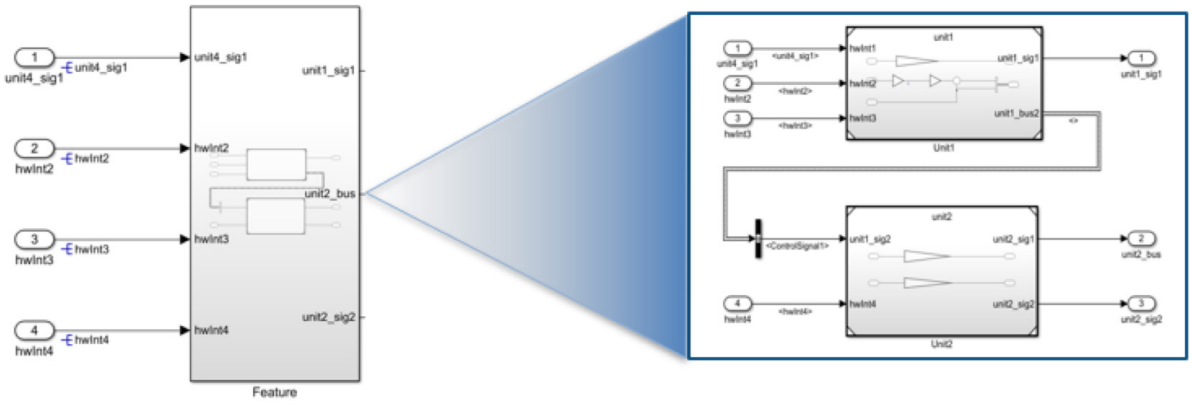

图 4. 将多个单元归入一个特性。

3. 在顶层模型中按照 ASIL 等级和 QM 等级进行拆分

ISO 26262 里有一个重要概念是互不干扰 (freedom from interference)。ISO 26262 有五个鲜明的安全等级（质量管理 [QM] 和 ASIL A–D），用于根据系统的功能安全方面对系统级和软件级功能进行分类。遵从 ISO 26262 的电子电气系统的组件可能分属不同的 ASIL 等级。例如，报告非严重诊断数据的算法部分可能归类为 QM 级别组件，而影响汽车制动能力的算法部分可能归类为更高级别的 ASIL 组件，这是因为如果发生故障，它造成的危害/伤害风险程度较高。

拥有多个 ASIL 组件的系统将因一个能将这些算法高效地划分为若干单独容器的架构而受益。其原因有二：

- 不同的 ASIL 等级可能有不同的开发、验证和确认要求。
- 对 ASIL 进行分隔和分段可免除干扰。

因为各个 ASIL 均可拆分，所以您应当选择一种建模结构来辅助拆分。使用模型引用，以便在部署算法时，每个 ASIL 的边界都有固定界限。因此，您应当将系统级模型的顶层拆分为多个模型引用，每个模型引用代表一个单独的 ASIL 等级的组件。请注意，在本例中，由于代码生成配置设置（请参见“代码生成配置”部分了解详细信息），系统级模型仅用于仿真，代码生成只能基于 ASIL 等级对组件单独完成。另一种选择是，在单元级别生成代码，然后集成代码。但是，在尽可能高的级别生成代码可以减少代码集成过程中的开销。

无论何时需要互不干扰，通过将模型引用用于每个 ASIL 或更大范围，都将为每个模型引用生成单独的函数和源文件。通过遵循此实践和代码生成配置最佳实践，每个部分都将拥有自己的源文件、共享代码和数据定义，更容易避免算法的不同片段相互干扰。图 5 展示了如何根据 ASIL 将模型进行分级拆分。

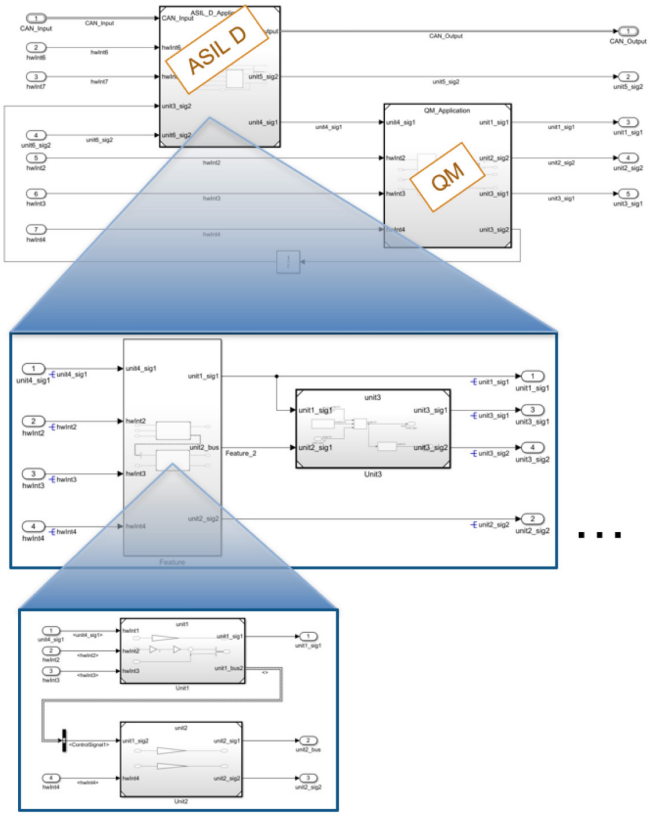

图 5. 基于 ASIL 的模型层次结构。

4. 集成级别不包括具体算法

ISO 26262 在具有代表性的架构中有多个测试级别的概念，包括单元级、集成级和系统级。通常情况下，软件单元必须根据目标 ASIL 等级进行各种级别的严格测试。例如，ASIL D 可能需要满足完整的修正条件和判定覆盖 (MCDC)，而对于 ASIL A 或 B，条件和判定覆盖或许可以接受。因此，单元是唯一应该实现算法功能的地方。如果算法内容出现在模型的集成级或系统级，可能更难实现设计的完整覆盖。所以，建议确保算法内容不会出现在单元级模型之外 (图 6) 。

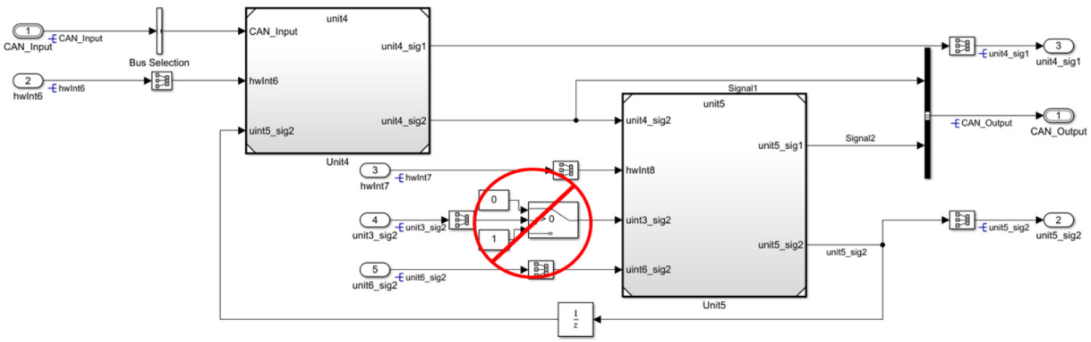

图 6. 避免算法内容出现在单元之外。

此最佳实践也可以对应到 MathWorks Automotive Advisory Board (MAAB) 规则 db_0143：模型级别的相似模块类型。

5. 使用模型指标监控单元复杂度

许多公司在开发周期后期发现，他们的算法验证难以达到 ISO 26262 推荐的覆盖度要求。其根源通常是在设计时没有考虑架构，在开发过程中没有管理单元模型的大小和复杂度。解决此问题的方法之一是，为整个开发团队设定单元大小指标阈值。这些阈值可能包括：

- 每个单元的最大输入和输出数
- 可重用的库
- 圈复杂度
- 元素数

要管理单元模型的复杂度，需要在开发周期的模型审核过程中审核这些指标。这将使得开发过程中的算法复杂度和范围可见。Simulink Check™ 中的 Model Metrics Dashboard（图 7）可以让此过程变得更简单。

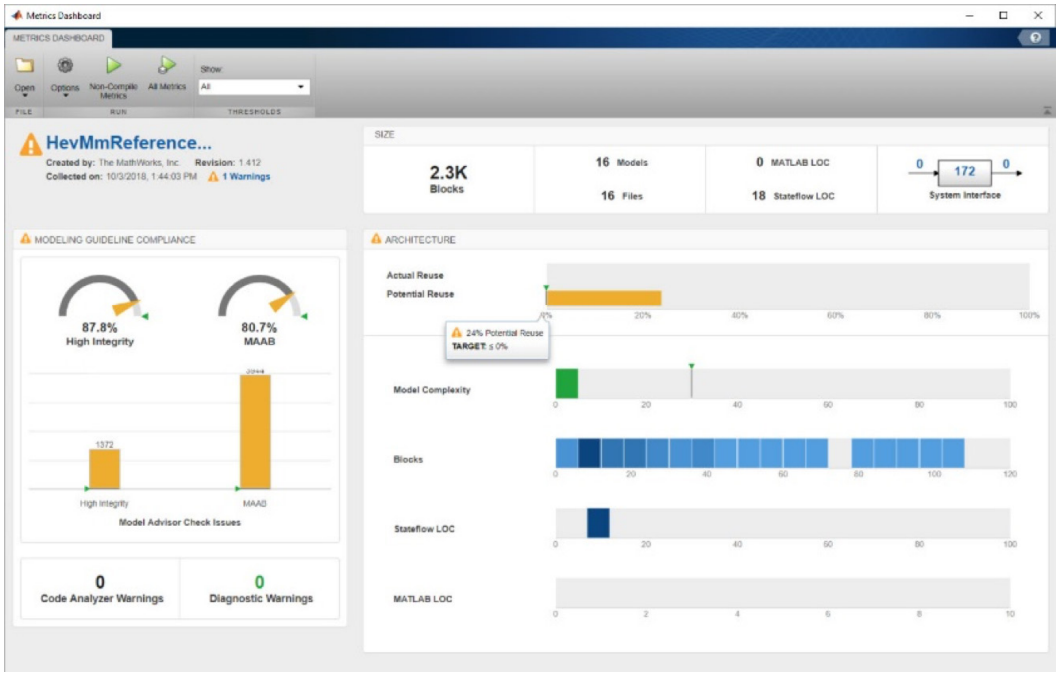

图 7. Simulink Check 中的 Model Metrics Dashboard。

Model Metrics Dashboard 可以让建模人员和模型审核人员轻松查看模块总数、MATLAB 代码行数 (LOC)、Stateflow® LOC、模型复杂度、模块数以及其他指标。控制板通过连续监控模型为设计工作流程提供支持，使您得以保证模型不被拆分为太小的单元（增加管理复杂度）或太大的单元（无法轻松测试，难以重用）。

阈值通常存在争议。这是因为往往无法从模型的单个方面管理模型复杂度，通常需要您分析多个指标，才能做出有意义的决定。例如，一个 Stateflow 模块的圈复杂度非常高，而它的模块数只有一个。

最近，MathWorks 和 Delphi Technologies、Bosch、PSA、Renault 和 Valeo 等几家汽车公司联合发表了一篇论文 Model Quality Objectives for Embedded Software Development with MATLAB and Simulink （使用 MATLAB 和 Simulink 开发嵌入式软件之模型质量目标），就指标和阈值提出了相关指导说明。例如，模型圈复杂度应为 30 或更低，模型元素数量应小于 500。

每家公司都有不同的阈值，MathWorks 咨询服务部门可以在这方面提供相关支持。

信号路由与接口定义

ISO 26262 第 6 部分针对单元和组件之间的接口复杂度和数据交换列出了许多注意事项。例如：

- 表 3 - 1b：软件组件大小和复杂度限制
- 表 3 - 1c：接口大小限制
- 表 7 - 1g：数据流分析
- 表 7 - 1k：接口测试

要实现这些目标，为单元、组件和不同 ASIL 组件之间的数据交换方式确定架构性策略非常关键。在与各个工程设计团队合作的同时，MathWorks 编写了多项最佳实践，用于减少分析单元级模型、组件和各种 ASIL 之间的数据接口交换所需的工作。本部分介绍了四项用于管理接口复杂度和数据交换的最佳实践。

6. 按 ASIL 等级、特性和速率对总线信号进行分组

ISO 26262 第 6 部分表 7 - 1g 建议开发团队执行数据流分析。要了解信号如何通过软件算法向下传递到单元层，数据流分析必不可少。这种分析可以根据提供方的特点，发现信号要求发生冲突的位置，或者不应直接使用各种信号的位置。要执行这种分析，必须制定总线层次结构策略，让开发人员更容易了解信号来自哪里，信号有哪些特点。如果您不指定如何对总线信号进行分层分组，则会出现下列问题：

- 总线划分和建模模式效率很低
- 开发人员之间总线分组不一致
- 拆分和重新创建总线时难以建模
- 代码生成效率很低

就像模型架构需要由上而下的设计方法一样，总线层次结构也需要这种方法。要管理每个信号的 ASIL 等级，需要根据任务速率和 ASIL 等级对总线层次结构中的信号进行分组（图 8）。根据 ASIL 等级对这些信号进行分组后，更容易确定信号提供方，以及确定信号是否正在 ASIL 等级更高的单元中使用。例如，如果信号由 QM 组件提供，但由 ASIL 组件使用，就需要执行额外分析，确保分析和考虑对此信号的依赖性。

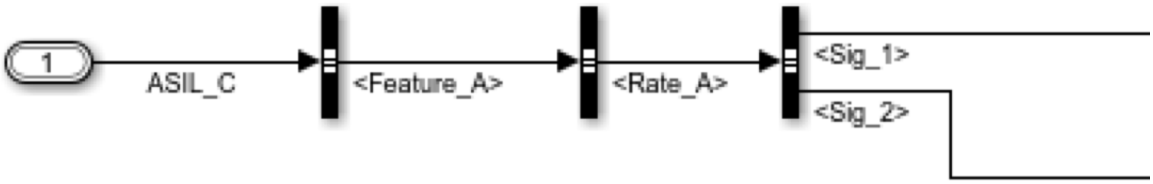

图 8. 对总线层次结构中的信号进行分组。

7. 仅向单元传递需要的信号

ISO 26262 第 6 部分表 3 和 表 7 就单元级接口和数据交换提出了重要建议。这两个表指出，应当尽可能减小软件组件接口的大小并降低其复杂度。此外，在确认过程中，用户必须执行接口测试。如果不必要的变量向下传递到单元级模型，则需要执行附加测试，确保这些信号不会影响单元。减少传递到单元级的输入有助于解决此问题。为此，您可以在总线信号进入单元级模型之前，将总线信号拆分出仅在对应单元中使用的信号（图 9）。

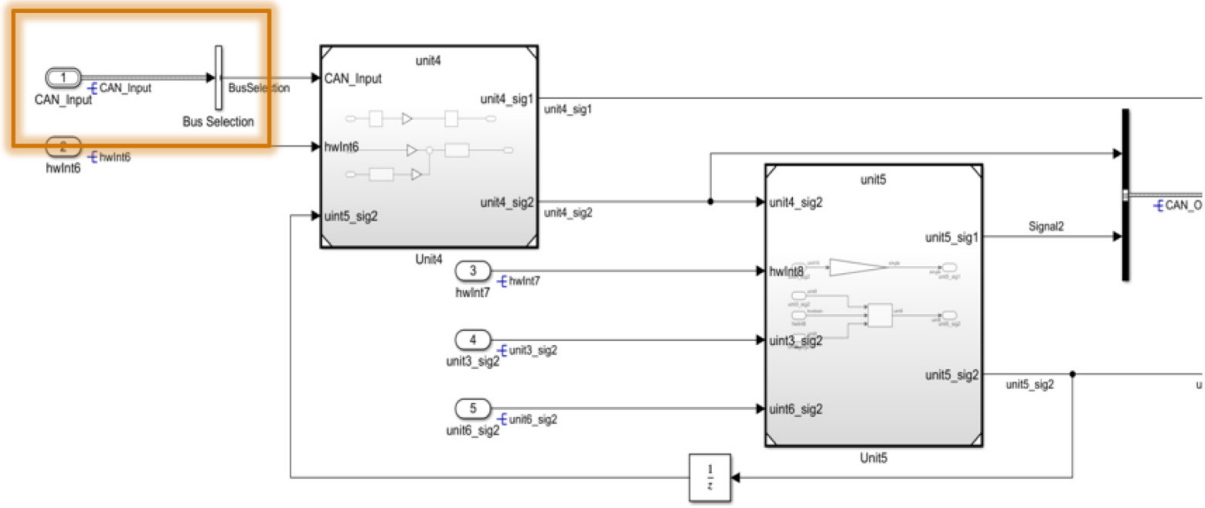

图 9. 拆分进入单元级模型的总线信号。

8. 优化信号和参数对象的布局

信号和参数对象的高级用例是定义模型和基础软件之间的接口。在此类用例中，参数对象通常用于指定标定量，并放置在模型模块中，例如 Gain 模块和 Lookup Table 模块中。信号对象的使用比较复杂，但它通常与内部信号相关联来支持标定工作，或者定义在根级的输入和输出端口上（图 10）。

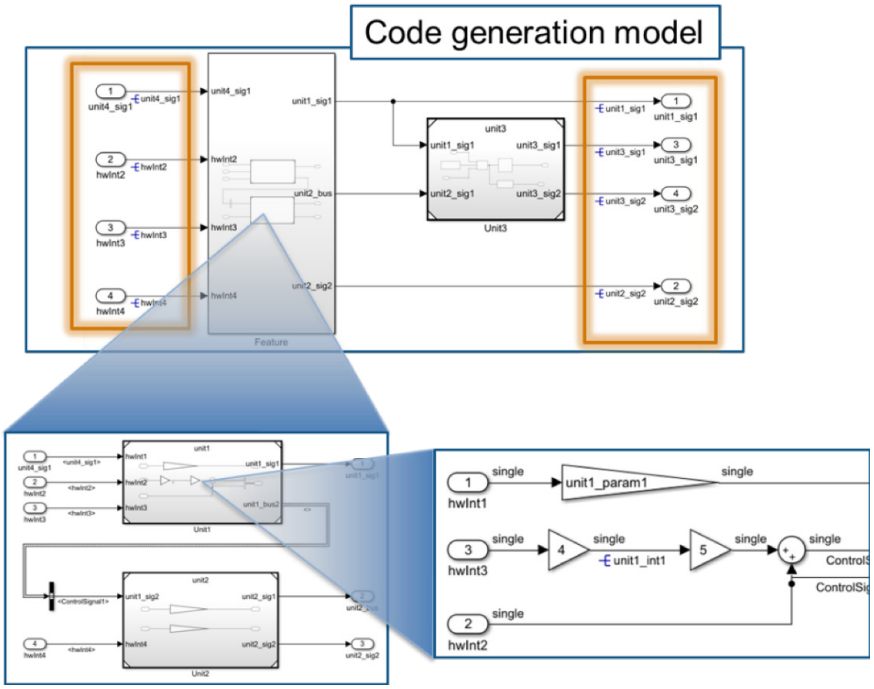

图 10. 放置信号和参数对象。

请务必注意，模型引用模块接口不包含信号对象。这是因为单元级边界上的信号被视作内部接口信号。这也假设正在最高 ASIL 分区生成代码，如上文最佳实践 6 中所述。对于内部接口信号，最好让 Embedded Coder® 根据其内部优化算法定义这些信号。但是，如图 11 所示，数据类型这类信息，确实需要在模型引用的接口定义处以端口配置的方式明确下来。

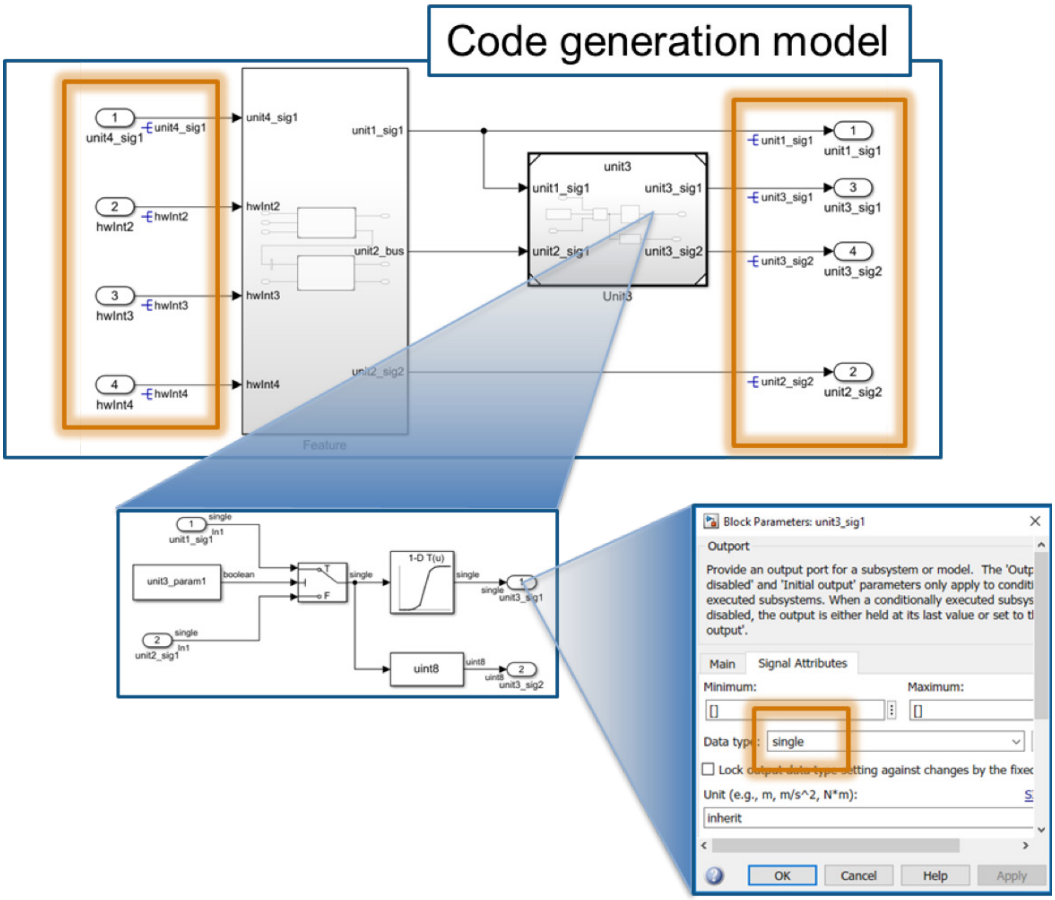

图 11. 定义端口的数据类型。

这项最佳实践要求将接口信号对象放置到代码生成模型边界。它还可以确保在为每个 ASIL 组件生成代码时，根级的输出端口可以正确指向其他 ASIL 组件对应的接口。

信号和参数对象的存储类，可以根据软件架构和编程实践去定义。可能的例外是，不同 ASIL 等级或者不同分区的模型之间的数据交互，需要加以保护，以满足免干扰的设计要求。

9. 对不同 ASIL 等级模块之间交换的数据进行保护

当为不同 ASIL 等级的模型单独生成代码时，有一点需要注意，即需要采取适当的方法来保护不同 ASIL 等级的功能模块之间的数据交换。关于在 ASIL 部分的根级输入和输出端口上使用存储类，目前已有多项策略。有一种普遍的方法是，使用数据对象的存储类设定为 Get 和 Set 函数，通过定义 Get 和 Set 函数，您可以为这些接口增加额外的保护，确保只有有权限的软件组件才能访问数据。

图 12 举例说明了如何才能在根级输入端口和对应生成代码上使用 GetSet 存储类。

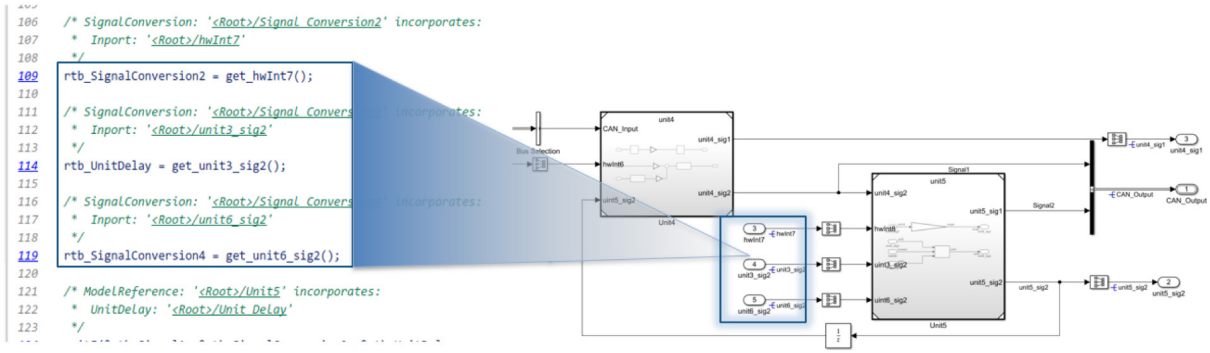

图 12. ASIL 组件之间的非 AUTOSAR 接口

图 13 说明了如何配置其中一个存储类，以确保 ASIL 之间的 Get 和 Set API 匹配。

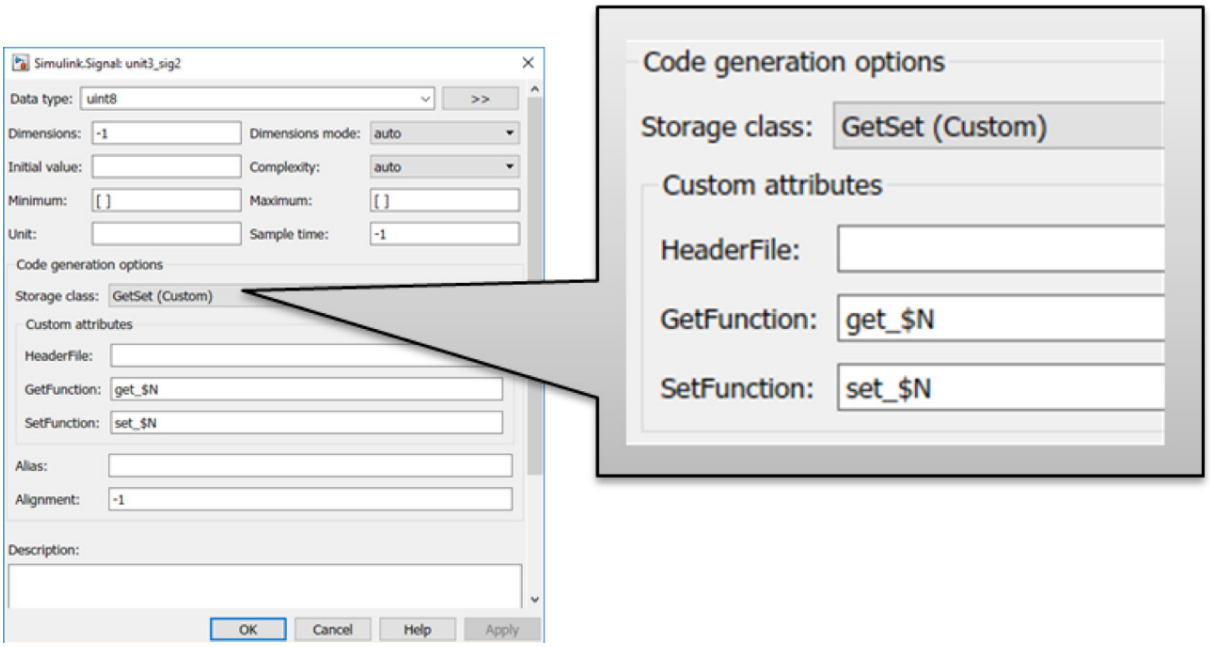

图 13. 配置根级 I/O 存储类的代码生成选项。

值得注意的是，GetSet 存储类对象仅为接口保护提供一个切入点。保护的具体实现通常通过由手动编码实现的底层软件完成，可使用 GetSet 存储类轻松集成。

代码生成配置

根据上述最佳实践开发模型后，依然需要特别注意代码生成配置设置，才能实现 ISO 26262 设定的目标。本节介绍了从代码布局和共享代码文件分隔等方面设置代码生成配置的最佳实践。下面列出的配置再次假设已遵循上述最佳实践。

10. 确定代码布局策略

ISO 26262 中的关键设计理念之一是不同 ASIL 等级模块之间互不干扰。要保证特定 ASIL 等级上的某个系统部分出现问题时，不会对另一 ASIL 等级上的功能造成影响，互不干扰必不可少。例如，如果 QM 或 ASIL A 组件出现问题或发生故障，设计应当将此功能与 ASIL D 等级的功能分隔开来，确保 ASIL D 等级的功能继续运行。在嵌入式系统中，有一种类型的错误值得注意，即应用的某些部分是否有权限访问它们不应访问的内存片段或函数。解决此问题的方法之一是，将功能拆分为单独的内存片段或者单独的微处理器核心。通过告知编译器应将变量和函数放入哪个存储区域，可以完成这项操作。图 14 展示了分隔应用的方法。

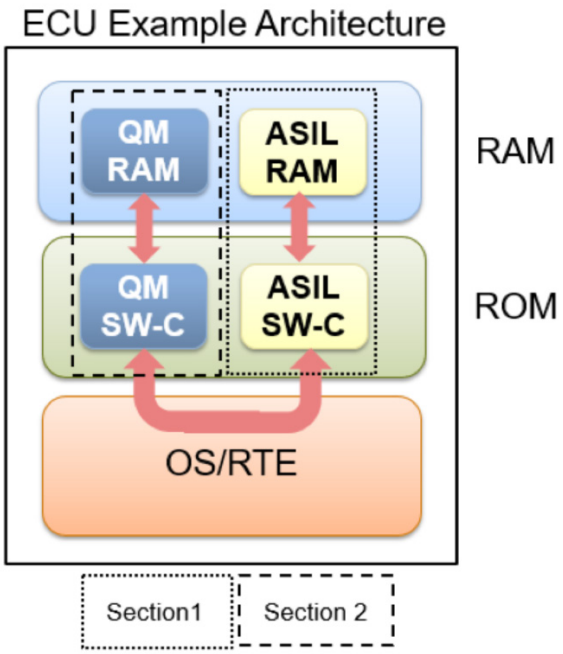

图 14. 对系统架构进行分段，确保互不干扰。

另一个方法是，将各个 ASIL 等级和 QM 等级拆分为单独的内存分区。拆分内存分区可以解决不同 ASIL 等级彼此不按预期交互的问题，例如，QM 等级的软件组件错误地读写受保护的 ASIL 等级的 RAM 分区。要完成此配置，可以在配置选项中选择内存分区，如图 15 所示。虽然不一定要使用此方法，但它的确可以简化整个工作流程。

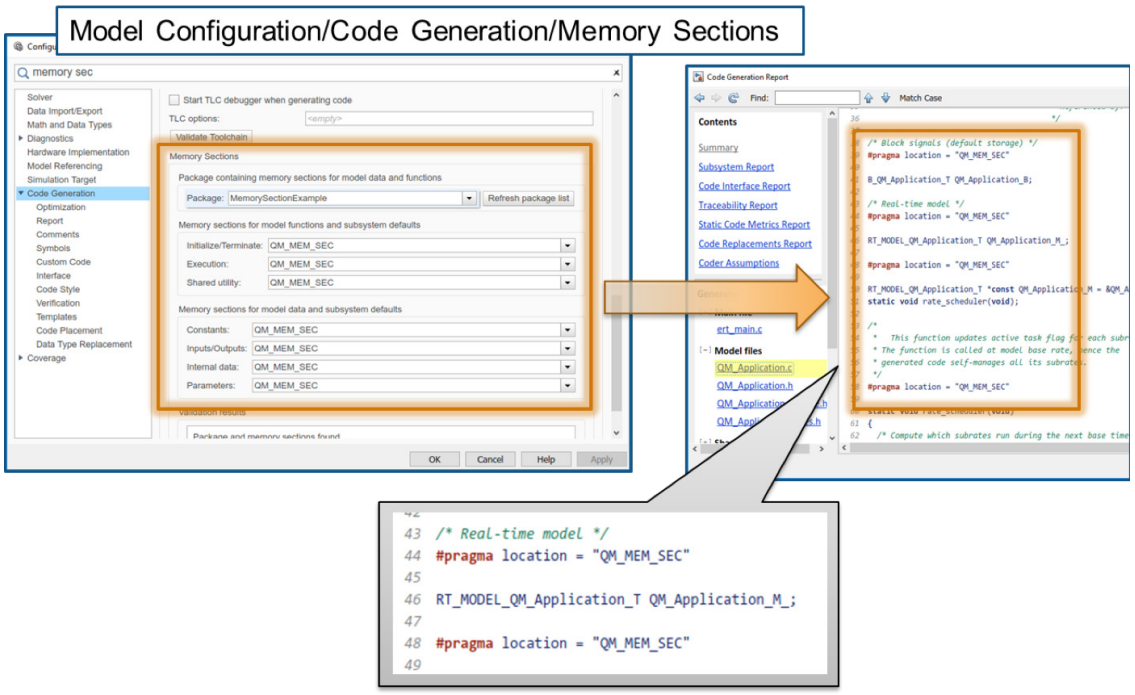

图 15. 配置内存片段的参数设置。

11. 共享的代码使用不同的名称标记

Embedded Coder 可以生成称之为 Shared Utilities 的共享代码文件。这些文件是一些基本函数（库函数），可能会跨过不同模型，在生成的代码中共享使用（函数名相同）。此方法会给与安全相关的应用造成问题，因为共享代码文件的来源没有任何差别。为利用分段概念将代码编译成单个应用，必须为不同 ASIL 等级功能生成独立的共享代码。此操作可通过代码生成配置完成，如图 16 所示。

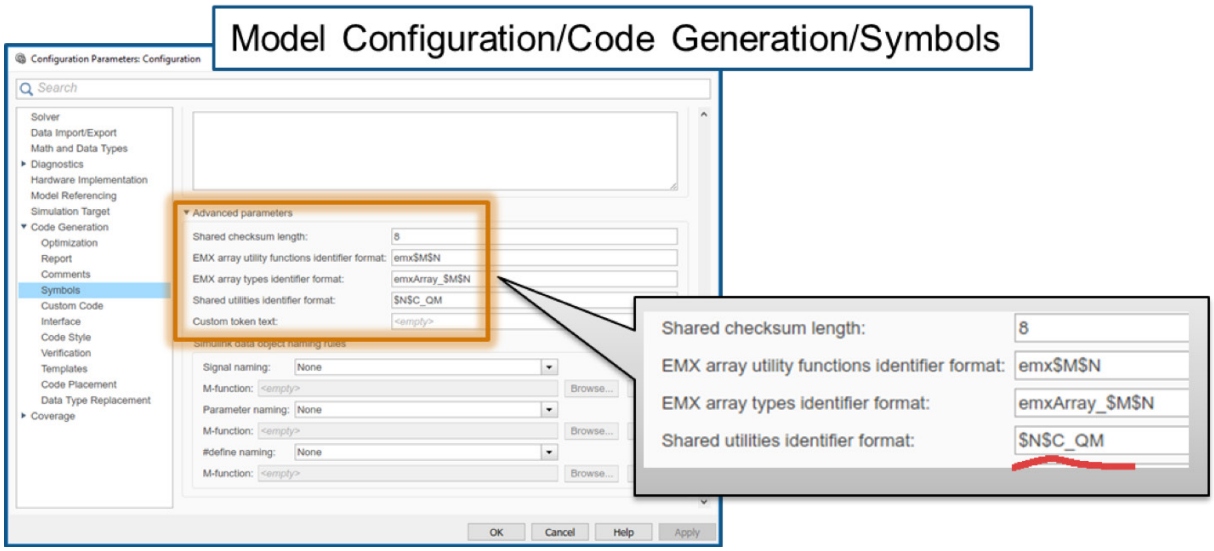

图 16. 配置共享代码设置。

有了上述设置，您就可以创建生成具有唯一标识符的共享代码。例如，根据上述 QM 后缀配置时，Lookup Table 模块的共享代码如图 17 所示。

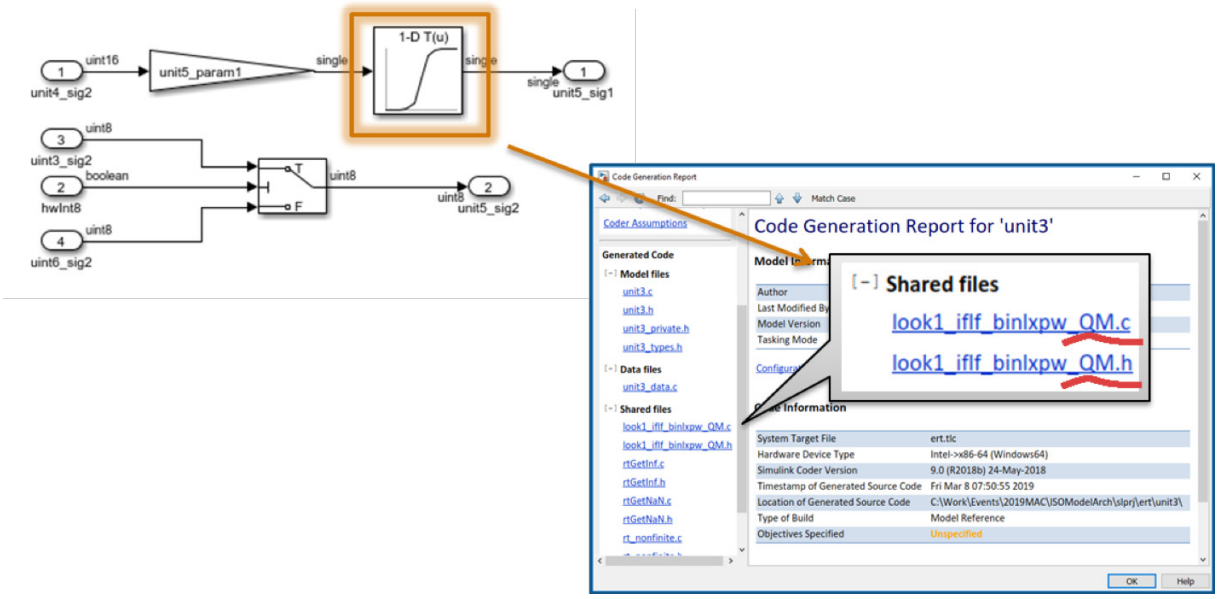

图 17. 为 QM 配置的共享代码。

小结与后续工作

本文档所呈现的结果是通过多个 MathWorks 咨询项目总结编写的最佳实践。事实证明，这些最佳实践可以推动 ISO 26262 的普及。但是，遵循这些最佳实践并不保证遵守 ISO 26262，因为它们只能满足部分 ISO 26262 要求，而且每个应用都具有独一无二的需求。

关于运用上述最佳实践来遵守 AUTOSAR 标准的后续工作正在进行中。*索取白皮书初稿。*

了解更多

- *MATLAB 和 Simulink 中的 ISO 26262 支持* - 概述
- *ISO 26262 流程部署咨询服务* - 咨询服务