# SIL 测试的一个典型报错

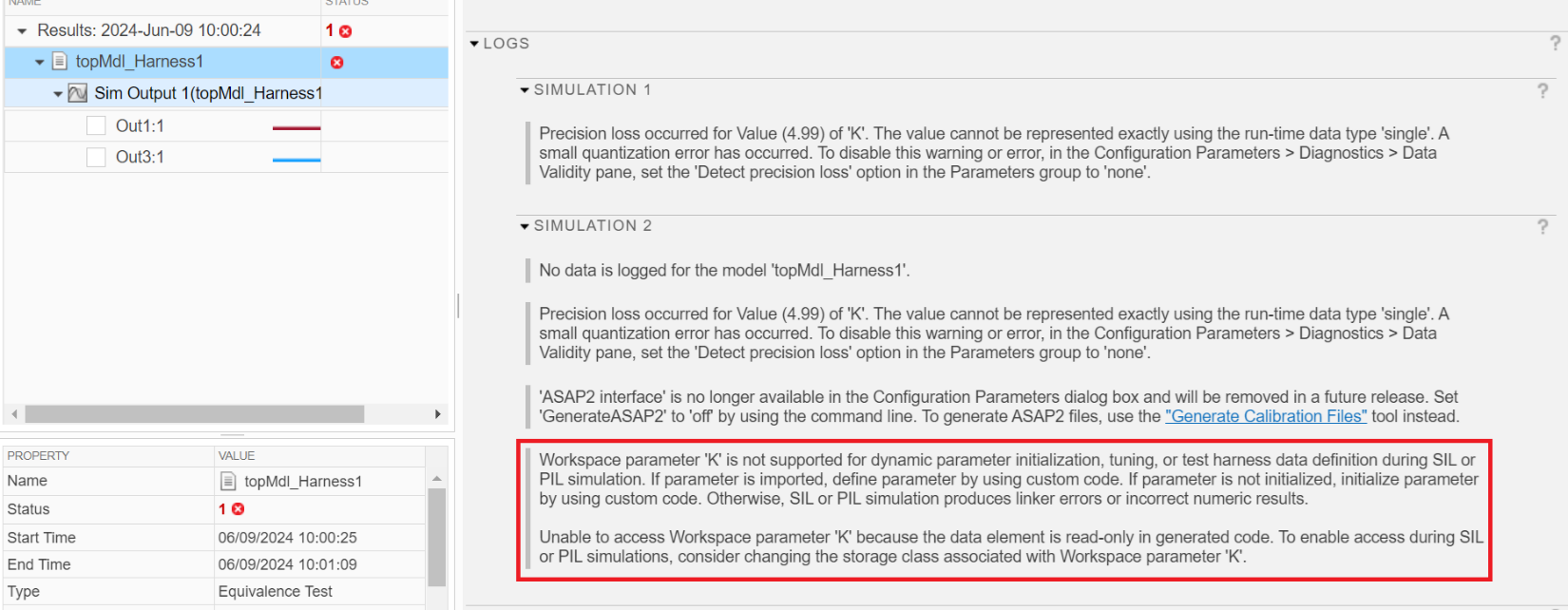
## 背景

这里结合一个 Simulink Test 用户遇到的问题、配合文档信息，探讨一个 SIL 测试的限制。

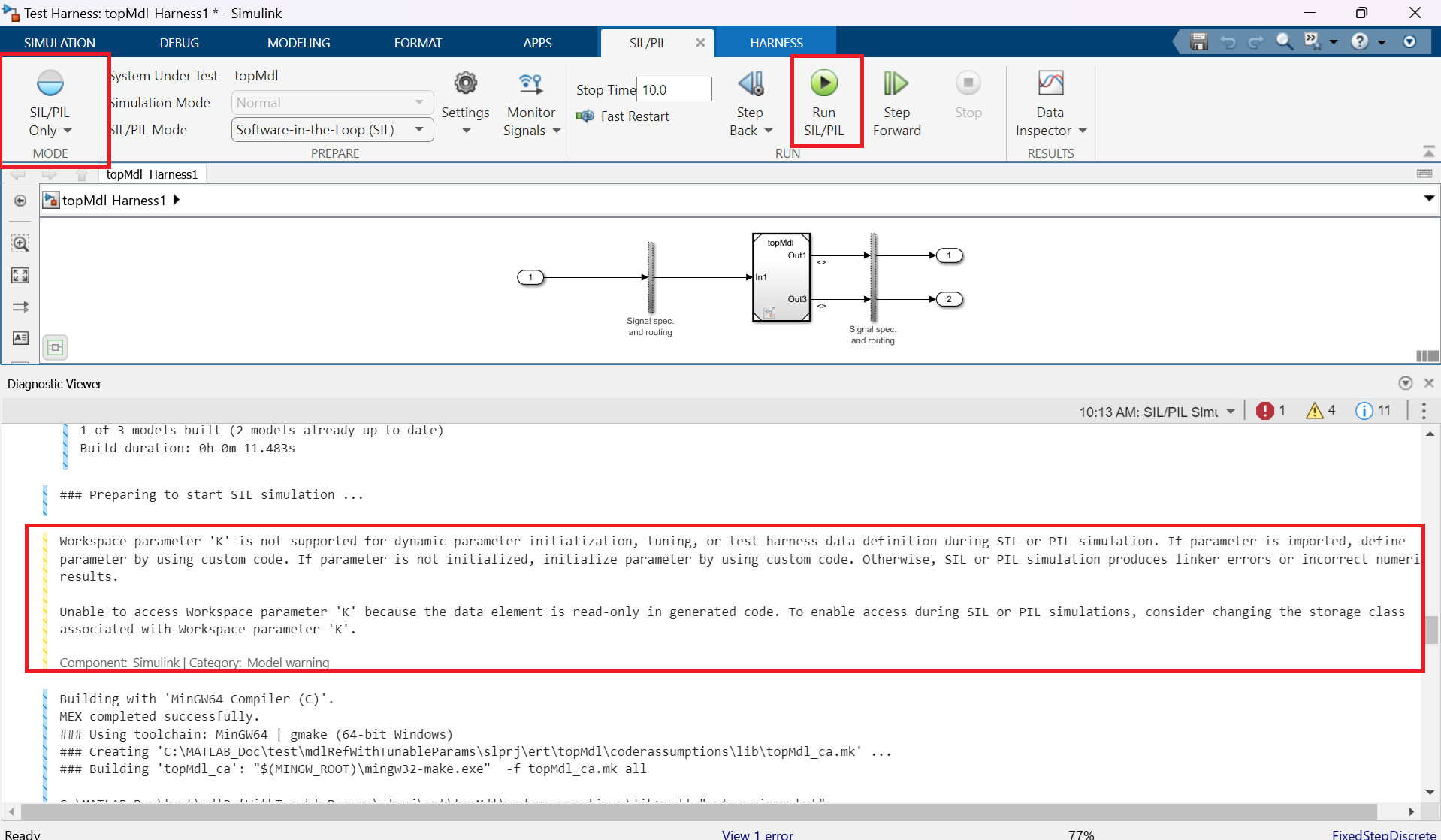
问题描述如下：

进行背靠背测试时， SIL 测试报错，报错信息提示某些参数不可调，必须由 Custom Code 来进行定义，否则会遇到 linker 报错：

Workspace parameter 'xxx' is not supported for dynamic parameter initialization, tuning, or test harness data definition with SIL or PIL. If this parameter is imported, you must explicitly define and initialize it using custom   
code. Otherwise, SIL or PIL simulation produces linker errors or incorrected numeric results.



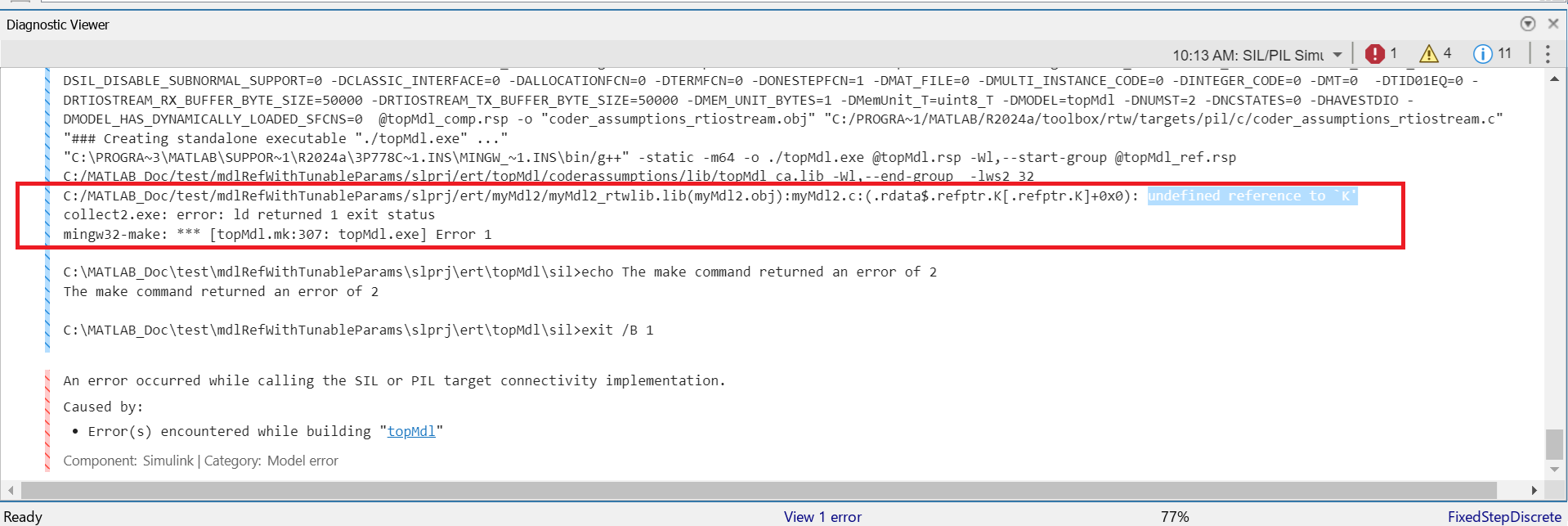
这个提示看起来只是一个 warning ，但是当我们去 harness 模型中打开 SIL/PIL Manager，尝试运行 SIL 测试，



就可以看到 linking 过程果然触发了报错，报这个对应的参数没有定义——以下截图中第一张是提示参数不可调的 warning，第二张展示了编译报错，这也是 Test Manager 中 SIL 测试没有成功的原因：

A screenshot of a computer

Description automatically generated



## 问题触发的条件

### SIL 仿真的限制

前面的 warning 信息与 SIL 仿真的一个限制有关：模型（或其包含的引用模型）中包含标定参数，其存储类为 Const/ConstVolatile 或者以此作为基础的自定义存储类时，SIL 仿真模式下无法对该变量进行定义、初始化、或者调校。

以下参考文档归纳记录了这个限制：

[SIL and PIL Limitations - MATLAB & Simulink - MathWorks 中国](https://ww2.mathworks.cn/help/ecoder/ug/sil-and-pil-simulation-limitations.html?#brydbkh)

A screenshot of a computer

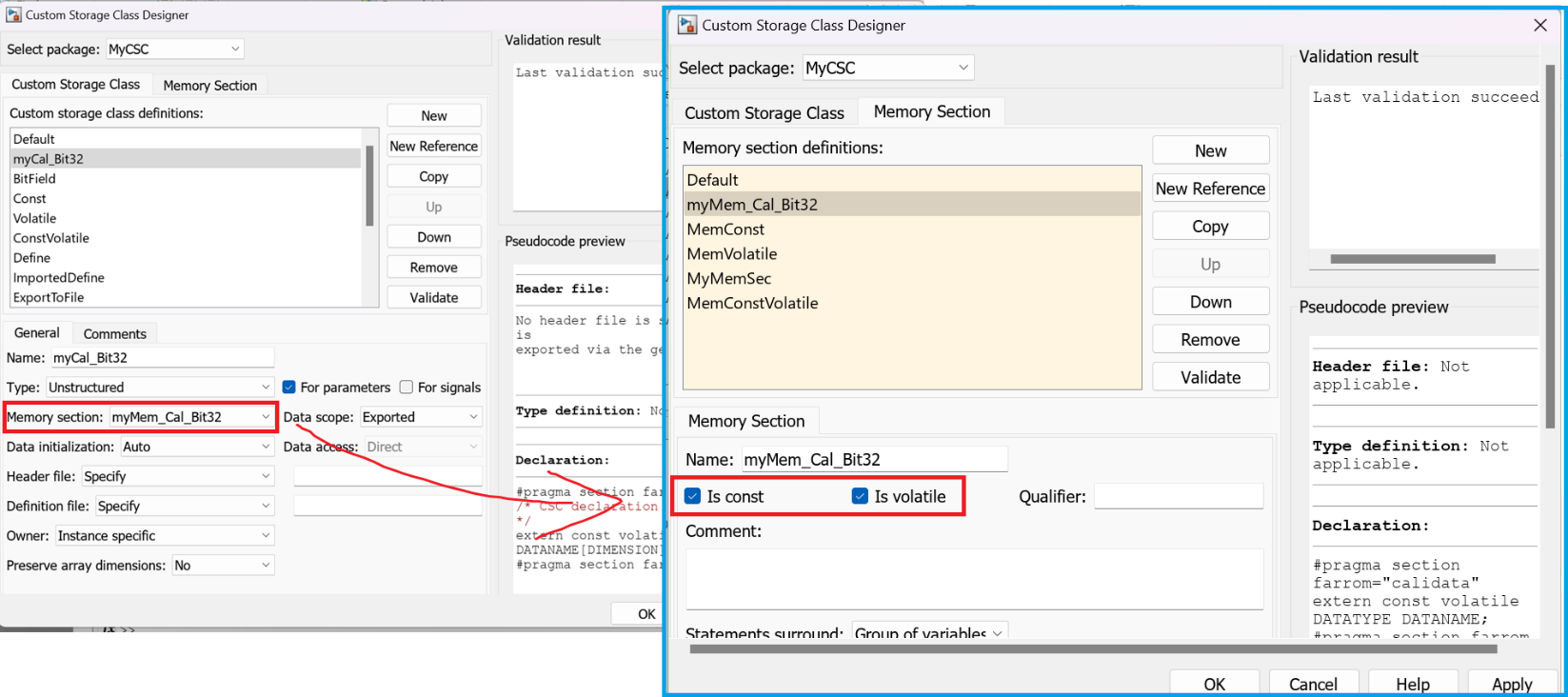
Description automatically generated

本例中用于复现的 myMdl2.slx 中使用了标定量参数 K，定义如下：

A screenshot of a computer program

Description automatically generated

此处为了复现原始的问题所以将 K 的存储类设置为自定义存储类（具体的存储类和 Memory Section 定义可以看以下截图），实际上如上一节描述， Simulink 自带的 Const 、 ConstVolatile 等存储类一样可以触发这个问题。



### 在涉及 harness 模型的 SIL 测试中报错的原因

究其原因，按照我的理解，是在SIL 测试需要对模型生成代码，而被测模型是以模型引用的形式包含在 harness 模型中的，故而如果使用的是常规的 harness 模型，生成代码时采取的方式是 model reference，这影响了变量 K 在代码中的定义生成的位置——也就是：



这句代码。

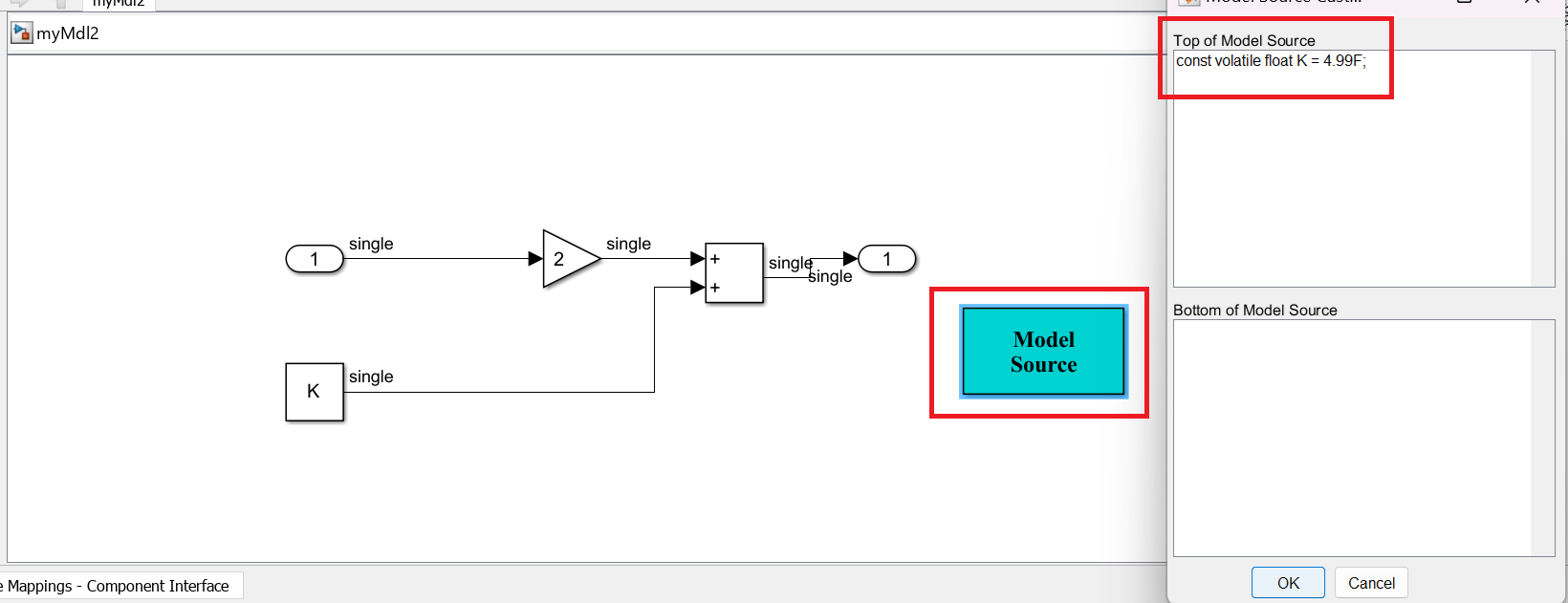
在不做多余设置的时候，K的定义会生成在顶层模型代码中，而非被引用模型的 .c 中。

正常模型编译时由于顶层模型会生成对应的 .c 文件，但在Normal 模式的 harness 中进行 SIL 测试时没有常规的顶层模型，所以找不到参数的定义。

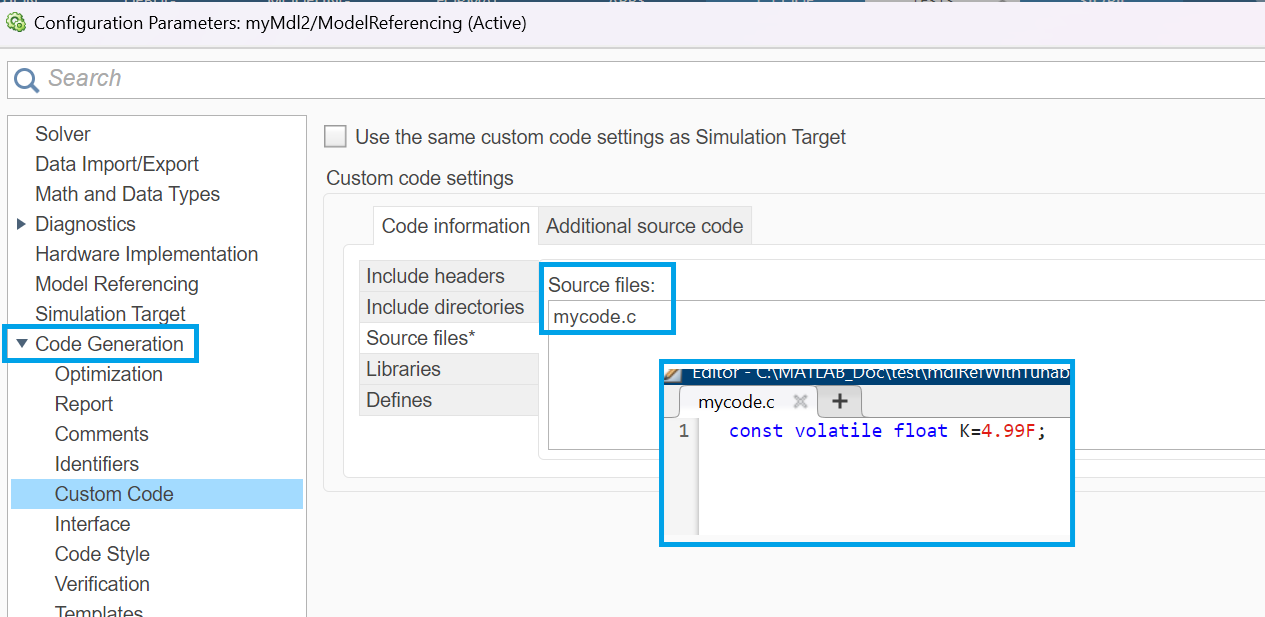
## 替代方案

针对这个原因，可以考虑以下几种方案：

【a】如报错中提示，利用 Custom Code 进行变量的定义，例如模型中插入 Model Source 模块，手动插入这句数据定义（及初始化）：



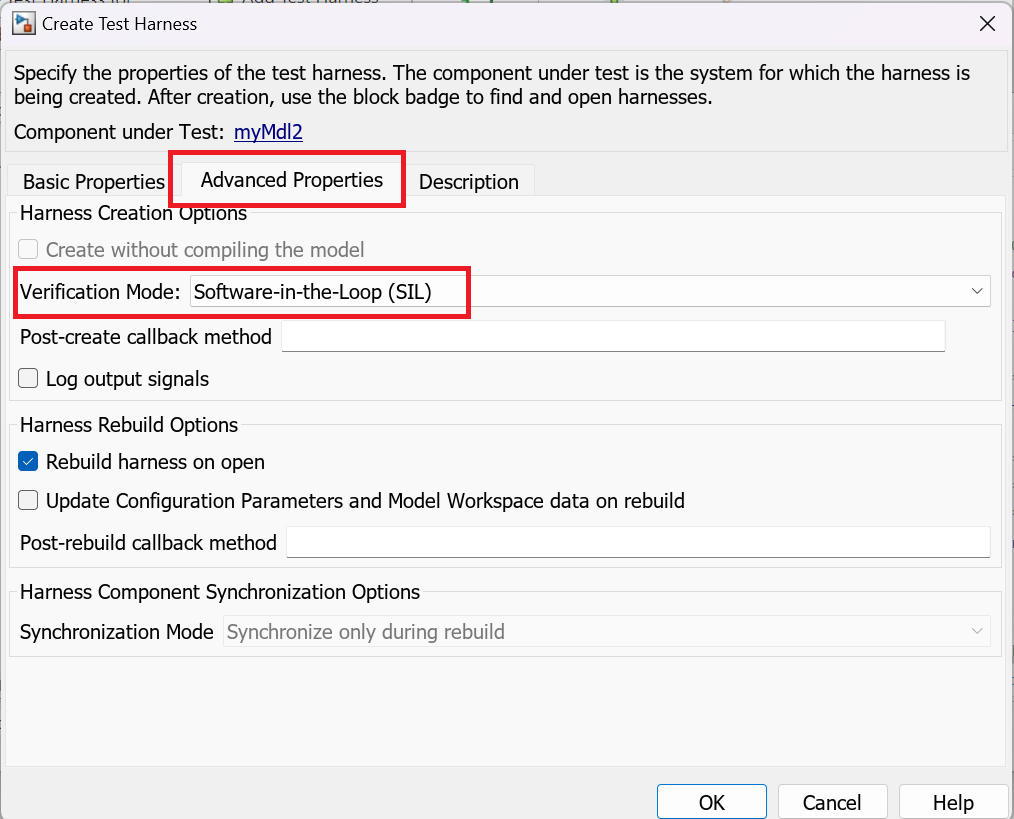
或者在配置中添加 Custom Code（注意由于是 SIL 测试，要影响代码生成行为，所以需要在 Code Generation 菜单下配置，而非 Simulation Target）：



注意，只能在使用参数 K 的模型本身中做这个操作才能规避报错；我在debug过程中做过的其他一些无用的尝试包括：在 harness 中想办法添加 Custom Code 、 为 K 指定变量声明和定义生成的文件等，由于 SIL 测试中并没有常规的顶层模型对应的 .c 文件生成（此时“顶层模型”由 harness 模型顶替了），所以以上方案都不解决问题。

【b】使用 SIL Harness：

创建 Harness 时，在高级选项中，验证模式选择为 SIL

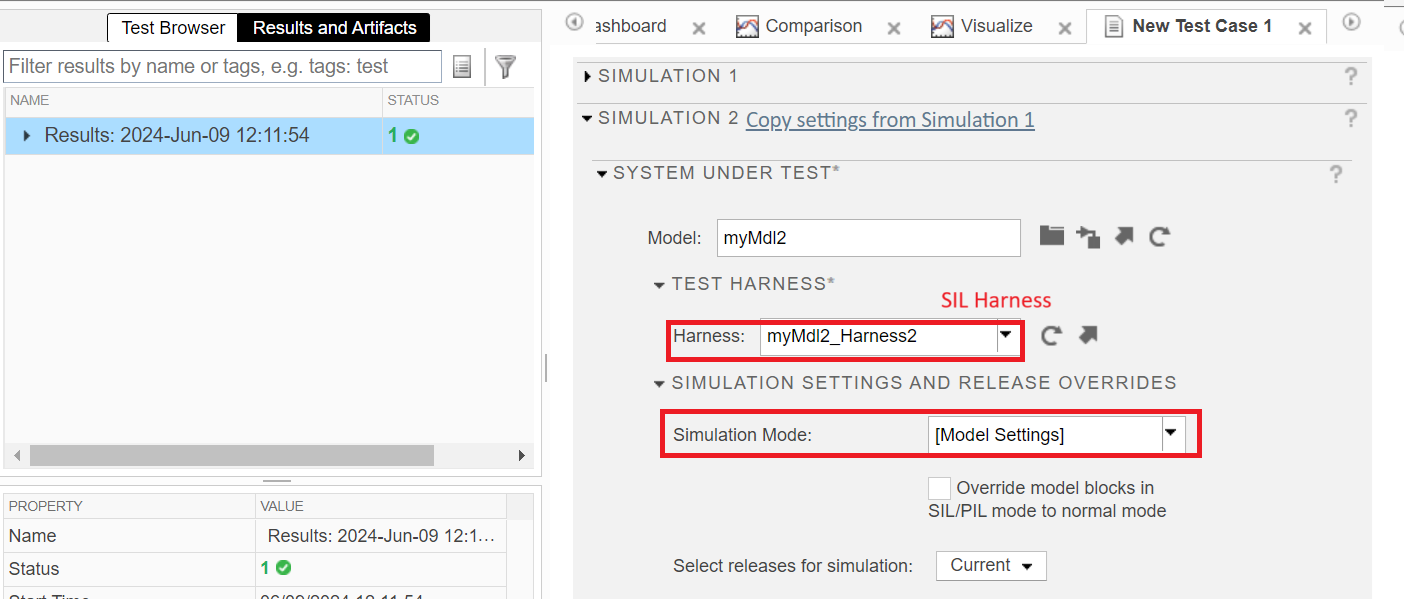


随后在 Harness 模型中以 normal 模式运行仿真即可（被测模型会自动编译为 SIL 模式）：

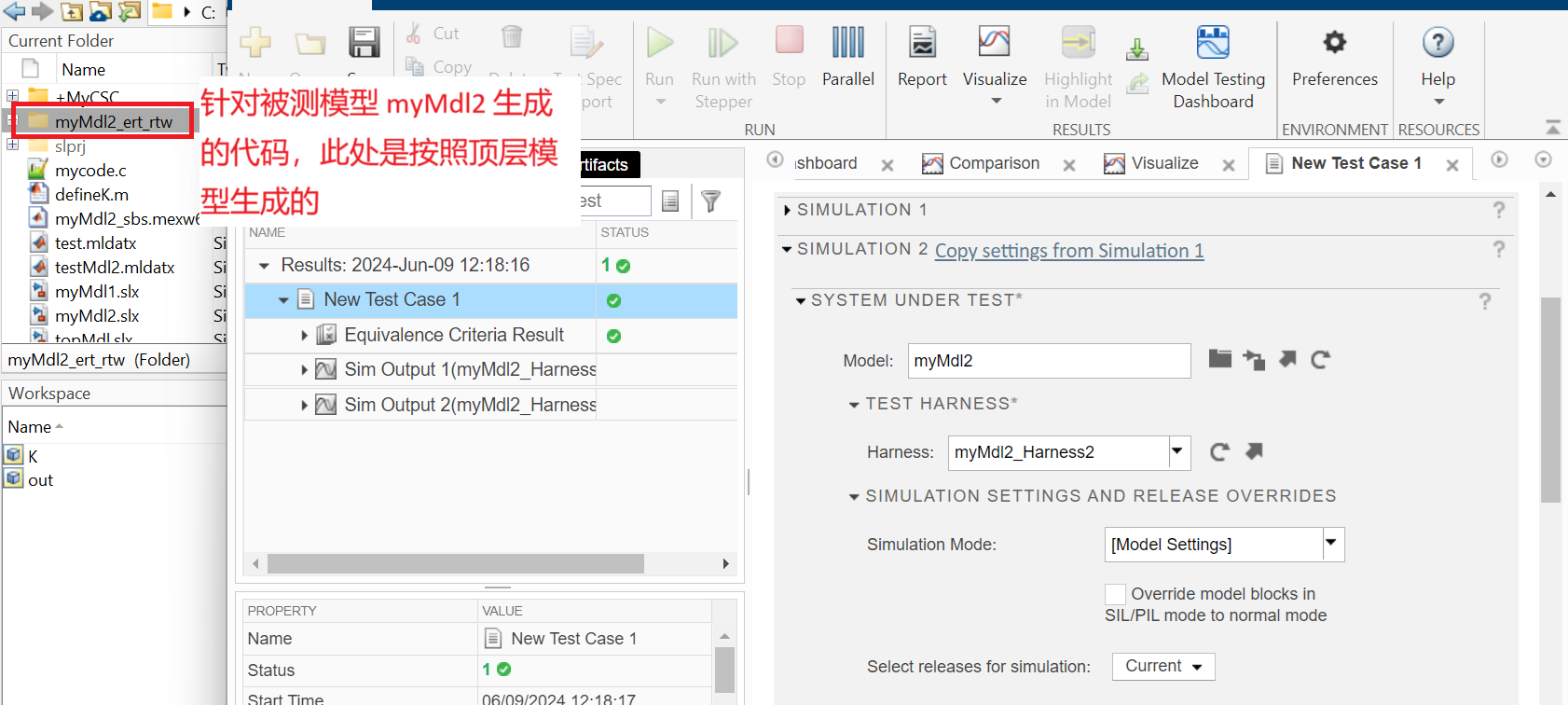
A diagram of a computer code

Description automatically generated with medium confidence

Test Manager 中，则需要把 Simulation 2 的 harness 修改为对应的 SIL Harness，仿真模式保持为 Model Setting 不变：

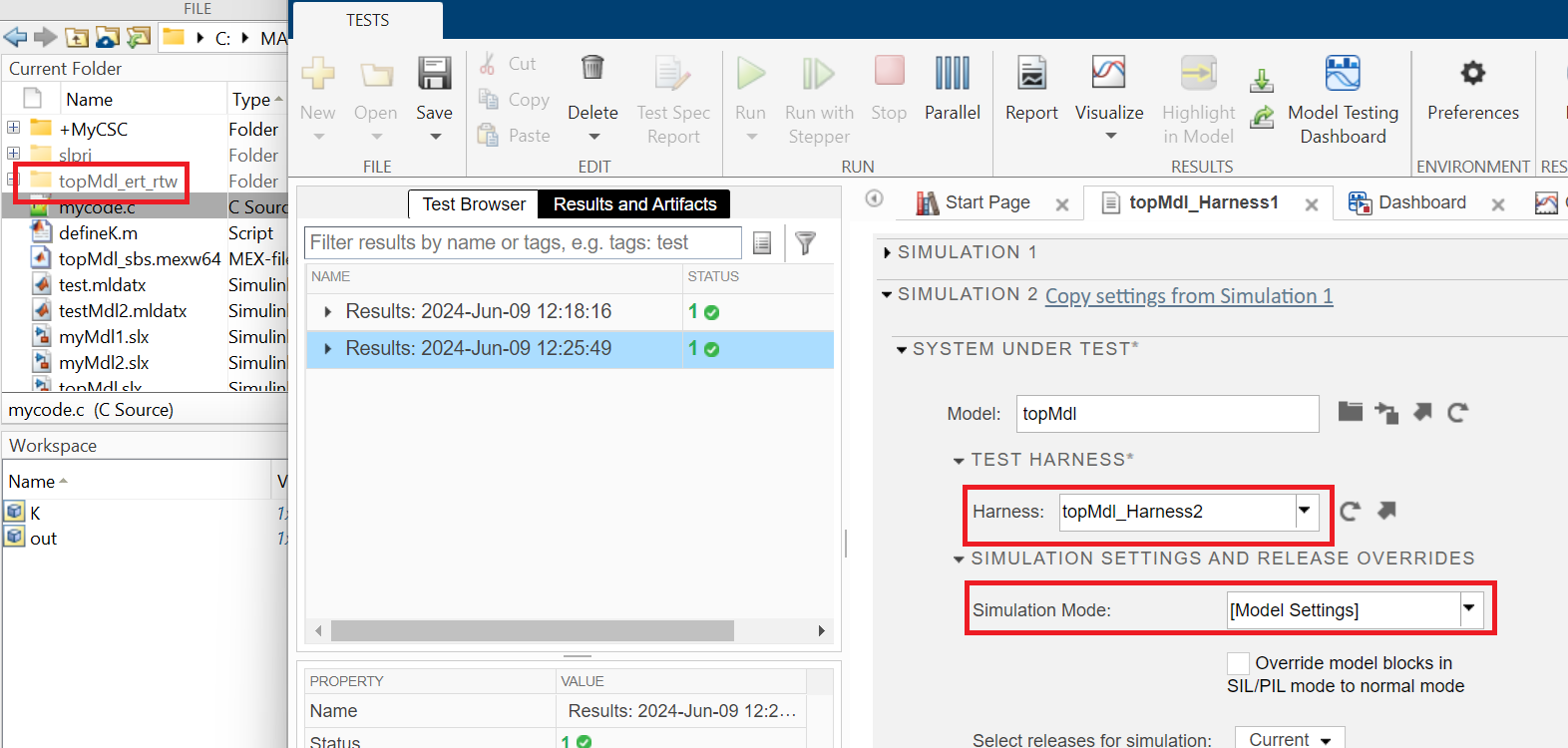


另外，仿真完成之后可以看到对应的 \_ert\_rtw 文件夹，说明被测对象是按照顶层模型的模式生成的代码，这也更符合 SIL 测试的精神：



另：

以下截图是复现原始报错的模型架构，即被测对象 topMdl.slx 包含模型引用 myMdl1.slx 和 myMdl2.slx，模型引用 myMdl2.slx 中又有标定量的情况，与包含标定量的单层被测模型的情况没有区别：



## 结论？

印象中使用普通 harness 进行测试在旧版本中还会触发其他报错。并且考虑使用 harness 模型去做测试的情境，在不改动被测对象、保证被测代码与最终产品代码一致的前提下，利用 SIL Harness 做 SIL 测试是否才是更为推荐的 workflow ？