

ECE MBD: Laboratory session

Numeric Sequence Lock

During these exercises you will be trained how to implement a state machine in Simulink, testing requirements and model coverage. Based on this result the algorithm shall be applied to the targeted hardware (Arduino) via code generation. All the steps performed will follow the MBD design procedure.

This laboratory unit considers:

- **Stateflow**
- **Signalbuilder**
- **Traceability of requirements document to model and vice versa,**
- **Model coverage analysis,**
- **Code generation**

As an example a numeric sequence lock shall be implemented. For the implementation consider the requirements document.

Preparation

As an example for the Stateflow implementation you can use the provided fuel indicator example and the corresponding help documents provided by Mathworks.

Laboratory work

Development process

1. Requirements:

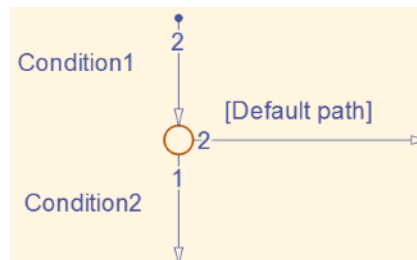
- a.) First read the requirements carefully and make an ordering of the requirements by priority. This is important as it may define the order for implementation!
- b.) Group the requirements to the categories: functional and none functional
- c.) According to your analysis are there any requirements missing or you would like to add?

2. Stateflow implementation:

- a.) Implement the state machine of the numeric sequence lock in Simulink, based on the requirements
- b.) Voltage Monitoring make a separate subsystem and use standard library blocks like, relational, logic, manual switch etc.

Recommendations:

- ✓ **Hint:** For improving the development process use breakpoints for debugging the Stateflow charts, therefore proceed as follows: Make a right click within the chart and select break point
- ✓ **Type cast:** Inside the state machine the values can be casted to the required data type by "uint8(value)"
- ✓ **Transitions:** beside the relational operator's transitions can be triggered only if a change happened: e.g.: hasChanged(u); Further operands are (hasChangedFrom, hasChangedTo), analyse the functionality of this operators first then implement it!
- ✓ **Junctions:** Figure 1 shows a possibility to check for two conditions and using a default path. In case condition1 is valid but condition2 not go to the default path.

**Figure 1: Junction****3. Testing:**

Test input: Use a signal builder block for preparing the input sequence entered to the algorithm. The numeric sequence lock algorithm has several execution paths therefore proper design of test cases is essential.

Consider: It is a rather hard task assuring a 100% coverage of all the states and conditions.

As a reference some test scenarios are explained in the "MBD_SeqLock_req_and_test.doc". Consider to complete this test plan in order to test the remaining requirements.

- a) Consider to split the test cases between positive and negative testing as analysed during the lecture
- b) **Prepare your test cases** using an EXCEL sheet (Test_Cases.xlsx) use different tabs for categorization of the different cases.
- c) **Import** these tests to the signal builder (within the signal builder block: File/Import from file)
- d) **Run** each test and analyse the results and correctness of the algorithm

Hint: The usage of the Data Inspector can support the testing

- ✓ **Question:**

Make a statement about the **coverage** of your algorithm

The following tasks can be seen as optional and have not to be considered for the report. In case you are curious about the tools provided by Mathworks for requirements and testing you may analyse the following points!

4. Linkage/Traceability of requirements:

Important: based on the following explanations apply the linkage of requirements only for the input scaling block.

Link the implemented requirements from the document to the model by using the tool as follows.

4.1. Links between Models and Requirements:

4.1.1. Source of requirements: Creating Bookmarks

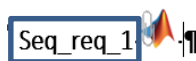
- a. Open your model.
- b. Open your Microsoft Word requirements document that contains bookmarks that identify requirements.
- c. Right-click a block in the model that you want to link to a requirement and select "Requirements/Open Outgoing Links Dialog"
- d. Click New
- e. Click Browse and navigate to the Microsoft Word requirements document that has bookmarks and click apply. (Consider that the document has to be on the Matlab path!)
- f. Open the document. The RMI populates the Document and Document type fields
- g. Click the Document Index tab of the Link Editor.

The Document Index tab lists all bookmarks in the requirements document, as well as all section headings (text that you have styled as Heading 1, Heading 2, and so on).

The document index lists the bookmarks in alphabetical order, not in order of location within the document

4.2. Create a Link from a Model Object to a Microsoft Word Requirements Document:

- a) Open the Controller requirements document select (mark) the text area to be linked (header)



The sequence for unlocking shall be 1,2,1,3 in this case an output flag SeqFound shall be "true"; otherwise "false".

- b) Open the model
- c) Select "Analysis/Requirements/Settings". The Requirements Settings dialog box opens.
- d) On the Selection Linking tab of the Requirements Settings dialog box:
 - o Set the Document file reference option to path relative to model folder

- Enable Modify destination for bidirectional linking

When you select this option, every time you create a selection-based link from a model object to a requirement, the RMI inserts navigation objects at the designated location

- e) In the model right-click a block to be linked and select Requirements Traceability > Link to Selection in Word. (Hint: That is working with all of the blocks also within the stateflow chart objects)

The RMI inserts a bookmark at that location in the requirements document and assigns it a generic name, in this case, for e.g: Simulink_requirement_item_1

4.3. Requirements visibility and navigation:

To verify that the link was created, select Analysis/Requirements/Highlight Model

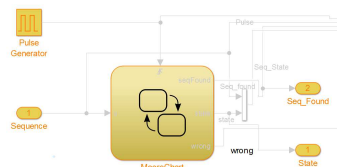
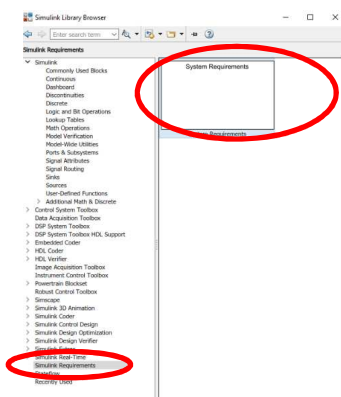


Figure 2: Highlighting of requirements

It is also possible to add the "System requirements" block from the Simulink library browser for showing the requirements (Consider: Only working on subsystems with links!!!)



Requirements document:

Sequence-Lock-requirements¶

Lock-requirements¶

Seq_req_1-The sequence for unlocking shall be 1,2,3,4

Seq_req_2-If the correct sequence is entered an output Seq_found shall be "true"

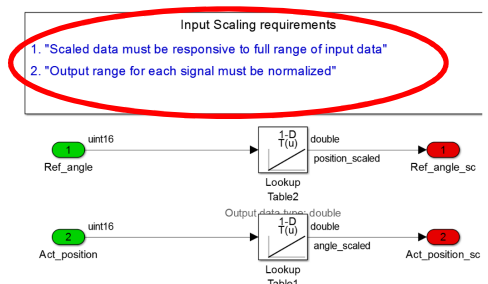
otherwise "false"

Seq_req_3-A wrong sequence entered for more than three times

shall cause a blocking for 5sec

Seq_req_4-In case the sequence value is "four" the system shall be locked again

Final Model:




To navigate to the link, right-click the block and select Requirements Traceability > "Seq_Lock_req1" => the section is displayed, selected in the requirements document. The same applies in the other direction from word to the model

✓ Question:

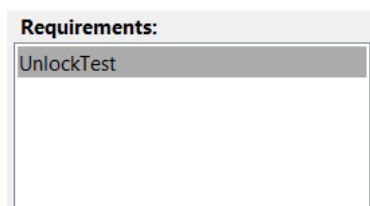
After finishing this section every requirement for the input scaling shall have a link to a model part.

4.4. Linkage of requirements to Signal Builder Block (Test Cases):

Links are attached to individual groups of signals, not to the Signal Builder block as a whole. Use this sort of links for test cases that are defined as Signal Builder groups.

At the far-right end of the toolbar, click the Show verification settings button  to display the Verification panel.

- a) A requirements pane opens on the right-hand side of the Signal Builder dialog box.



- b) Place your cursor in the window, right-click, and select Open Link Editor.

The Requirements Traceability Link Editor opens

- c) Click New. In the Description field, enter your test description
- d) When you browse and select a requirements document, the RMI stores the document path as specified by the Document file reference option on the Requirements Settings dialog box, Selection Linking tab.
- e) Browse to a requirements document and click Open
- f) In the Location drop-down list, select Search text to link to specified text in the document.
- g) Next to the Location drop-down list, enter User Input Requirement
- h) Click Apply to create the link
- i) To verify that the RMI created the link, in the Simulink Editor, select the User Inputs block, right-click, and select Requirements Traceability
- j) Right-click the link label under Requirements and select View to open the related requirements data. It can be even in a Microsoft Excel document. The Simulink icon in the linked cell allows navigation back to this signal group.

Via this method you can also link the requirements for testing/test plan, to your model and keep traceability

✓ **Question:**

Show the linkage of your different test cases to the requirement document test plan?

4.5. Traceability/Report:

For creating a report about the applied requirements select “Analysis/Requirements/Report/Generate Report”. The Requirements Management Interface (RMI) searches through all the blocks and subsystems in the model for associated requirements. The RMI generates and displays a complete report in HTML format

The report and its objectives can be modified by the settings menu “Analysis/Requirements/Settings”

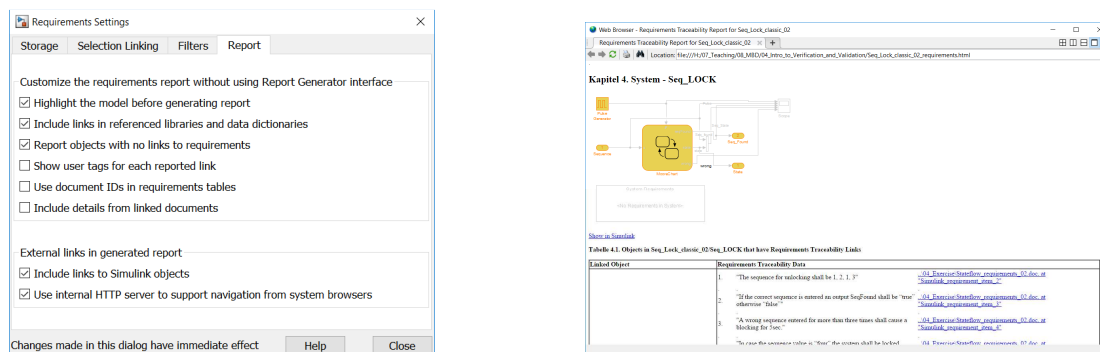


Figure 3: Settings and report

✓ Question:

Go through the report and identify and explain the different sections and their meaning

At the end of each simulation the running total is updated with the most recent results. By selecting the cumulative tab, the report over all the cases can be created.

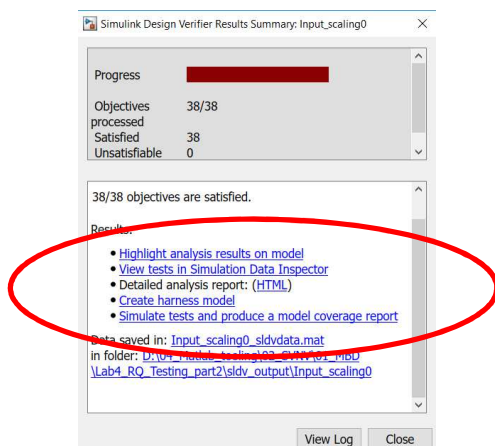
✓ **Question:**

Is the model covered by 100%? show and explain your result

- ⇒ If yes your test cases are well defined,
- ⇒ If not you have to rethink your test cases, or algorithm in order to get 100% coverage

4.6. Automatic Test Case Generation for increasing coverage

- a) Right-click the Input_scaling subsystem, and select Design Verifier > Enable 'Treat as Atomic Unit' to analyze. (See Appendix)
- b) The settings used for the test generation can be further specified via the design verifier options
- c) To start the subsystem analysis and generate test cases, right-click the Input_scaling subsystem, and select Design Verifier > Generate Tests for Subsystem.
- d) The Simulink Design Verifier software analyses the subsystem. When the analysis is complete, view the analysis results for the subsystem by clicking one of the following options:



- e) By opening the harness model the prepared test cases can be evaluated. The different tests are accessible via the signal builder groups.

It is further possible to reuse the obtained coverage results from the tests prepared using the requirements by selecting them in the options of the Design Verifier > Test Generation > Existing coverage data.

✓ **Question:**

Have you achieved full coverage now? Why you may use the automatic test generation?

Appendix:

Coverage Metrics:

Execution coverage (EC): Execution coverage is the most basic form of coverage. For each item, execution coverage determines whether the item is executed during simulation.

Decision coverage (DC): Decision coverage analyzes elements that represent decision points in a model, such as a Switch block or Stateflow states. For each item, decision coverage determines the percentage of the total number of simulation paths through the item that the simulation traversed.

Condition coverage (CC): Condition coverage analyzes blocks that output the logical combination of their inputs (for example, the Logical Operator block) and Stateflow transitions. A test case achieves full coverage when it causes each input to each instance of a logic block in the model and each condition on a transition to be true at least once during the simulation, and false at least once during the simulation. Condition coverage analysis reports whether the test case fully covered the block for each block in the model.

Modified Condition/Decision Coverage (MCDC): Modified condition/decision coverage analysis by the Simulink Verification and Validation software **extends** the **decision and condition coverage** capabilities. It analyses blocks that output the logical combination of their inputs and Stateflow transitions to determine the extent to which the test case tests the independence of logical block inputs and transition conditions.

- A test case achieves full coverage for a block when a change in one input, independent of any other inputs, causes a change in the block output.
- A test case achieves full coverage for a Stateflow transition when there is at least one time when a change in the condition triggers the transition for each condition

Atomic Subsystem:

An **atomic subsystem** executes as a unit relative to the parent model. Subsystem block execution does not interleave with parent block execution. You can extract atomic subsystems for use as standalone models.

Coverage result SIL:

The following figure shows how the result looks like and where you find the adequate links

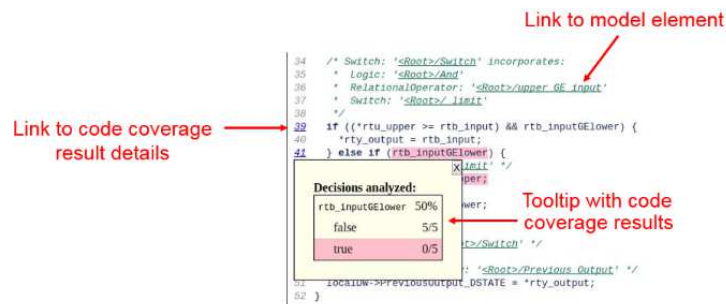
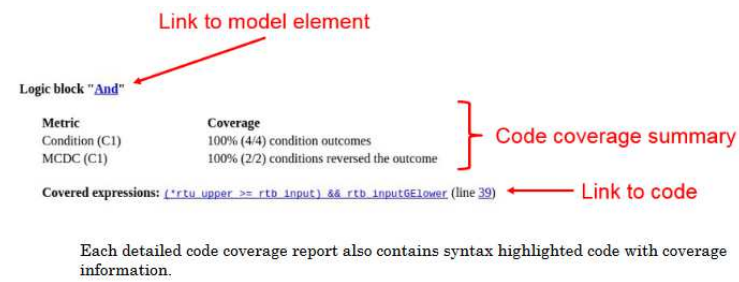


Figure 4: SIL Coverage result