Simulink Logic Simplifier Tool

March 2021

McMaster Centre for Software Certification (McSCert)

Contents

1 Introduction

The Simulink Logic Simplifier tool automatically simplifies Simulink designs to reduce complexity and save developers time during development and refactoring. The tool specifically focuses on simplifying implementations of combinatorial logic in Simulink, such as those making use of logical operators, relational operators, and many others (a list of all supported blocks is given in Section 1.1). After a simplification is performed, the tool also facilitates verification to ensure that the behaviour of the original system is preserved in the results.

1.1 Supported Blocks

All Simulink blocks that are supported during simplification are listed in Table 1, as well as some notes on how simplification is performed in more complex scenarios. These blocks can be found in the built-in block libraries provided by Simulink. Those that are not supported, although they may be selected for simplification, will not be simplified (i.e., will remain in the simplified system). Stateflow designs are not supported at this time.

Table 1: Summary of supported Simulink blocks.

Name	**Block/Mask**	**Notes**
Inport	Block	–
Outport	Block	–
Subsystem	Block	Subsystems can be handled in a variety of ways, depending on the needs of the user. The approach can be altered via tool configuration. The options are: 1. Look under Subsystems to simplify as much as possible. 2. Simplify the contents of Subsystems. 3. Make no attempt to simplify Subsystems.
Constant	Block	–
Logical Operator	Block	AND, OR, NAND, NOR, and NOT are supported. XOR and NXOR are not supported.
Relational Operator	Block	–
If	Block	Translated into logical AND and OR operators by the tool, therefore, does not produce reliable results if the output is not logical.
Switch	Block	Translated into into logical AND and OR operators by the tool, therefore, does not produce reliable results if the output is not logical.
Merge	Block	–
Goto	Block	–
From	Block	–
Ground	Block	–
Compare To Constant	Mask	–
Compare To Zero	Mask	–

2 How to Use the Tool

This section describes what must be done to setup the tool, as well as how to use the tool.

2.1 Prerequisites and Installation

1. Use MATLAB/Simulink 2015b or newer.
2. Install the Auto Layout Tool.
3. Install the MathWorks toolbox Simulink Design Verifier. This allows for verification.
4. To install the Simulink Logic Simplifier, use one of the following approaches:
   (a) **Download the `.zip` from GitHub**
      i. Unzip the contents into your desired location.
      ii. Add the unzipped folder and subfolders to your MATLAB search path.
      iii. Download the Simulink-Utility in the same manner. Add the folder and subfolders to your MATLAB search path also. This is a dependency for the tool to work correctly.
   (b) **Use the Git command line**
      i. Use the following command to download the tool and any necessary submodules.
      ```
      git clone --recursive https://github.com/McSCert/Simulink-Logic-Simplifier
      ```
      ii. Add the folder and subfolders to your MATLAB search path.
   (c) **If you already have the files**
      i. Add the tool folder and subfolders to your MATLAB search path.
5. Run sl_refresh_customizations to refresh the Context Menu.
6. Ensure your model is open and unlocked.

Troubleshooting: If running the command "`which SimplifyLogic`" indicates that the script is not found, then the tool needs to be added to the MATLAB search path. For information on adding files to the MATLAB search path, please see the MathWorks documentation.

2.2 Getting Started

Please ensure the model is open (or loaded, for command line use), and unlocked. The tool can be used via the Simulink Context Menu, which can be viewed by right-clicking in a model with some blocks selected. The following options are available. These options are shown in Figure 1, and explained in greater detail in Section 2.3.

- *Simplify Logic*
- *Simplify Logic and Verify Results*
- *Configuration*

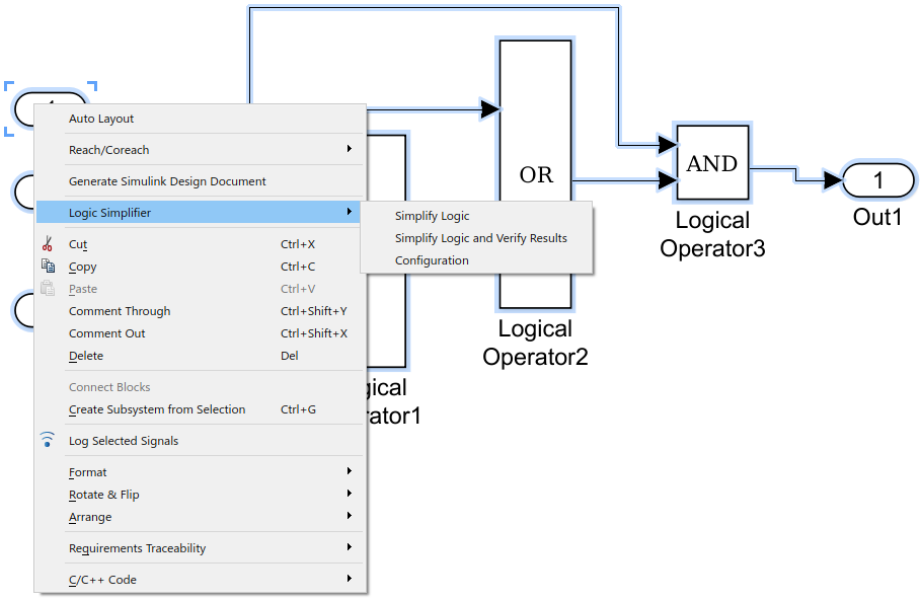

Figure 1: Simulink Context Menu with tool options visible.

2.3 Functionality

This section describes the tool functionality when it is used from the Simulink Context Menu (Figure 1).

Simplify Logic

Right-clicking on a selection of blocks in a system, and then selecting `Simplify Logic` from the Context Menu will:

1. Create a "new logic model" with the filename being the system containing the selected blocks with `_newLogic.slx` appended along with a number to make the filename unique if needed*.

2. Simplify the selection of blocks and generate it in the new logic model along with the remaining blocks from the system containing the selected blocks†. Users should analyze the simplification to make sure that it is correct.

3. The Auto Layout Tool will be run on the simplified blocks. Other blocks will be mostly undisturbed from the original system except from being shifted to avoid overlapping with the new blocks.

Simplify Logic and Verify Results

Right-clicking on a selection of blocks in a system, and then selecting `Simplify Logic and Verify Results` from the Context Menu will:

1. Perform all of the steps performed by the Simplify Logic option.

2. Create a "harnessed model" with the filename being the system containing the selected blocks with `_orig_with_harness.slx` appended along with a number to make the filename unique if needed*.

3. Create a "harnessed new logic model" with the filename being the system containing the selected blocks with `_newLogic_with_harness.slx` appended along with a number to make the filename unique if needed*.

4. Create a "verification model" with the filename being the system containing the selected blocks with `_verify.slx` appended along with a number to make the filename unique if needed *.

5. The "verification model" will feed the same inputs to the harnessed models and compare values of the outputs via the `==` operator and send the results to Design Verifier Proof Objectives. The user should select `Analysis > Design Verifier > Prove Properties > Model` from the menubar. If in the results summary shows that all objectives were processed and proven valid and that 0 were falsified, then the simplification was valid (shown in Figure 6). Otherwise the simplification produced an incorrect result or the verification model was not generated properly (or both).

*When this tool creates a model, it is always saved in the `Logic_Simplifier_Results` folder which is found with the rest of the files for the tool. When this tool names a model, if the name is not a valid model name, instead of using the name of the original system in the filename, `DefaultModel` is used.

†Assumes default configuration parameters.

*When this tool creates a model, it is always saved in the `Logic_Simplifier_Results` folder which is found with the rest of the files for the tool. When this tool names a model, if the name is not a valid model name, instead of using the name of the original system in the filename, `DefaultModel` is used.

Configuration

Right-clicking anywhere in the model and then selecting `Logic Simplifier > Configuration` from the Context Menu, will open the configuration file for the tool (`SimulinkLogicSimplifier\src\config.txt`). The following options are available to the user, and can be modified to fit the user's needs.

- `subsystem_rule` – Customize handling of subsystems.
- `blocks_to_simplify` – Customize which blocks to simplify.
- `generate_mode` – Customize which blocks to generate.
- `extra_support` – Customize handling of different block/mask types (Advanced).

Please see the configuration file for more details regarding parameter usage and accepted values. These parameters can be modified with MATLAB open, and do not require that MATLAB be restarted for the changes to take effect.

2.4 Errors and Warnings

Any errors or warnings during tool use will be visible in the MATLAB Command Window. Typically, errors will be shown when the model is locked or function parameters are incorrect.

A common warning occurs when the model's UnderspecifiedInitializationDetection parameter is not set to Classic. This parameter must be set to Classic in order for the tool to provide more reliable simplification results. The warning follows:

```
Warning: The SimplifyLogic function may result in unexpected
results if the 'UnderspecifiedInitializationDetection' model
parameter is not set to 'Classic', please check the results
carefully.
```

2.5 Known Bugs

There are a few known bugs that users should be aware of:

1. Switch and If blocks may be over-simplified in cases where they pass non-Boolean values. This is caused due to the tool translating Switch and If blocks into equivalent representations using `&`, `|`, $\sim$, `<`, `<=`, `>`, `>=`, `==`, $\sim$`=` (the underlying simplification engine uses this representation).
2. When the verification model is created, two "harnessed models" are created along with it. One of the models is created by first copying the blocks selected for simplification into a model and then unused input ports, unused output ports, and Gotos, Froms, Data Store Reads, and Data Store Writes that send a signal out of the selected blocks are each connected to

an Inports or Outports. The other model does the same with the simplified blocks. This functionality is not fully functional as the added Inports and Outports need to correspond with the same equivalent logic in both models and this will not always be the case. This functionality also currently fails when some inputs are unused after simplification because they were redundant originally. Therefore, in general, the harness should not fail if it is not needed and all inputs are needed. The harness is not needed when there are no unused input ports, unused output ports, and no blocks sending implicit data outside the selection.

3 Example

Use the command `SimplifierDemo` in the Simulink command window to open the example model, shown in Figure 2.

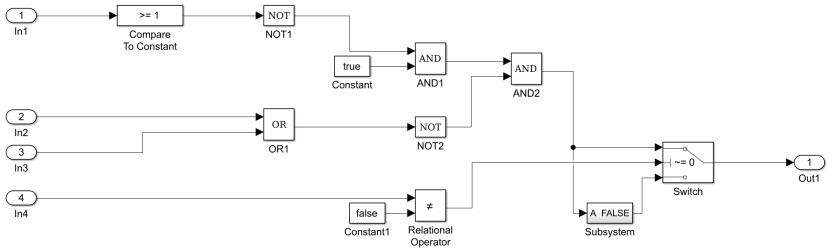

Figure 2: Demo model before simplification.

To simplify this logic, select all the blocks, e.g., with `ctrl+A`. Then, right-click any of the blocks and select `Logic Simplifier > Simplify Logic` from the Context Menu. The resulting model opens automatically and is shown in Figure 3*. We can see many simplifications have occurred:

- NOT1 was removed, and the operator parameter in Compare To Constant was simplified to the less than ($<$) operator, as shown in gen_RelationalOp of the simplified model. This is a valid simplification because if A is not greater than or equal to 1, then one can equivalently say that A is less than 1 ($\neg(A >= 1) \Leftrightarrow (A < 1)$).
- DeMorgan's Law ($\neg(A \vee B) \Leftrightarrow (\neg A \wedge \neg B)$) was applied over OR1 and NOT2. This produced a logical AND operation with NOTs on its inputs.
- A true constant fed into AND1. Using the Identity Law ($A \wedge 1 \Leftrightarrow A$), this can be simplified to remove the true input of the AND.
- In the original model, a false constant and In4 fed into a $\neq$ Relational Operator. This was simplified to simply use In4. Note that this simplification in and of itself is invalid if In4 does not output a Boolean.

- The Subsystem contained logic which reduced to false. This reduction occurred because the subsystem-rule parameter of the configuration file was set to full-simplify.
- Switch passed its first input when In4 was true, and otherwise passed its third input which was always false. This simplified to a logical AND of the first and second inputs.
- There were numerous logical ANDs with two inputs during the simplification that eventually reduced to a single logical AND with four inputs.

In other words the tool did all these simplifications so the user did not have to.

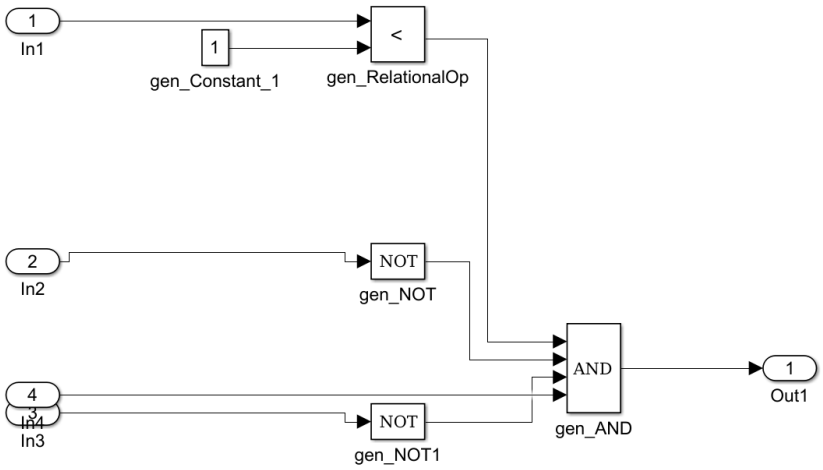

Figure 3: Demo model after simplification.

To perform the same simplification, but also prove that the simplified model is behaviourally equivalent to the original, select the `Simplify Logic and Verify Results` option instead. This will create the simplified model shown in Figure 3, but also create a verification model, shown in Figure 4.

To begin the property proving, select `Analysis > Design Verifier > Prove Properties > Model` from the menubar, as shown in Figure 5.

A "Simulink Design Verifier Results Summary" window will open and display the progress of this operation, as shown in Figure 6. Once the proving operation is complete, the summary window will update to show how many of the properties it was able to prove to be valid or false. The number of properties corresponds to the number of outputs that should be equivalent. Detailed result reports are available in the links.

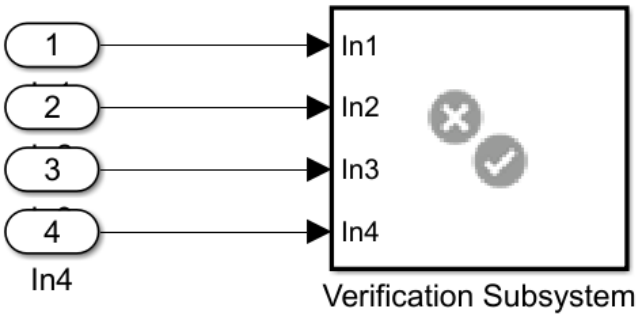

Figure 4: Generated verification model.

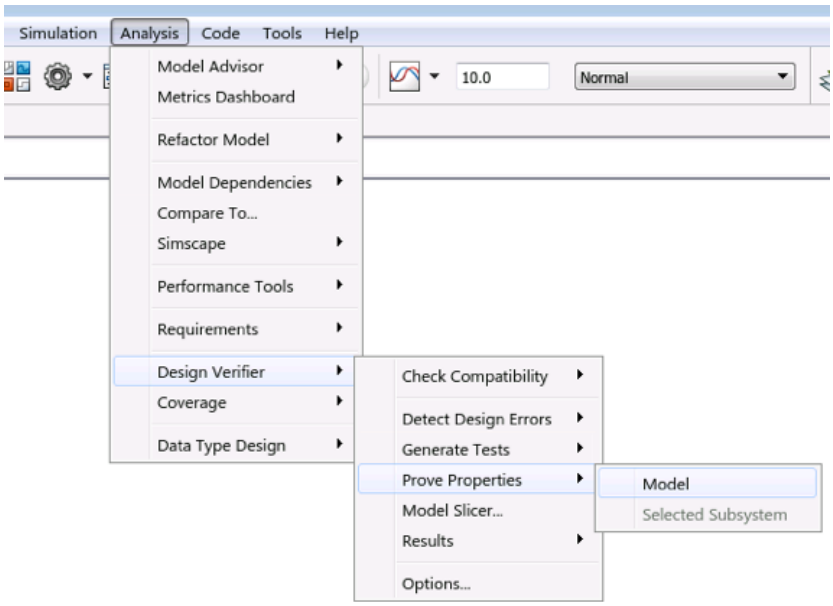

Figure 5: How to begin property proving.

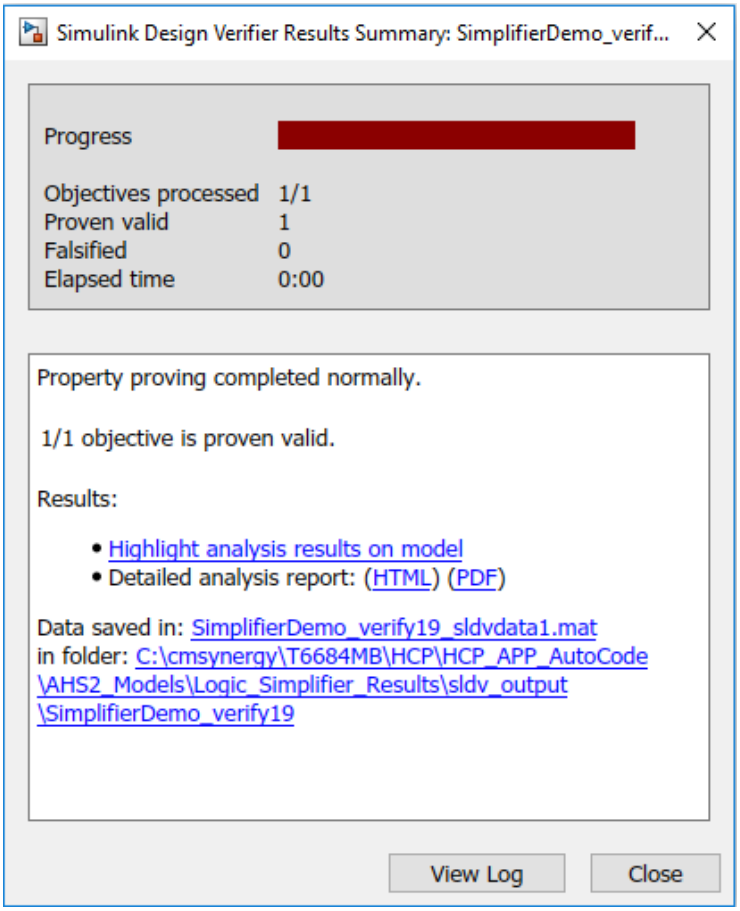

Figure 6: Simulink Design Verifier results summary, with all objectives processed and proven valid.

4 Matlab Commands

The tool can also be used via the MATLAB command line, with the following functions.

Function	`SimplifyLogic`
Syntax	`[newEqu, oldEqu] = SimplifyLogic`(*blocks*, *varargin*)
Description	Construct a model with simplifications of the given blocks (i.e., perform the same steps as the Simplify Logic option from the GUI).
Inputs	*blocks*: Cell array of blocks (indicated by fullname). *varargin*: Boolean indicating whether or not to verify the results. Defaults to false. [Optional]
Outputs	*newEqu*: Cell array of equations found for the blocks as given. *oldEqu*: Cell array of equations found for the blocks after the simplification process.

Function	`makeVerificationModel`
Syntax	`verificationModel = makeVerificationModel`(*address*, *model1*, *model2*, *saveDir*)
Description	Construct a model (`.mdl`) which can be used to verify equivalence between two models using Simulink Design Verifier.
Inputs	*address*: Verification model name. *model1*: First model to verify. *model2*: Second model to verify. *saveDir*: Fullpath of directory to save the constructed model in.
Outputs	*verificationModel*: Fullpath of location and name of the constructed model.