Simulink Logic Simplifier Tool

August 2018

McMaster Centre for Software Certification (McSCert)

1 Introduction

The Simulink Logic Simplifier tool automatically simplifies Simulink systems to reduce the complexity of designs and save developers time during development and refactoring. The tool specifically focuses on simplifying implementations of combinatorial logic, 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 using Simulink Design Verifier that behaviour of the original system is preserved in the results.

1.1 Supported Blocks

All 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).

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 user's needs. The approach can be altered via tool configuration. The different options are: 1. look under SubSystems to simplify as much as possible 2. simplify the contents of SubSystems, 3. or 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

Please ensure the following, before using the tool:

- Use MATLAB/Simulink 2015b or newer.
- The MathWorks toolbox Simulink Design Verifier is installed (allows verification of results).
- The tool is present in your MATLAB path.
- The Auto Layout Tool is present in your MATLAB path. To download, see https://www.mathworks.com/matlabcentral/fileexchange/51228-auto-layout-tool.
- The model is open (or loaded, for command line use).

2.2 Getting Started

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*

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 folder in the current directory (`pwd`) named `Logic_Simplifier_Results`.
2. 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*.

*When this tool creates a model, it is saved in `<pwd>\Logic_Simplifier_Results` where `<pwd>` is the current directory in MATLAB. When this tool names a model, if the name is not a valid model name, then instead of using the name of the original system in the filename, `DefaultModel` is used.

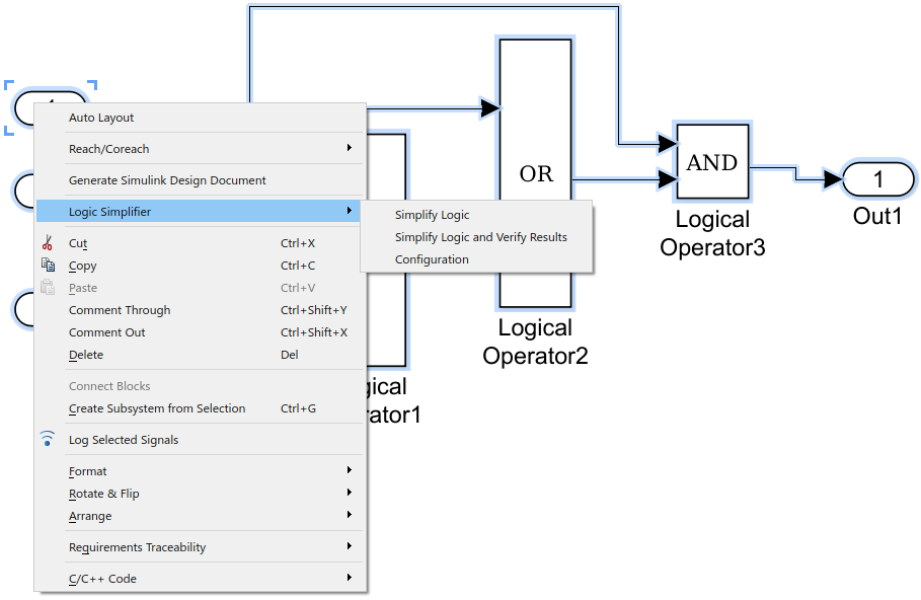

Figure 1: Simulink Context Menu with tool options visible.

3. 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.

4. 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*.

[†]Assumes default configuration parameters.

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 `==` 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).

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.
```

*Whenever this tool names a model, if the name is not a valid model name, then instead of using the name of the original system in the model name, `DefaultModel` is used.

2.5 Known Bugs

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

- Switch and If blocks may be over-simplified in cases where they pass non-Boolean values. This is cause due to the tool translating Switch and If blocks into `&`, `|`, $\sim$, `<`, `<=`, `>`, `>=`, `==`, $\sim$`=`.
- When the verification model is created, two "harnessed models" are created by adding inports and outports to a copy of the original system and a copy of the simplified system and connecting those inports and outports to implicit dataflow blocks and unconnected ports. The harnessing does not work fully, so the verification works best when it is not needed.

3 Example

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

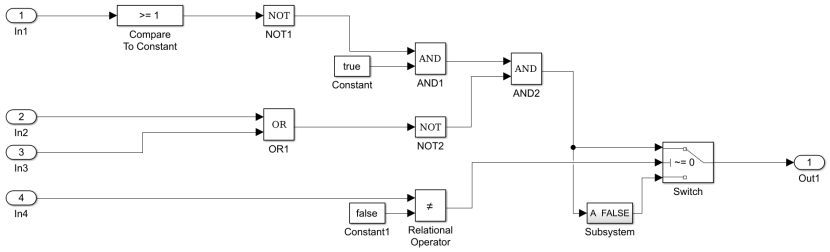

Figure 2: Demo model before simplification.

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.
- DeMorgan's Law was applied over OR1 and NOT2, producing a logical AND operation and NOTs on the inputs.
- A true constant fed into AND1. Using the Identity Law (A&1 = A), this can be simplified as only other 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.

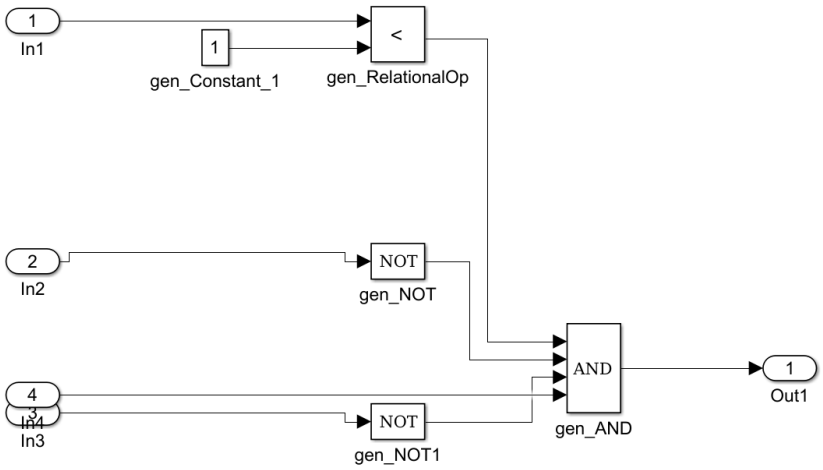

Figure 3: Demo model after simplification.

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 completed, 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.

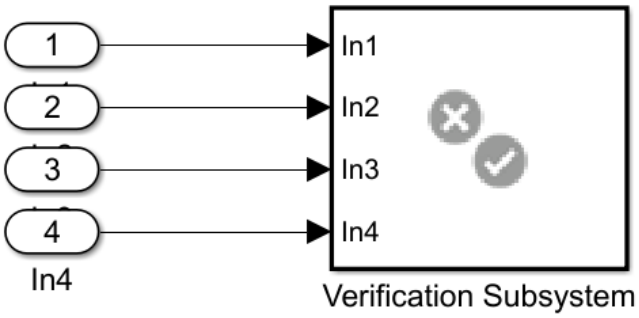

Figure 4: Verification model.

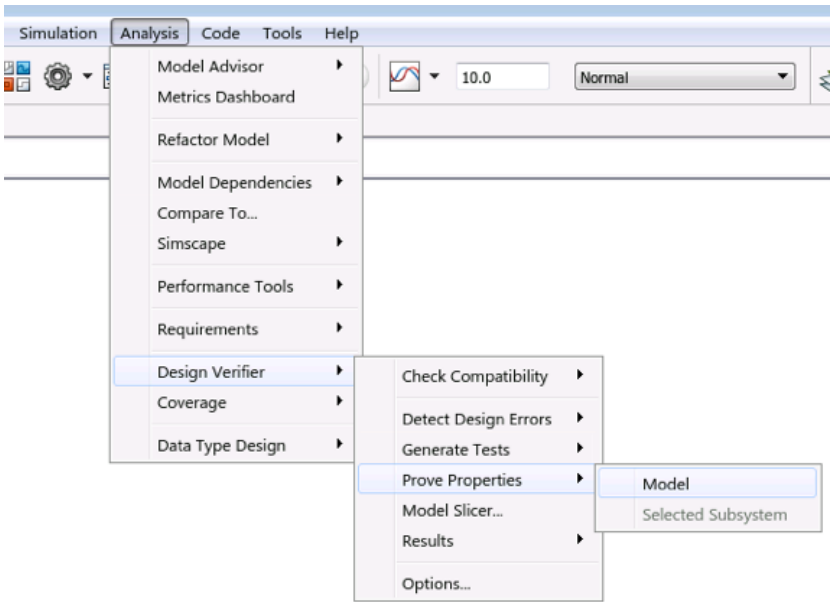

Figure 5: How to begin property proving.

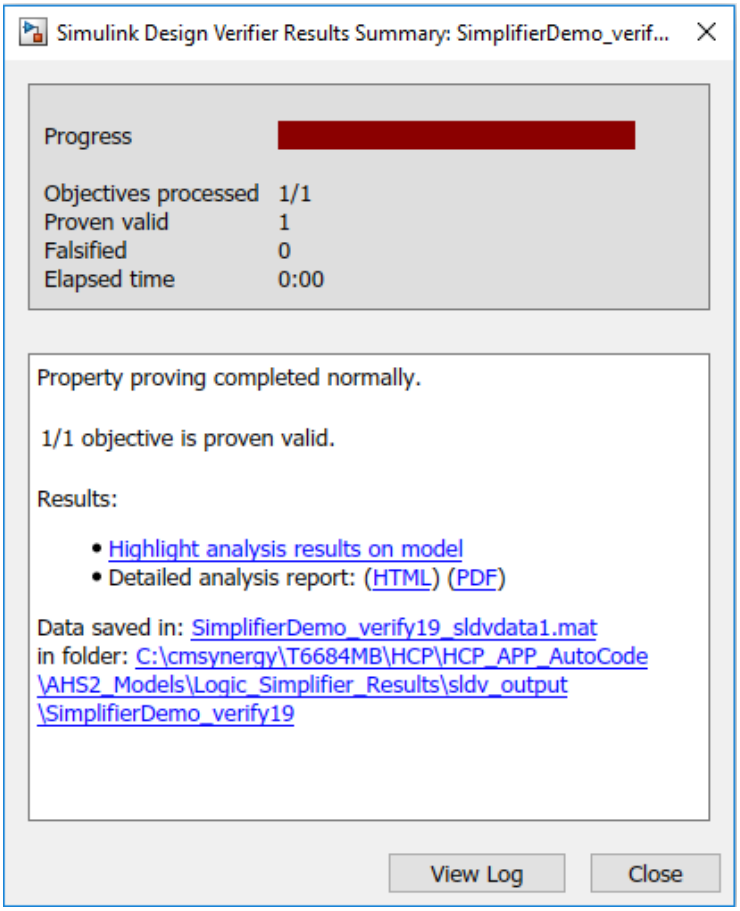

Figure 6: Simulink Design Verifier results summary. All objectives should be processed and proven valid. None should be falsified.

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.