Design Patterns for Integrating Simulink with Stateflow

By Michael Carone

Engineers who use Simulink® and Stateflow® within Model-Based Design often need to integrate state machines and control logic designed in Stateflow with Simulink blocks, subsystems, and components. Common tasks include calling a Lookup Table block from Simulink to perform interpolation on a specific Stateflow variable. Stateflow is often used to enable or disable Simulink subsystems that represent specific tasks, such as startup and shutdown, or individual controller types. Another common procedure involving both Simulink and Stateflow is controlling the behavior of system components, such as the guidance and navigation system of an airplane or a set of sensors located in an automobile.

Products Used

- MATLAB®
- Simulink®
- Stateflow®

To complete tasks and procedures like these requires a seamless interface between Simulink and Stateflow. Starting with R2008b, Stateflow users can create and embed Simulink functions directly inside their Stateflow charts. This article reviews three design patterns for using Simulink functions inside Stateflow: modeling algorithms, scheduling tasks and controllers, and controlling components.

Calling Simulink Functions in Stateflow

Adding Simulink functions to Stateflow is a straightforward process: you simply drag the function into the Stateflow workspace and then double-click the function to open a new Simulink editor window. In the example shown in Figure 1, two Simulink functions, `init` and `steady`, are embedded inside the Stateflow chart. The same syntax used to call graphical functions, truth-table functions, and Embedded MATLAB™ functions is used to call Simulink functions. Simulink windows opened in Stateflow include the same functionality as standard Simulink windows.

Design Patterns

The three design patterns described below are arranged in order of complexity, from simple algorithm development to component-based design.

Modeling Algorithms

Using a Simulink block or algorithm modeled within a Stateflow chart is an efficient way to include reliable Simulink algorithms

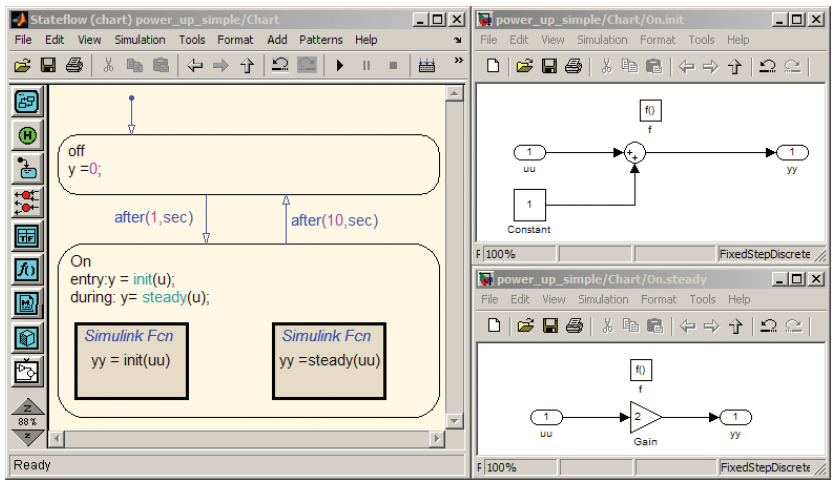

Figure 1. Left: Stateflow chart incorporating Simulink functions created using the Simulink function button (the white icon in the left navigation bar). Right: Function details.

within your control logic. In Figure 2, a Simulink Lookup Table block is incorporated into a Stateflow chart using Simulink functions. The block is used to perform a linear interpolation of one of the input variables to the Stateflow chart. The state machine compares the output of this function to another Stateflow variable to determine whether the state machine should be in the S1 or S2 state.

There are several other possibilities for including Simulink algorithms within Stateflow. For example, custom library blocks can be added to Simulink functions. You can then add your own reusable Simulink algorithms to the logic and call these algorithms throughout the state chart. The blocks that can be added to Simulink functions are not limited to those in the Simulink library; for instance, you can also add the Fast Fourier Transform block from Signal Processing Blockset™.

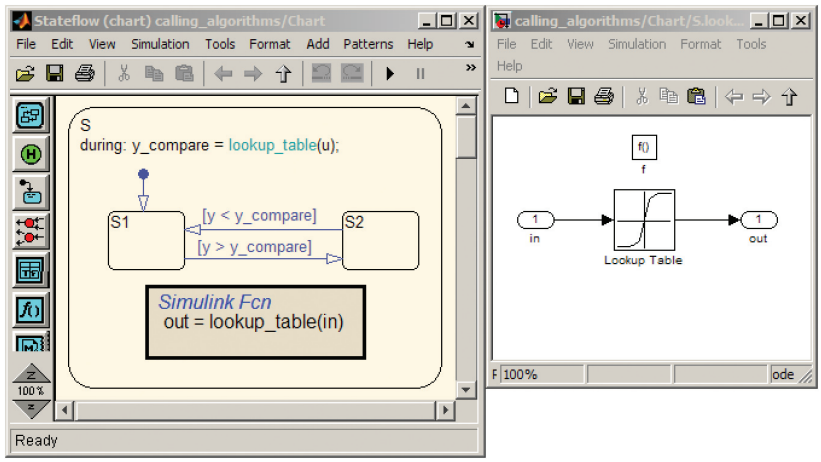

Figure 2. Design pattern for calling Simulink algorithms from Stateflow.

Scheduling Tasks and Controllers

Stateflow is commonly used to model schedulers that determine when certain actions take place. For example, Stateflow can be used to schedule exactly when a system should start up, perform a certain operation, and then shut down. The tasks that are performed when the system is in each of those states are often modeled in Simulink.

With Simulink functions, you can directly associate these tasks with the corresponding state (Figure 3). In this design pattern, three phases of a process are represented by three Stateflow states. When the system is in phase 1, the Stateflow chart executes the task that is modeled in the Simulink subsystem (task 1). After 60 seconds, the system enters phase 2, and task 2 is executed. After another 60 seconds, task 3 is executed.

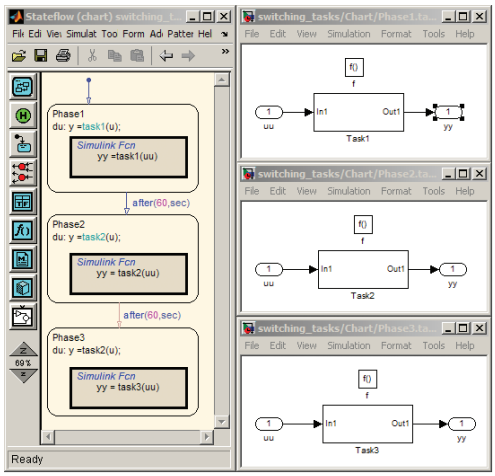

Figure 3. Design pattern for scheduling tasks modeled in Simulink from Stateflow.

A key advantage of this approach is the readability of the Stateflow chart—for example, it is immediately apparent that the task to be executed when in phase 1 is the task that is modeled within the "task1" Simulink function.

Another application of this design pattern is to activate different types of controllers based on the state of the system. For example, in one state, the controller can behave in an open-loop fashion. When the difference between the desired response and the actual response reaches a certain threshold value, the state machine can transition to the closed-loop state, at which point the closed-loop controller, modeled in Simulink, is activated.

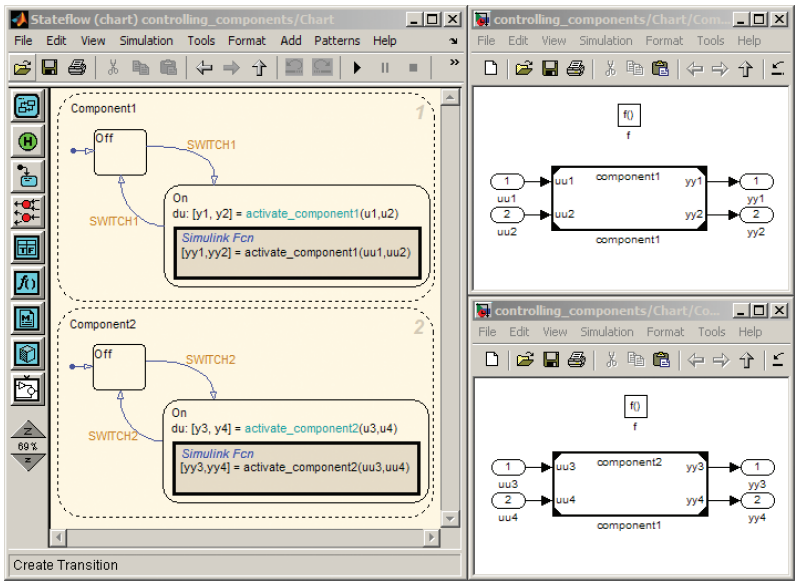

Figure 4. Design pattern for controlling components within Stateflow.

Controlling Components

Another common application for Stateflow is to control the behavior of components modeled in Simulink. Figure 4 shows two independent components represented by two parallel states in Stateflow. These components can be either off or on. When one of these components is switched on, the Simulink component is activated from the Stateflow chart. The Simulink functions shown in Figure 4 look very similar to those shown in Figure 3, with one major difference: The blocks shown in Figure 4 are Model blocks, not Subsystem blocks. Model blocks link to models that can be developed and tested independently. These blocks are important for componentized design and large-scale modeling.

Extending this Approach

This article has described several ways that you can combine the dynamic systems modeling capabilities in Simulink with Stateflow control logic design tools. The design patterns discussed here are only a subset of the tasks that you can perform by integrating Simulink and Stateflow. For example, you can create componentized state charts by adding separate Stateflow charts inside the Model blocks shown in Figure 4. If your model is logic-centric, you can configure it so that the state chart is the baseline of the model and you add Simulink functions to call out to Simulink. You can also create variable multirate Simulink subsystems by executing Simulink functions within Stateflow, which acts as the scheduler. ■

Resources

VISIT
www.mathworks.com

TECHNICAL SUPPORT
www.mathworks.com/support

ONLINE USER COMMUNITY
www.mathworks.com/matlabcentral

DEMOS
www.mathworks.com/demos

TRAINING SERVICES
www.mathworks.com/training

THIRD-PARTY PRODUCTS AND SERVICES
www.mathworks.com/connections

Worldwide CONTACTS
www.mathworks.com/contact

E-MAIL
info@mathworks.com

91642v00 01/09

For More Information

- Webinar: Stateflow Design Patterns
  www.mathworks.com/wbnr30311

Download This

- Stateflow Design Patterns
  www.mathworks.com/stateflow-patterns