FSMCreator User's Guide

by Ayla Solomon

Table of Contents

Introduction

Finite State Machines

The FSMCreator is used to build finite state machines, a logic construct that models the behavior of an object or system; in this case, the nanopore system. Each state, of which there can be few or many, represents the inputs to and outputs of the system under certain conditions. The inputs to the FSM are voltage levels and triggers, based on the current, and state time. The outputs are constant output voltage and ramp output voltage. When the user wants the FPGA to switch state based on the inputs, that is a state transition.

Starting the Program

To use this program, Java Runtime Environment 6/1.6 or later must be installed. (To check, open the Command Prompt, found under Accessories, and type `java –version` and then enter. If the version displayed is 1.6.0_01, you have the correct version.) If this version is not installed, then an update is available for free at www.java.com.

To open the program, double-click the FSMCreator.jar icon. This will automatically start the program.

The first thing displayed is a small dialog box that asks you to enter a name for the first state. This can be any value, alphanumeric or symbolic. It is not length-constrained (to a point), though for ease of use a shorter title is better. You don't have to enter a title at the time; continuing without entering a value will set the first state title to the default title State 0. States are indexed starting at zero, regardless of what they are named.

Interface Details

State-Independent Values

The state-independent values are located on the right side of the interface, under Sampling Parameters and Trigger Controls. This pane is separate from the state tabs and will not change or multiply when the states are modified in any way. Except the window step size field, which is integer-only, these are decimal-only fields. Any non-number value entered into them (excluding decimal points) will be cleared once the field is deselected. In the window step size field, any decimal will be truncated to form an integer. The filtering alpha must be between 0 and 1 to save, though the value is not enforced during editing.

State Panes

The tabbed pane on the left side of the interface is the state area. Each tab represents a different state and everything bounded by the tab is state-specific, aside from the Write FSM button. There are three main areas in the state pane: The state values, the transitions subpane, and the state controls.

State Values

The state values are the fields at the top of the state pane—ramp voltage output, constant voltage output, and state time. Like the other components in the state pane, these are state-specific. These fields, like the state-independent values, only hold only decimals.

The note pane to the far right of the state pane can contain any keyboard character and is designed to allow the user to annotate the state. Clearing it removes all notes and affects nothing else.

State Controls

The buttons at the bottom of the state pane are the state controls. Create State will add another state to the tabbed pane and will pop up the same state title dialog box that was shown when the program was opened. Once again, it is not required that anything be entered into the field; a default title will be used in nothing is input.

Deleting a state affects more than just the current state. It updates the state titles of states appearing after the deleted state—the number in parentheses is the state number—to reflect the new updated state number. All transitions to that state are removed and, if the deleted state was the last state, the other states can no longer transition to it. Because this can constitute a major setback if a state is accidentally deleted, it opens a dialog box to confirm that the user wants to delete a state.

Clearing a state deletes all transitions, clears the state values, clears the notes box, and resets the selector boxes for the state transitions. This only affects the currently selected state.

Writing the FSM is not state-specific and does not affect the FSM. It will open a save dialog and will save the FSM as a .txt file. The file is not meant to be read, but an example annotated file is included for curious users in the Supplements section.

Transition Controls

Transition controls include all of the multi-option selectable boxes, the list to their right, and the transition buttons at the bottom right above the state controls. The first five multi-option boxes, or comboboxes, contain the values True, False, and Don’t Care (default). The Go To State box contains the state numbers of all states except the current state. When only one state is present (such as when the program is first opened), this box is blank because there are no other states. For this reason, no transition can be added at that time—an error box will pop up if an attempt is made to do so. This box will also pop up if a state is deleted and then an attempt is made to transition to it.

The box at the right, initially blank, is the transitions list. It contains all transitions currently added to the state. When a transition is added, is appears in the form MMMMM N, where the M’s are values from the first five comboboxes (1 is true, 0 is false, X is don’t care) and N is the state to which it transitions. Once a state is in the list, double-clicking on it will set all comboboxes to reflect that transition.

The clear transition buttons do as they say—clearing the last transition deletes the transition on the bottom of the list and clearing the selected transition clears the transition currently highlighted. If no transition is highlighted and that button is pushed, nothing happens.

The Menu Bar

The File menu contains the predictable File items—New, Open, Save, Save As, and Close, as well as Write FSM and Save Printable Version. New and Open close the current FSM without saving and warn the user of that. Closing the FSM also pops up a dialog to make sure the user wants to close.

Save and Save As is different from Write FSM (which does the same thing as the button). They save the FSM as a .fsm file, as opposed to .txt. These files, like the .fsm files, are meant to be neither read nor edited by the user. The purpose of the save option is to allow a user to come back to unfinished FSMs or to edit already complete ones to perform a different function. The difference between Save and Save As is standard—if the file has already been created and Save is selected, it saves to the same file without popping up a save dialog. Save As will pop up the save dialog box every time and allow the user to save the same FSM under a different name.

Opening an FSM is executed as it is in any other program; however, the program will only open .fsm files. A .txt file will not show up on the list of openable files and therefore it is impossible to open a file created by the Write FSM function.

Saving a printable version creates a .txt file representing the FSM in a way that is easy to read and understand. This file is recommended for documenting an FSM as it is the only really readable format produced. An annotated example is included at the end in the Supplements section.

The State menu contains state-specific controls, all of which are also present as buttons, and so do the exact same things, except for the Change State Title control. Selecting this will pop up a title-change box for the currently selected state. If nothing is entered or the Cancel button selected, the state title will be set to default.

Nitpicks and Problems

Saving and Writing FSMs

There are some quirky things that the program does when it is asked to save. The largest concern, or the most irritating if it isn’t known, is that if a field is edited (such as any state or machine control boxes or the note pane) and remains selected when the FSM is saved or written, that edit will not stick in the saved version. To avoid this, it is essential that the field is deselected after editing, either by selected a different field or by changing states. Once it is deselected, the program logs the change and it will be saved.

Keyboard Shortcuts

There are keyboard shortcuts for all menubar items. If a user wishes to use one while editing an FSM, either nothing or a state tab may be selected for it to work. The fist time a user clicks a field or button, something becomes selected and after that point there is always something selected, whether the user can see it or not.

Creating New States

When a new state is created and the Go To State lists in all the states are updated, sometimes the lists will go blank. This does not mean that no states can be transitioned to, it is just an inexplicable bug. Dropping down the Go To State list and clicking the blank space will revive the list and allow the user to select any state.

Supplements

Annotated Save Printable Version file: Simple 3-state machine

```
Epsilon(v): 4
Filtering Alpha(0<=a<1): 0.9
Window Step Size (samples): 2
Trigger Value (1): 25
Trigger Value (2): 0
Level Value (3): 0
Level Value (4): 0.0
```

These are the machine values.

```
################
##Use a high voltage to get a hairpin in the pore. When one is detected,
transition to Stall.
Capture (0)
State Time(ms): 0
Constant Output Voltage(mV): 180
Voltage Ramp(V/s): 0

Transitions:
When: Voltage Trigger 1 is true, go to state 1.
```

This is the 0th state. The hash marks signal an annotation. If no carriage return is used in the notes pane, it will wrap and may be harder to decipher.

```
################
##Wait 10 ms to make sure the hairpin is secure. If it is not and leaves the
pore, go back to Capture.
Stall (1)
State Time(ms): 10.0
Constant Output Voltage(mV): 30
Voltage Ramp(V/s): 0.0

Transitions:
When: State Time Completion is true, go to state 2.
When: Voltage Trigger 1 is false, go to state 0.
```

For the transitions, no ‘don’t care’ value is included, only the true or false values.

```
################
##Once the hairpin is secure in the pore, ramp the voltage until is shears
and translocates through.
Ramp (2)
State Time(ms): 0
Constant Output Voltage(mV): 30
Voltage Ramp(V/s): 1

Transitions:
When: Voltage Trigger 1 is false, go to state 0.
```

Annotated Write FSM file, for the curious

(Machine values. Same as seen in the GUI.)

```
4
0.9
2
25
0
0
0.0
```

(Number of states.)

```
3
```

(Transitions—Two to a number, which is a 16-bit decimal. There are 32 different possible transitions (2^number of inputs) per state, though only a few are used.)

```
0
257
0
257
0
257
0
257
0
257
0
257
0
257
0
257
```

(Begin transitions for state 1.)

```
0
513
0
513
0
513
0
513
0
513
0
513
0
513
0
513
```

(Begin transitions for state 2.)

```
0
514
0
514
0
514
0
```

```
514
0
514
0
514
0
514
0
514
```

(State values. These have been optimized to run on the FPGA.)

(Delay values for each state.)

```
0
1
0
```

(Delay multiplier for each state. The delay value and multiplier together form the FPGA-optimized delay.)

```
0
1887
0
```

(Constant output voltage for each state.)

```
5898
983
983
```

(Ramp rate values for each state—the y-value of the ramp.)

```
0
0
189
```

(Step increment values for each state—the x-value of the ramp.)

```
0
0
1
```