An Introduction to Active-HDL™ FSM

Introduction

Active-HDL™ FSM, a finite state machine graphical entry tool, is the latest addition to the *Warp™* design development environment. Active-HDL FSM generates both VHDL and Verilog IEEE compliant code from a graphical state diagram behavior model. State machines can be implemented in Cypress CPLDs using the *Warp* software. The state editor has an intuitive graphical user interface, fully featured for developing all types of state machines. The Active-HDL FSM editor makes state machine design easier by enabling users to view the behavior using the intuitive bubble-diagram representation. The purpose of this application note is to provide an introduction to the Active-HDL FSM editor, while highlighting key features of the software and illustrating the steps required to create a state machine.

Installing the FSM Editor

Active-HDL FSM is installed as part of the *Warp* installation. The FSM editor can be invoked from the Galaxy environment by selecting *Tools → Active State Editor*. To uninstall the Active-HDL Simulator and the Active-HDL FSM editor, click on the Add/Remove programs in the Windows control panel and select Active-HDL Sim.

State Machine Types

There are generally three types of state machines: Mealy, Moore, and combined Mealy/Moore. A Mealy State machine has outputs that are a function of the current state and primary inputs. A Moore State machine has outputs that are a function of the current state only, and produces outputs directly from the state register using combinatorial logic. A combined Mealy/Moore state machine has both types of outputs. All types of state machines can be created in the Active-HDL FSM editor, with combinatorial and registered outputs, asynchronous and synchronous resets, and both positive and negative clock edge triggering. The following section briefly describes setting up and creating a state machine.

Creating and Developing a New Design with the FSM Tool

Using the Design Wizard

The *Design Wizard* is a user friendly, step by step process for initializing a new design environment. After launching the Active-State Editor from Galaxy's menu, start the design wizard by selecting *New Design*. Navigate through the Design Wizard by clicking the *Next* button (see *Figure 1*). Define the HDL language, filename, port map, and number of state machine for the environment. After clicking *Finish*, the Wizard sets up the state machine environment. All features can be altered at any time after completing the Design Wizard.

HDL Languages (VHDL/Verilog)

The FSM editor can generate two types of HDL code, VHDL and Verilog. The HDL languages are IEEE compliant (1164 VHDL, 1364 Verilog) and readable by the *Warp* synthesis tool. Choose a HDL language by selecting *Synthesis → Configuration*. (See *Figure 2*.) The default tool is *Warp*. Specify the output code language as VHDL or Verilog.

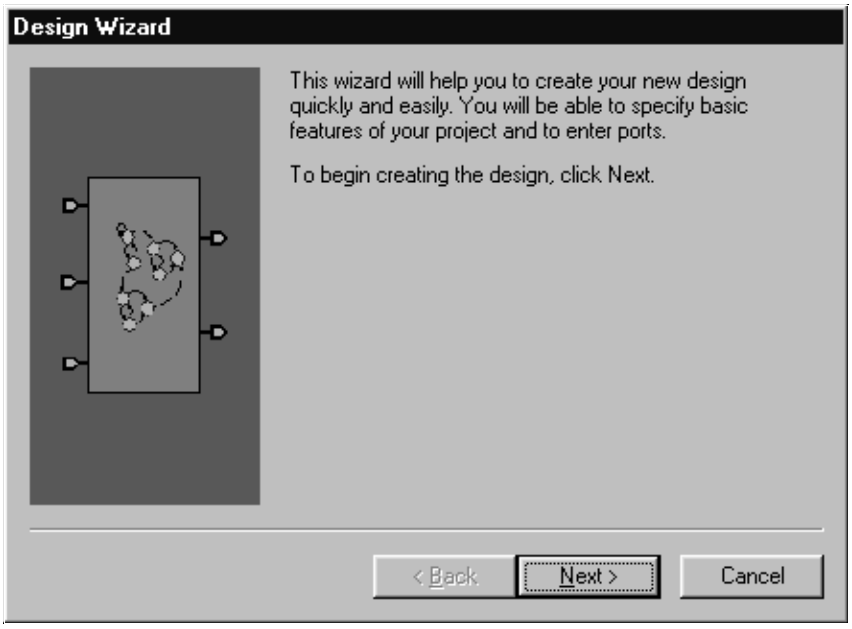

Figure 1. Design Wizard

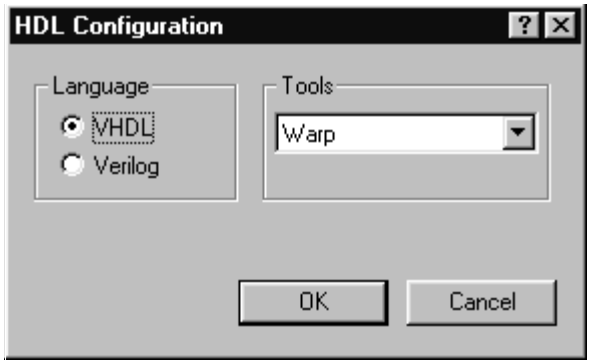

Figure 2. HDL Configuration

Library Headers

Header declarations are necessary for including any standard or user libraries needed for a design. To view and edit header information select *Synthesis → Select Libraries.* If VHDL is the output language, the Library Headers window contains some of the default library include statements (see *Figure 3*). IEEE Verilog doesn't require any library header information, therefore in Verilog mode the Library Headers window is temporarily disabled. The default header is required for *Warp* synthesis and should not be removed, additional headers may also be added.

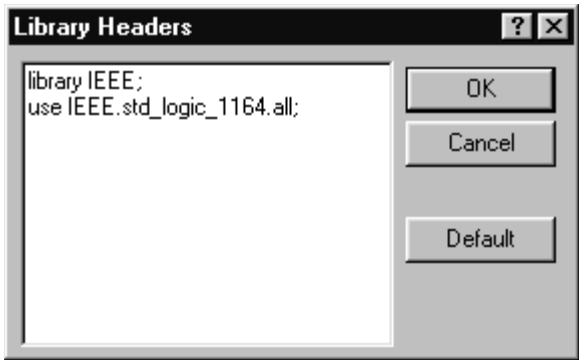

Figure 3. Library Headers

Editing the State Machine Properties

The state machines general properties, resets, and default conditions define the State Machine Properties. To edit the machine properties, *Right-click → Properties* on the state machine canvas. The Machine Properties window opens. The three tab options (General, Reset, and Defaults) are described in the following three sections.

General Machine Properties

With the General tab selected, several of the state machine's properties can be defined including the machine name, input clock signal with edge sensitivity (see Adding a Clock Input), and the encoding method (see *Figure 4*). In HDL code, all the states in one machine are represented by an encoding scheme: Binary, Gray, Johnson, One-Hot, or User defined.

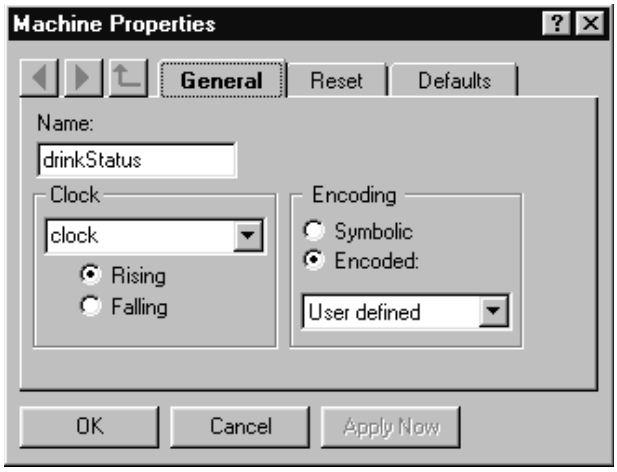

Figure 4. General Machine Properties

Machine Resets

Click the Reset tab to edit the reset signals and their states (only available after defining new states, see Adding States). To set up the reset signal, choose an input port and reset state using the drop down boxes (see *Figure 5*). Use the type (asynchronous/synchronous) and active level options (active HIGH/LOW) to define the reset characteristics.

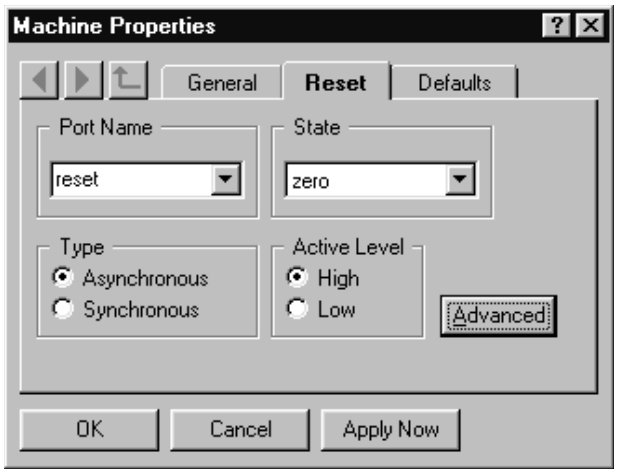

Figure 5. Machine Resets

Default States

The state machine defaults and illegal states are defined on the Defaults tab (see *Figure 6*). Unsatisfied conditions (Don't care, Default state, Hold) and illegal states (Don't care, Trap state) can be selected.

Figure 6. Default States

Creating a State Machine

The State Machine Environment

The FSM environment is separated into two portions: the port map and the state machine canvas. The top portion is the port map section; the lower enclosed section is the state machine canvas (see *Figure 7*). In the port map section, all the input, output, and bidirectional ports are declared. The declared ports in the port map section define the entity and module sections of the VHDL and Verilog code respectively. When creating a new state machine, the first step is to declare the port map of the machine. The state diagram in bubble and arrow representation is created on the state machine canvas. The state machine's signal name is defined in the upper left corner of the state machine canvas.

Adding a Clock

To add a clock to the state machine select *FSM → Input Port*, and place the transparent input port attached to the mouse pointer in the port map section of the editor environment. Click once on the port label to select it. Click again to edit the text field. *Right-click → Properties* to bring up the Properties window on the input port. Check the clock button and click *Apply now.* The Clock input icon is altered, displaying a clock waveform inside the input port symbol.

Adding Inputs and Outputs

To add inputs and outputs to the state machine follow the same initial steps as adding a clock input. Select one of the port types under the FSM menu, and place the transparent symbol attached to the mouse pointer in the port map section of the editor environment. Click once on the port label to select it, click again to edit the text field. Port buses can be declared by opening the Port Properties window, *Right-click → Properties.* Manipulate the range arrow buttons to define the appropriate bus size. Output and bidirectional ports need to be defined as registered or combinatorial. In the same Port Properties window, use the left and right arrow tabs located at the top left of the window to select a port then set the appropriate bus size and select registered or combinatorial.

Creating States

To create a new state, select *FSM → State*. Place the transparent symbol attached to the mouse pointer on the state machine editor canvas. *Right-click → Properties* on the state bubble to bring up the State Properties window. Enter a new name in the name field, and if necessary, a coded enumera-

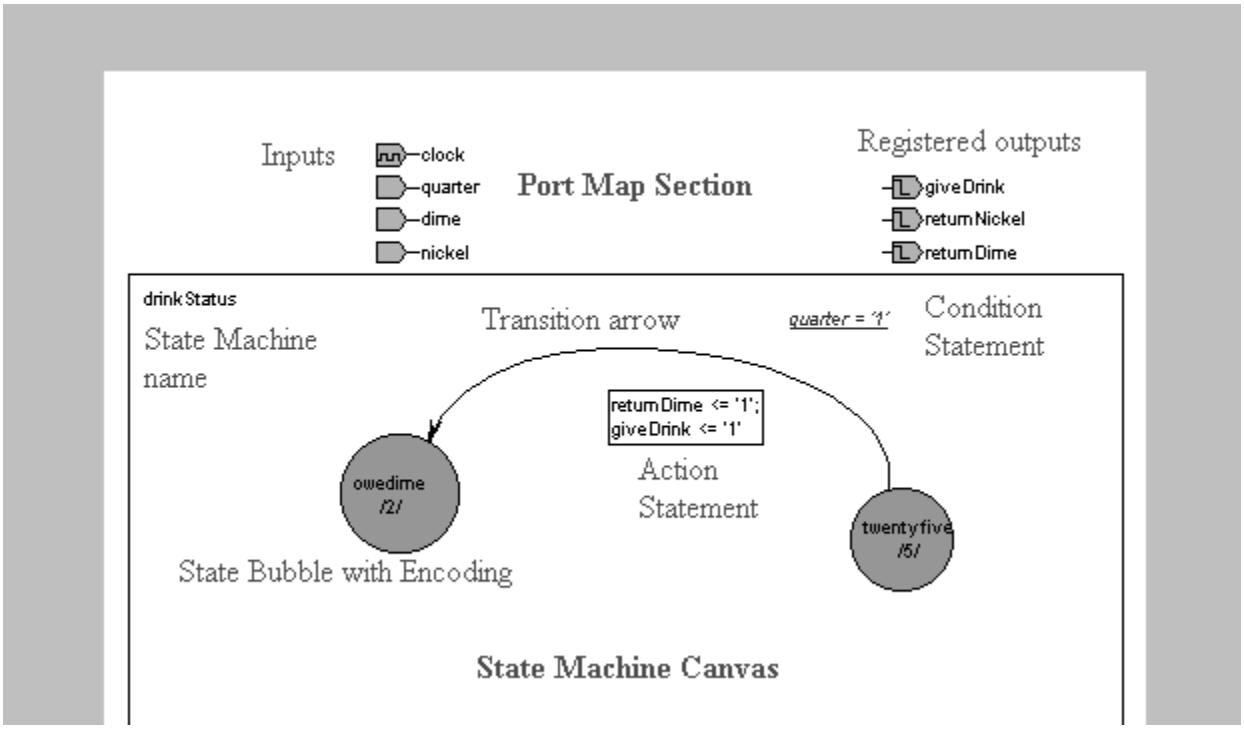

Figure 7. Labeled Example of the FSM Environment

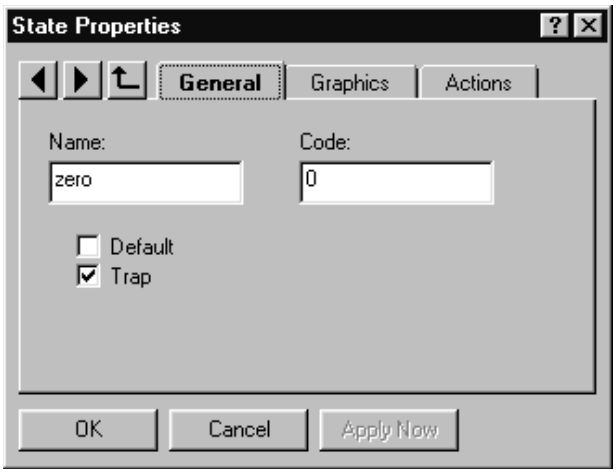

Figure 8. General State Properties

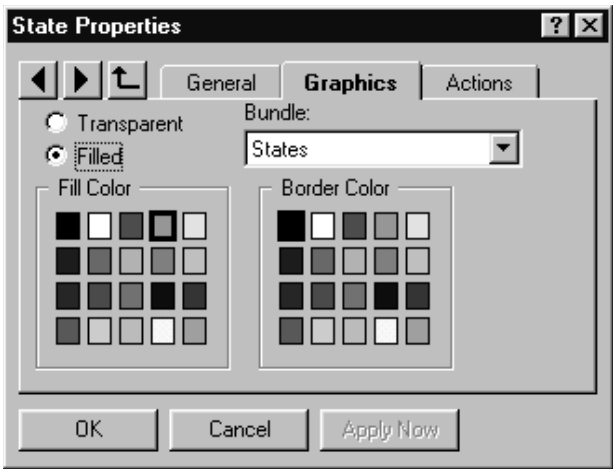

Figure 9. Graphics Tab

tion (see *Figure 8*). Use the arrow tabs to toggle between all the states and their properties.

In the State Properties window, two state attributes can be altered, the visual graphics and the actions defining the state. On the Graphics tab (see *Figure 9*), select a user-defined or default bundle and customize the current state by using the filled, transparent, and color selections. A group of different objects all sharing the same visual characteristics can be grouped together into bundles (see Editing the Appearance of the State Machine). The Actions tab allows the user to define entry, state and exit actions for the selected state bubble (see Adding Action Events).

Default and Trap States

The Default State is the state that the machine transitions to if none of the transition conditions are valid. If the machine transitions or resets to an undefined state the state machine will transition to the Trap state on the next clock cycle. One Default and Trap State can be defined for a single state machine. To define the Default and Trap States *Right-click → Properties* on the state machine canvas, and select the Default tab. Both the Default and Trap States can be selected from the drop down list boxes.

State Transitions

Depending on the type of state machine being designed, the outputs are based on the current state and possibly a set of primary inputs. A state machine transitions between states based on conditions of input signals and the current state. A transition is known as the movement between states in a state diagram. If a transition condition arises then the state will shift to a new state and perform the transition action statement. To add a transition, select *FSM → Transition*. Click the state bubble where the transition will originate from, then click on the final state to complete the transition declaration. To make the transition easily visible in the state diagram, adjust the arrow handles on the selected transition arrow. To define transition conditions and actions *Right-click → Properties* on the transition arrow, and click the HDL tab (see *Figure 11*).

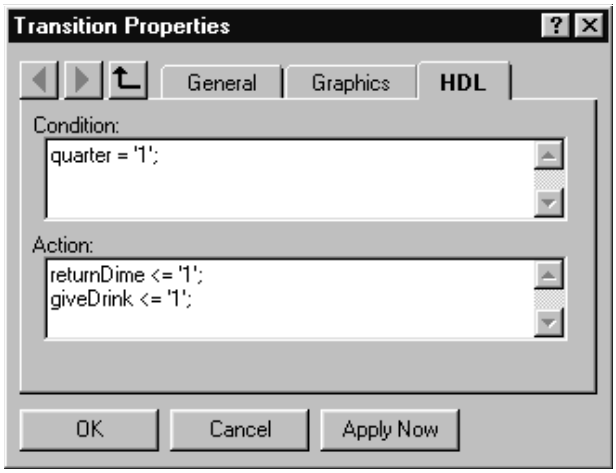

Figure 11. State Transitions

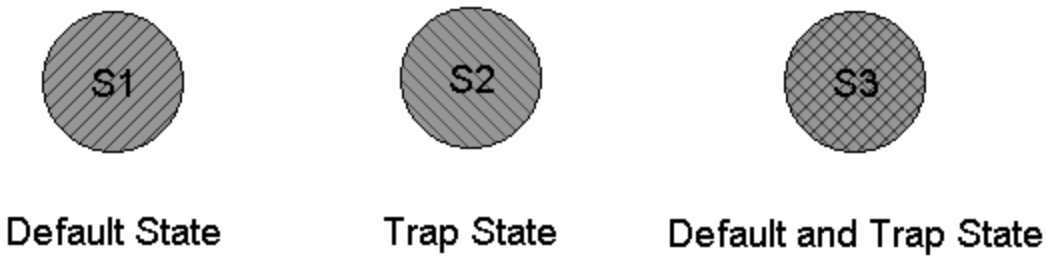

Figure 10. Active-HDL FSM Representation of Default and Trap States

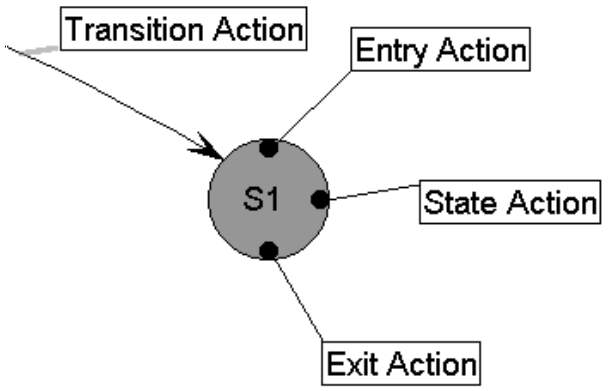

Figure 12. Action Events

Adding Action Events

An action statement defines the output port connections based on the current state or a set of transition conditions. There are four ways of declaring an action statement: transition condition (see State Transitions), Entry action, State action, and Exit actions (see *Figure 12*). An Entry action occurs whenever any state transitions into the designated state. State actions occur whenever the defined state is the current state. An Exit Action occurs whenever the state transitions from the designated state to any other state. Actions can be defined by *Right-clicking → Properties* on a selected state, and choosing the Actions tab. A new Action event can be created by selecting the action event from the toolbar or from the FSM menu.

Syntax

One state diagram can be created in the FSM editor which can target both VHDL and Verilog output code. There are a few minor changes that must be made to ensure proper compilation with the Galaxy compiler. The syntax of Condition and Action statements must be altered depending on the target output HDL language. In both VHDL and Verilog, Action statements must have semicolons following each statement. The Active-HDL FSM conditions statements are inserted into if-then clauses in VHDL and Verilog, and don't require semicolons. Multiple conditions are defined, as they would be in VHDL or Verilog, using logical operators (and/or syntax). *Table 1* shows examples of conditions and action statements for both VHDL and Verilog.

GUI Features

Editing the Appearance of the State Machine

Every object in the state machine editor has predefined appearance properties, which can be edited in two ways. A bundle can be created which defines the appearance properties for a specified group of objects. Otherwise, each object can be individually customized by *Right-clicking → Properties* on the object and select the Graphics tab. To edit the default editor bundles or to create new bundles open the Settings window by selecting *Edit → Settings*. This window classifies all the object bundles into three different sections: Text, Line, and Fill. Click on the bundle name and edit the characteristics to change all previous and future objects in that bundle.

Snap to Grid

A grid can be placed in the background of the FSM state machine canvas. All objects on the canvas can be snapped to a grid of any size by selecting *View → Grid* to open the Grid Window. Define the grid characteristics, and select the Snap to Grid checkbox.

Manipulating the Objects on the Canvas

All the objects on the canvas can be moved and manipulated. The states can be resized, and the transitions can be reshaped and repositioned on the state diagram. Moving the position of the state bubbles doesn't destroy the transition link connections between states. If a connection between a transition and a state is broken, the end of the transition will be crossed out.

Add Graphical Text to the State Canvas

Graphics and text can be added to the state diagram to add clarity. Use the Draw toolbar or menu to add text, bezier curves, circles, or rectangles to the canvas. The graphical objects all have handle boxes for altering the shape and size of the graphic. These graphics do not affect the source code generated

HDL Code Generation

The Active-HDL FSM editor can output both VHDL and Verilog code depending on the configuration setting. Select *Synthesis → Configuration* and then the desired HDL format. To generate the HDL code, select *Synthesis → HDL Code Generation*. If no errors are detected, the compiled code can be viewed immediately in Windows Notepad. The Active-HDL FSM editor does not contain a syntax checker. Any syntax errors in the condition or action statements will not generate errors in the FSM editor. If an error is detected in the state machine, a red error box will enclose the problem sections of the state diagram. All errors are displayed in the error detection window at the bottom of the FSM environment.

Summary

This application note provides a brief introduction to the Active-HDL FSM state machine editor. The FSM editor can be used to develop a state machine flow and output correct and efficient HDL code. The Active-HDL FSM editor creates IEEE standard VHDL and Verilog source code, which are both inputs into *Warp*. Further information and details are available in the online help.

Table 1. Example of VHDL and Verilog Condition and Action Statements in the FSM Editor

	Condition Statements		Action Statements	
Language	**VHDL**	**Verilog**	**VHDL**	**Verilog**
Single Statement	dime = '1'	dime	giveDrink <= '1';	giveDrink = 1;
Multiple Statements	dime = '1' and quarter = '0'	dime and !quarter	giveDrink <= '1'; returnDime <= '1';	giveDrink = 1; returnDime = 1;

Warp is a trademark of Cypress Semiconductor Corporation. Active-HDL is a trademark of Aldec Incorporated.