Creating Safe State Machines

Definition & Overview

Finite state machines are widely used in digital circuit designs. Generally, when designing a state machine using a hardware description language (HDL), the synthesis tools will optimize away all states that cannot be reached and generate a highly optimized circuit. Sometimes, however, the optimization is not acceptable. For example, if the circuit powers up in an invalid state, or the circuit is in an extreme working environment and a glitch sends it into an undesired state, the circuit may never get back to its normal operating condition.

This paper discuses a general methodology when creating a state machine using the HDL Designer Series™ *State Diagram Editor*. You can specify the design so that synthesis tools will not optimize away those unused states, and thus a "safe" state machine can be generated.

Topics

Definition and Overview

1. 'Safe' and 'Unsafe' State Machines
2. Hard-encoded 'Safe' State Machines
3. Creating 'Hard' Encoded 'Safe' State Machines Using the State Diagram Editor

Self-Test

Correct Answers

Glossary

1. 'Safe' and 'Unsafe' State Machines

Not all state machine designs are "unsafe." "Safe" depends on how many states are in a design and how you define the state encoding styles:

"Safe" State Machines

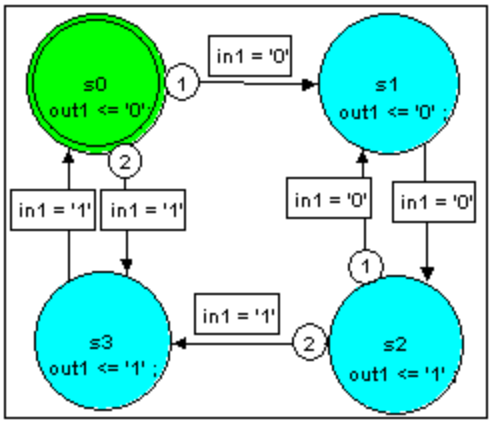
If the number of states (N) is a power of 2 and you use a binary or gray-code encoding algorithm, then the state machine is "safe." This ensures that you have M number of registers where N = 2M. Because all of the possible state values (or register statuses) are reachable, the design is "safe."

"Unsafe" State Machines

If the number of states is not a power of 2, or if you do not use binary or gray-code encoding algorithm (e.g., one-hot), then the state machine is "unsafe."

For example, Figure 1 shows a design that contains four states:

Figure 1: NRZ-to-Manchester Conversion



The number of states is four if you use the one-hot encoding algorithm. For example:

```
s0 => 0001
s1 => 0010
s2 => 0100
s3 => 1000
```

Thus there are 12 more states that are not defined. Generally, these states are covered by the "`others`" (for VHDL) or "`default`" (for Verilog) branch of the case statement. The default operation of the synthesis tool will optimize away unreachable states in order to get a high-performance circuit. However, the optimization will create an "unsafe" circuit.

2. Hard-encoded 'Safe' State Machines

Regardless of how many states you have, if you use the "bit-level" encoding scheme, the optimized result will be "safe." This requires you to specify bit patterns for your states. For example, in VHDL use `std_logic_vector` to define your state type, then you can detect the undesired states using the "`others`" statement in the state decoding process or by explicitly defining if:
`current_state = (undesired states) or current_state /= (desired states).`

For the example shown in Figure 1, you can use the following statements to declare the state values:

```
SUBTYPE STATE_TYPE IS std_logic_vector(3 DOWNTO 0);

CONSTANT s0 : STATE_TYPE := "0001";
CONSTANT s1 : STATE_TYPE := "0010";
CONSTANT s2 : STATE_TYPE := "0100";
CONSTANT s3 : STATE_TYPE := "1000";

SIGNAL current_state : STATE_TYPE ;
SIGNAL next_state : STATE_TYPE ;
```

The example uses one-hot encoding; it also applies to any other encoding algorithm.

3. Creating 'Hard' Encoded 'Safe' State Machines Using the State Diagram Editor

The *State Diagram Editor* of the HDL Designer Series fully supports creating hard-encoded "safe" state machines as previously described. To achieve this result, the following options need to be set in the *State Machine Properties* dialog box:

Figure 2: Menu: State Machine Properties > Generation

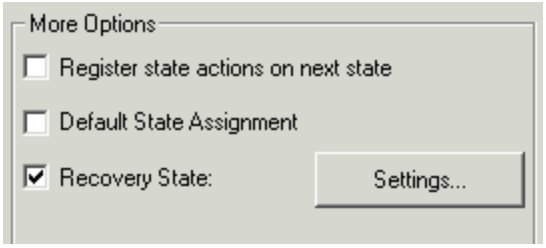
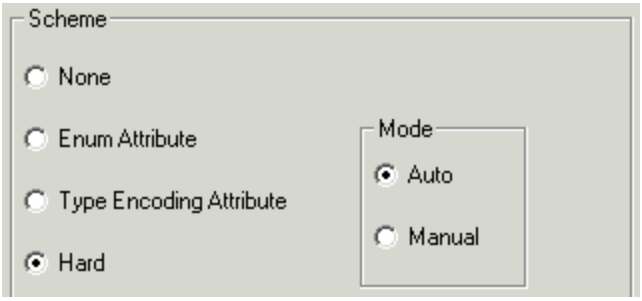


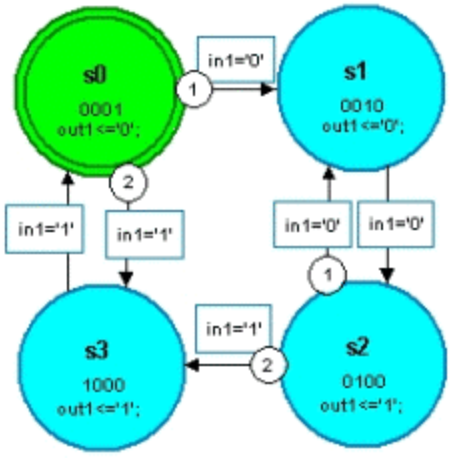
Figure 3: Menu: State Machine Properties > Encoding > Generation



The *Recovery State* can be defined as *<start_state>, <current_state>*, or a specific state. In the generated HDL, this will be the "others" (for VHDL) or "default" (for Verilog) branch of the case statement. The encoding mode for *Hard* can be *Auto* or *Manual* . If *Auto* is chosen, the width of STATE_TYPE will be calculated by using the least number of bits based on the number of states. If *Manual* is selected, you must define the value for each state in the State Diagram Editor, and all the values must have the same size.

Figure 4 shows a "hard" manual encoded state machine using the one-hot scheme.

Figure 4: NRZ-to-Manchester Conversion Using the Hard Manual One-Hot Encoding Scheme



Self-Test

1. An unsafe state machine design is always a malfunction.

    a. True

    b. False

2. Basically, a safe state machine design costs more logical gates.

    a. True

    b. False

3. A safe state machine design must have N states where N = 2**M, and M is a positive integer.

    a. True

    b. False

4. A state machine design that is encoded using a binary or gray encoding algorithm is always safe.

    a. True

    b. False

5. A state machine design using the one-hot encoding style is always unsafe.

    a. True

    b. False

6. You can use any kind of encoding style to create a safe state machine design, provided that you use the "hard-coded" to make sure that all of the states (valid or invalid) are reachable.

    a. True

    b. False

7. If a state machine has eight valid states, which state machine encoding styles should be used to create a safe state machine?

    a. All encoding algorithms can be used

    b. The binary or gray encoding styles

    c. one-hot

    d. all encoding styles, provided that you use the bit-level encoding scheme!

    e. none of the above

8. By using HDL Designer Series tools, you can create a safe state machine design only when you choose "Hard" mode and select a "Recovery State"

    a. True

    b. False

9. If a design has five states, when you select 'Hard' 'Auto' mode, how many bits of STATE_TYPE will the tools of HDL Designer Series pick?

    a. 3

    b. 4

    c. 5

    d. none of the above

10. In the same design as in Question 9, when you select 'Hard' 'Manual' mode, how many bits of STATE_TYPE can you use?

    a. 3

    b. 4

    c. 5

    d. all of the above

Correct Answers

1. An unsafe state machine design is always a malfunction.

    a. True

    b. False

2. Basically, a safe state machine design costs more logical gates.

    a. True

    b. False

3. A safe state machine design must have N states where N = 2**M, and M is a positive integer.

    a. True

    b. False

4. A state machine design that is encoded using a binary or gray encoding algorithm is always safe.

    a. True

    b. False

5. A state machine design using the one-hot encoding style is always unsafe.

    a. True

    b. False

6. You can use any kind of encoding style to create a safe state machine design, provided that you use the "hard-coded" to make sure that all of the states (valid or invalid) are reachable.

    a. True

    b. False

7. If a state machine has eight valid states, which state machine encoding styles should be used to create a safe state machine?

    a. All encoding algorithms can be used

    b. The binary or gray encoding styles

    c. one-hot

    d. all encoding styles, provided that you use the bit-level encoding scheme!

    e. none of the above

8. By using HDL Designer Series tools, you can create a safe state machine design only when you choose "Hard" mode and select a "Recovery State"

    a. True

    b. False

9. If a design has five states, when you select 'Hard' 'Auto' mode, how many bits of STATE_TYPE will the tools of HDL Designer Series pick?

    a. 3

    b. 4

    c. 5

    d. none of the above

10. In the same design as in Question 9, when you select 'Hard' 'Manual' mode, how many bits of STATE_TYPE can you use?

    a. 3

    b. 4

    c. 5

    d. all of the above

Glossary

Acronyms Guide

HDL
Hardware Description Language

NRZ
Non–Return-to-Zero

VHDL
VHSIC (Very-High-Speed Integrated Circuit) Hardware Description Language

Definitions

NRZ–to-Manchester Conversion
As in the example shown, this is a Moore sequential network converting an NRZ–coded bit stream to a Manchester-coded bit stream