An Introduction to FSM Path Coverage

Technical Backgrounder

Jeffrey Barkley, Marketing Manager
Simon Green, Engineering Project Manager
TransEDA

Introduction

With verification now consuming the majority of a chip s development time and resources, coverage analysis technology has emerged to give designers more information on, and control over, the verification process. This has enabled Coverage Closure which is defined as reaching a set of consistent, specific coverage targets during verification, to become an achievable and important milestone on the road to design success.

While coverage analysis has evolved into being a standard part of most design flows, it is still evolving and advancing with powerful new capabilities. This paper presents a new addition to the coverage metrics commonly available today. It is focused on FSMs (finite state machines) within the design and how an analysis of them can provide some insight into their functionality. This paper will first present some definitions for FSM coverage, then introduce a new FSM path coverage metric and finally walk through some examples contrasting this new metric with the currently available alternatives illustrating the critical link between the FSM s functionality and the new metric.

FSM Coverage

Finite state machine (FSM) coverage deserves special attention given the important role FSM's play in describing the control functionality in many designs. It has been shown in several papers how developing tests to increase FSM coverage has detected difficult-to-find bugs.

Traditional coverage analysis tools have provided the following FSM metrics:

- **Visited state coverage**, which ensures that every state of an FSM has been visited
- **Arc coverage**, which ensures that each arc between FSM states has been traversed.

Design teams often aim for over 95% coverage on these metrics, but it is possible to achieve even 100% state and arc coverage without exercising all functionality of the FSM. Thus visited state and arc coverage do not necessarily measure the extent to which the functionality of the FSM has been verified. As a result coverage analysis tools have introduced a metrics referred to as FSM sequences or paths to measure a path through a sequence of visited states or arcs.

FSM Path Coverage

With this new metric comes the difficult task of describing and measuring an FSM path. Consider a simple FSM such as the one shown in the figure below.

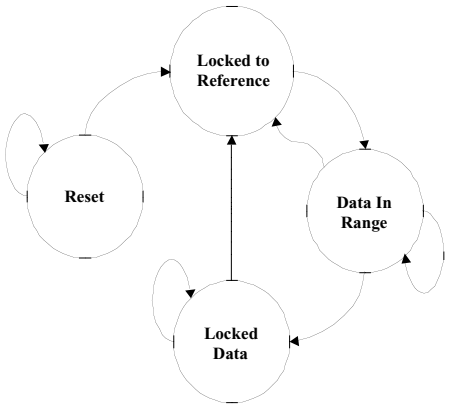

Figure 1: State diagram of a simple state machine

Looking at the FSM you can see that it has the following functionality:

- Wait in Reset until locked to a reference
- Acquire data
- If the reference is lost go to Locked to Reference

A series of arcs that traverses every arc in the FSM at least once is known as an **arc tour**. Completing an arc tour of the FSM would involve the following path of state and arcs through the FSM:

Copyright 2002© TransEDA, Inc.

1. Reset→Reset
2. Reset→Locked_to_Reference
3. Locked_to_Reference→Data_In_Range→Locked_to_Reference
4. Locked_to_Reference→Data_In_Range→Locked_Data→Locked_to_Reference

Figure 2: Main functionality of the FSM

These are the basic paths through the FSM and embody the main functionality of the FSM. There are two arcs missing that are needed to complete the tour; they are the same state loop in the states Data_In_Range and Locked_to_Reference.

Reachability Based Path Extraction

Some coverage analysis tools automatically perform a reachability analysis of the FSM s states. From this analysis they then derive what they call sequences which is the path between each state and every other reachable state. For this simple FSM other than Reset every state is reachable from every other and these tools would generate nine different paths.

1. Reset→Locked_to_Reference
2. Reset→Locked_to_Reference→Data_In_Range
3. Reset→Locked_to_Reference→Data_In_Range→Locked_Data
4. Locked_to_Reference→Data_In_Range
5. Locked_to_Reference→Data_In_Range→Locked_Data
6. Data_In_Range→Locked_Data
7. Data_In_Range→Locked_to_Reference
8. Locked_Data→Locked_to_Reference
9. Locked_Data→ Locked_to_Reference→Data_In_Range

Figure 3: Paths based on reachability

For this simple FSM there are already nine paths and this analysis has not accounted for the same-state loops on Data_In_Range and Locked_Data.

The reachability analysis does not account for these same-state loops in generating the paths. However, if the analysis does not account for same-state loops the FSM s ability to wait will have not been verified. Since the reachability analysis does not know how many times the same-state loop may be taken it cannot account for these loops as part of its analysis. So while this automatic method saves you the preparation time it misses the critical same-state loops and increases your analysis effort and time as you filter out the redundant paths and generate paths for the additional important functionality.

Manual Path Specification

Other coverage analysis tools do not provide any automatic methods of path extraction, but they do provide you a manual way to specify the path using a series of states and some wildcards (*=0+, ?=1+). This increases your preparation time as you identify and specify these paths, however it saves you analysis time.

For the above FSM you would need to manually specify the following:

1. Reset→Reset?
2. Reset?→Locked_to_Reference
3. Locked_to_Reference→Data_In_Range?→ Locked_to_Reference
4. (Locked_to_Reference→Data_In_Range?)?→Locked_Data? →Locked_to_reference
5. Locked_to_Reference→Data_In_Range→Data_In_Range?
6. Locked_to_Reference→Data_In_Range→Locked_Data→Locked_Data?

Figure 4: Paths based on manual specification

When you have manually specified the paths with wildcards there are fewer path than generated automatically by reachability analysis, but you needed to study the FSM and then manually describe the paths to the coverage tool.

TransEDA's FSM Path Approach

The FSM path coverage metric in TransEDA s VN-Cover" Coverage Analysis solution automatically provides a compact, meaningful set of FSM paths enabling deeper insight into a design during verification. TransEDA s unique approach avoids the overwhelming quantity of paths generated by simple reachability analysis and reduces the time required to specify paths manually. VN-Cover s FSM path metric automatically provides a complete representation of an FSM s functionality while minimizing the complexity. This results in shorter analysis time and improved verification productivity.

VN-Cover automatically extracts all FSM paths and provides a concise metric that fully represents the FSM paths. Critical in this ability to concisely represent FSM paths and their coverage is to:

- Separate the FSM s initialization behavior from cyclic
- Identify cyclic behavior in the FSM
- Combine small cycles into the larger cycles that may be present in a FSM
- Allow measurement of paths reached compared with possible paths

The result of this analysis is that it allows you to focus on the FSM s functionality.

Focus On Functionality

By identifying the cyclic behavior, VN-Cover s FSM path metric provides a representation of the design s

functionality without you being lost in the details. Three types of paths are then used to describe FSM s functionality:

- A **link** is a directed set of states that does not repeat. An example would be an initialization sequence.
- A **cycle** is a directed set of states that returns to its starting state without passing through any state more than once.
- A **supercycle** is a longer cycle that identifies the high level functionality and provides a concise summary of the overall structure of the FSM

Supercycles and cycles allow patterns representing important functionality in the FSM to be automatically recognized. Supercycles report design functionality that is not always apparent to the user when looking at the FSM. Using VN-Cover on the simple FSM example it would report that there are three supercycles.

Supercycle 1

The first supercycle would be the Reset state s same-state loop.

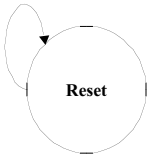

Figure 5: State diagram of Supercycle 1

This represents the functionality of the FSM to wait in Reset until it has Locked to Reference. Once it has then this state is ***linked*** to the FSM s main functionality.

Supercycle 2

The second supercycle represents the FSM s main functionality.

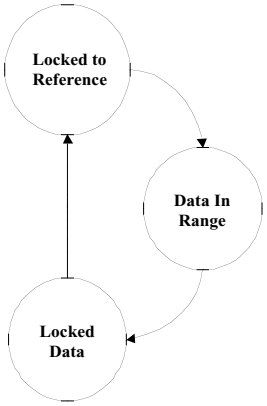

Figure 6: State diagram of Supercycle 2

This is the FSM s correct operational functionality. It describes that the FSM must first be Locked to Reference, then get Data In Range, and finally Locked Data in.

Supercycle 3

The third supercycle would be the FSM s secondary functionality.

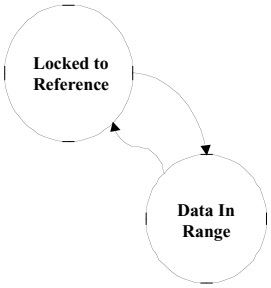

Figure 7: State diagram of supercycle 3

This represents the functionality when an error occurs and the reference is lost. The system has to abandon data acquisition and start over again. This automatic extraction of functionality into supercycles, cycles, and links shortens the analysis time and makes it easier for the design and verification engineer to develop additional vectors for the unverified functionality resulting in shorter verification effort.

Simplify Complexity

Hanging off these supercycles would be several smaller cycles. These cycles represent a wide range of things. They can be temporal characteristics, corner cases, or border conditions of a supercycle s functionality.

For supercycle 2 shown in the above there would be two cycles. The first cycle would be the same-state loop that would occur while the FSM was in state Data In Range waiting for the data to come into range.

Copyright 2002© TransEDA, Inc.

Figure 8: State diagram for cycle 1

The second cycle would be the same-state loop that would occur while the FSM was in state Locked Data waiting for the data to be locked into the system.

Figure 9: State diagram for Cycle 2

For supercycle 3 shown in the above there would be one cycle. The cycle is in common with Supercycle 2 and is the same-state loop that would occur while the FSM was in state Data In Range waiting for the data to be in range.

Functionally, this cycle could be covered by either supercycle as in both cases the path to this cycle is the same: Locking the reference an waiting more than one clock period in the Data In Range state.

This shows that the three supercycles:

- Reset
- Locked_to_Reference→Data_In_Range→Locked _Data
- Locked_to_Reference→Data_In_Range

And the following two cycles

- Data_In_Range
- Locked_Data

Describe the functionality of the FSM. The only thing that is missing now is the link from Reset to the main functionality of the FSM.

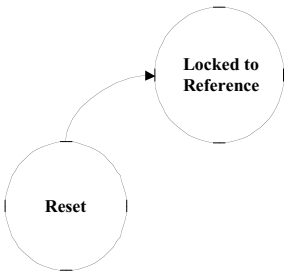

Figure 10: Link from the Reset state to the initial state

Including this link will provide you with the complete representation of the state machine:

- Reset→Locked_to_Reference

This new FSM path coverage metric, supported by the concept links, cycles and supercycles, is a valuable new measure of verification completeness necessary for Coverage Closure.

A Complex Example

Many people consider the JTAG TAP Controller to be a relatively simple FSM. However, if you look at the figure below and considered enumerating all of the paths of states and arcs through it you would quickly decide that maybe it is not so simple after all.

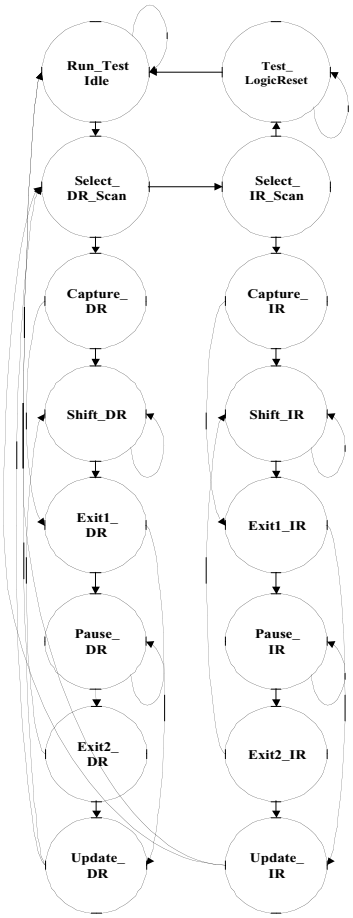

Figure 11: State Diagram of the TAP Controller

Using VN-Cover s FSM Path analysis on this FSM it detects two simple supercycles that represent the overall functionality of the TAP controller:

1. Test_Logic_Reset same-state loop
2. Test_Logic_Reset→Run_TestIdle→Select_DR_Scan→Select_IR_Scan

Figure 12: TAP controller's supercycles

The first supercycle is the Run_Test1Idle same-state loop and it represents the important functionality of the FSM to wait in the idle state.

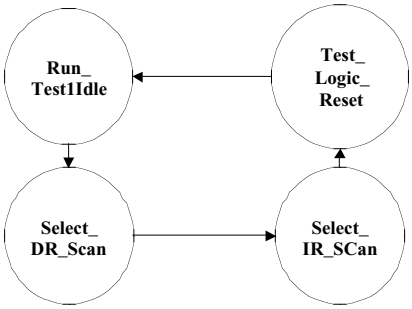

Figure 13: TAP controller supercycle 2

The second supercycle represents the overall functionality of the FSM, which is to select and scan the Data Register and or select and scan the instruction register.

VN-Cover s path analysis also detected the smaller cycles that are the paths through the data and instructions registers:

- Cycles through the Data Register that start and end at Run_TestIdle.
- Cycles through the Instruction Register that start and end at Run_TestIdle.
- Cycles through the Data Register that start and end at Select_DR_Scan.
- Cycles through the Instruction Register that start and end at Select_DR_Scan.

There were also some smaller same state loops and two smaller loops one each in the instruction and data register. The full set of cycles is included in Appendix A.

VN-Covers FSM path analysis resulted in the following:

- Two Supercycle representing the FSM s main functionality
- Twenty-three Cycles representing the functionality of the smaller cycles in the FSM
- There are no Links in this FSM.

Taken together these FSM paths clearly describe the functionality of the FSM in easy to understand pieces. This higher level of abstraction provides you with improved verification productivity and shorter analysis time because you can easily understand what is the unverified functionality

If you had used the reachability analysis available from other coverage analysis tools where there would have been a total of 240 FSM paths. Most of these 240 would be redundant sequences, but it still would not have included the behavior represented by the same-state loops.

You can also contrast this with having to analyze the FSM and then manually write your own sequences. If you did this you would find that the best that you can create by hand are 25 paths that are almost identical to the ones automatically generated by VN-Cover. You can see a listing of those paths in Appendix B.

Conclusion

This paper has described how TransEDA s FSM path coverage metric provides an automatic and efficient way to see, understand and verify the functionality represented by FSMs. This unique coverage metric provides users with more useful information than is traditionally found in coverage analysis tools, enabling designs to be released sooner and with more confidence.

Copyright 2002© TransEDA, Inc.

Appendix A: VN-Cover Cycles for TAP Controller

VN-Cover automatically extracts the following cycles:

1. Run_TestIdle→Select_DR_Scan→Capture_DR→Shift_DR→Exit1_DR→Pause_DR→Exit2_DR→Update_DR
2. Run_TestIdle→Select_DR_Scan→Capture_DR→Shift_DR→Exit1_DR→Update_DR
3. Run_TestIdle→Select_DR_Scan→Capture_DR→Exit1_DR→Pause_DR→Exit2_DR→Update_DR
4. Run_TestIdle→Select_DR_Scan→Capture_DR→Exit1_DR→Update_DR
5. Run_TestIdle→Select_DR_Scan→Select_IR_Scan→Capture_IR→Shift_IR→Exit1_IR→Pause_IR→Exit2_IR→Update_IR
6. Run_TestIdle→Select_DR_Scan→Select_IR_Scan→Capture_IR→Shift_IR→Exit1_IR→Update_IR
7. Run_TestIdle→Select_DR_Scan→Select_IR_Scan→Capture_IR→Exit1_IR→Pause_IR→Exit2_IR→Update_IR
8. Run_TestIdle→Select_DR_Scan→Select_IR_Scan→Capture_IR→Exit1_IR→Update_IR
9. Select_DR_Scan→Capture_DR→Shift_DR→Exit1_DR→Pause_DR→Exit2_DR→Update_DR
10. Select_DR_Scan→Capture_DR→Shift_DR→Exit1_DR→Update_DR
11. Select_DR_Scan→Capture_DR→Exit1_DR→Pause_DR→Exit2_DR→Update_DR
12. Select_DR_Scan→Capture_DR→Exit1_DR→Update_DR
13. Select_DR_Scan→Select_IR_Scan→Capture_IR→Shift_IR→Exit1_IR→Pause_IR→Exit2_IR→Update_IR
14. Select_DR_Scan→Select_IR_Scan→Capture_IR→Shift_IR→Exit1_IR→Update_IR
15. Select_DR_Scan→Select_IR_Scan→Capture_IR→Exit1_IR→Pause_IR→Exit2_IR→Update_IR
16. Select_DR_Scan→Select_IR_Scan→Capture_IR→Exit1_IR→Update_IR
17. Run_TestIdle
18. Pause_DR
19. Shift_DR
20. Pause_IR
21. Shift_IR
22. Exit1_DR→Pause_DR→Exit2_DR→Shift_DR
23. Exit1_IR→Pause_IR→Exit2_IR→Shift_IR

Figure 14: TAP controller's paths automatically generated by VN-Cover

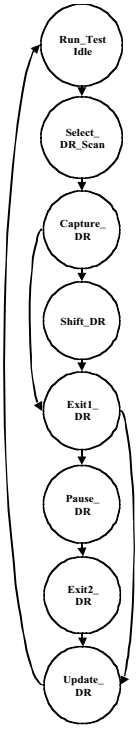

Figure 15: TAP controller cycles 1 through 4

Looking at the cycles 5 through 8 you get the same set, but for the instruction register.

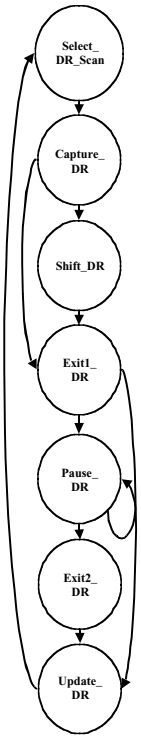

Figure 16 :TAP controller cycles 9 through 12

Looking at the cycles 13 through 16 you get the same set, but for the instruction register.

Copyright 2002© TransEDA, Inc.

There are also some same-state loops with each of the Supercycles that enable the state machine to wait. Lastly there are two cycles that the backward loop contained in both the data and instruction register.

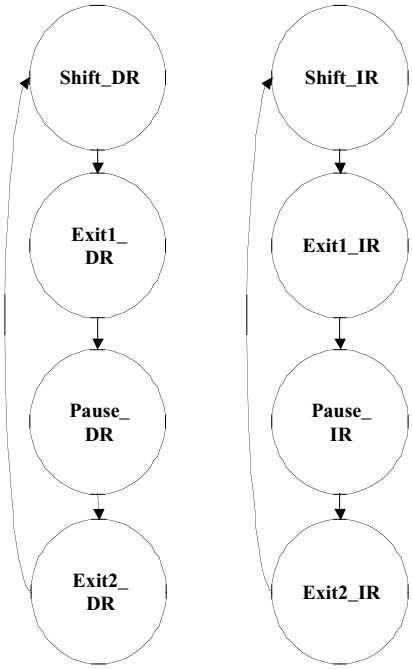

Figure 17: TAP controller cycles 22 and 23

Appendix B: Manually Created Paths For TAP Controller

If you analyzed the TAP controller and then wrote a set of paths to full represent its functionality the best that you can create by hand are 25 paths. You will notice that these paths are almost identical to the ones automatically generated by VN-Cover:

1. Test_Logic_Reset→Test_Logic_Reset?
2. Test_Logic_Reset→Run_TestlIdle?→Select_DR_Scan→Select_IR_Scan
3. Run_Test1Idle→Select_DR_Scan→Capture_DR→Shift_DR?→Exit1_DR→Pause_DR?→Exit2_DR→Update_DR→Run_Test1Idle
4. Run_Test1Idle→Select_DR_Scan→Capture_DR→Shift_DR?→Exit1_DR→Update_DR→Run_Test1Idle
5. Run_Test1Idle→Select_DR_Scan→Capture_DR→Exit1_DR→Pause_DR?→Exit2_DR→Update_DR→Run_Test1Idle
6. Run_Test1Idle→Select_DR_Scan→Capture_DR→Exit1_DR→ Update_DR→Run_Test1Idle
7. Run_TestIdle→Select_DR_Scan→Select_IR_Scan→Capture_IR→Shift_IR?→Exit1_IR→Pause_IR?→Exit2_IR→Update_IR→Run_Test1Idle
8. Run_TestIdle→Select_DR_Scan→Select_IR_Scan→Capture_IR→Shift_IR?→Exit1_IR→Update_IR→Run_Test1Idle
9. Run_TestIdle→Select_DR_Scan→Select_IR_Scan→Capture_IR→Exit1_IR→Pause_IR?→Exit2_IR→Update_IR→Run_Test1Idle
10. Run_TestIdle→Select_DR_Scan→Select_IR_Scan→Capture_IR→Exit1_IR→Update_IR→Run_Test1Idle
11. Select_DR_Scan→Capture_DR→Shift_DR?→Exit1_DR→Pause_DR?→Exit2_DR→Update_DR→Select_DR_Scan
12. Select_DR_Scan→Capture_DR→Shift_DR?→Exit1_DR→Update_DR→Select_DR_Scan
13. Select_DR_Scan→Capture_DR→Exit1_DR→Pause_DR?→Exit2_DR→Update_DR→Select_DR_Scan
14. Select_DR_Scan→Capture_DR→Exit1_DR→Update_DR→Select_DR_Scan
15. Select_DR_Scan→Select_IR_Scan→Capture_IR→Shift_IR?→Exit1_IR→Pause_IR?→Exit2_IR→Update_IR→Select_DR_Scan
16. Select_DR_Scan→Select_IR_Scan→Capture_IR→Shift_IR?→Exit1_IR→Update_IR→Select_DR_Scan
17. Select_DR_Scan→Select_IR_Scan→Capture_IR→Exit1_IR→Pause_IR?→Exit2_IR→Update_IR→Select_DR_Scan
18. Select_DR_Scan→Select_IR_Scan→Capture_IR→Exit1_IR→Update_IR→Select_DR_Scan
19. Run_TestIdle.Run_TestIdle?
20. Pause_DR.Pause_DR?
21. Shift_DR.Shift_DR?
22. Pause_IR.Pause_IR?
23. Shift_IR.Shift_IR?
24. Exit1_DR→Pause_DR?→Exit2_DR→Shift_DR?
25. Exit1_IR→Pause_IR?→Exit2_IR→Shift_IR?

Figure 18: Manually generated FSM paths

Copyright 2002© TransEDA, Inc.