Hierarchical Control FSMs to Enhance Control Path Coverage

Sathyanarayan Balaji

Ittiam Systems Pvt Ltd, Bangalore, India

sathyanarayan.b@ittiam.com

ABSTRACT

Control path coverage and verification is one of the greatest challenges faced by verification engineers today. This paper proposes the use of hierarchical FSM in designing control path as a means to enhance control path coverage.

Systems are generally designed with modules in a hierarchical fashion. This work proposes the use of an FSM to control each datapath module, unless the control is extremely trivial. The FSMs at the lower level are controlled by the FSMs at the higher level. The resulting control tree provides better control path coverage.

Random control logic is generally tightly coupled to the datapath, thereby necessitating complex directed test vectors for achieving required coverage metrics. Control paths designed through hierarchical FSMs give better controllability and observability; thereby giving an edge over the traditional random control logic based design, and can even be tested in a standalone setup.

1.0 Introduction

Control path coverage and verification is one of the greatest challenges faced by verification engineers today. There is an increasing need to tackle the problem of completely verifying large and complex designs. Cut-throat competition necessitating faster time-to-market adds another dimension to this problem. More often than not, teams yield to 'chip release' pressure and the result is devastating – chip failures, time effort and money wasted in chip testing, debugging and chip re-spin. Many companies, in the past, have seen their bottom-line recede due to the above reasons.

This paper talks about a design methodology for control path that aids in faster and easier verification. Section 2 discusses about control path verification. The concept of design for verification is discussed in section 3. Section 4 talks about FSMs as a means to design control path, with emphasis being laid on the use of hierarchical FSMs. Section 5 talks about test beds for such an approach.

2.0 Control Path Verification

A plethora of methods are available for quantitative analysis of verification completeness. Almost all the simulation tools provide options to obtain coverage metrics. Based on the coverage report, engineers can prepare directed test vectors to exercise the untested portion of the design and thereby achieve better code coverage. Although 100% coverage does not guarantee an error free design, to some extent it provides a systematic or a quantitative measure of the completeness of verification process.

Often it has been found that the verification complexity increases exponentially with the increase in control path complexity. Verifying the data path is considerably simpler due to availability of reference models. That is not the case however with control path. It is generally not easy to generate test patterns that can help achieve 100% coverage, especially for complex multi-level test logic implemented in HDL. Obtaining 100% coverage for control path is a tedious task. References [2][3] discuss such nodes in the control tree, which they refer to as 'hard-to-test' (HTC) nodes.

3.0 Design for Verification

The problem of control path verification is similar to what is faced in post-fabrication testing. Such issues are overcome in post-fabrication testing by addressing them right at the design phase. 'Design for testability' (DFT) helps in reducing the test pattern length and hence the testing time. This is achieved in DFT by introducing additional circuit components on-chip that enhances the observability and controllability of the various portions of the chip. A similar approach can be adopted to reduce the complexity of functional verification. Such an idea is already floating around and verification gurus call it 'design for verification' or DFV.

In most of the ASIC designs, emphasis is laid on the datapath – be it for verification or optimization. As a usual practice, designers tend to implement control path in the form of random logic that is closely or rather tightly bound to the datapath. This method of tightly binding the datapath and control path, though technically or functionally correct, has its own disadvantage. The following figure illustrates how generally control logic is implemented tightly bound to the datapath:

```
…
…
if (time == 10'd1n && condition_A == 1'b1)
begin
        perform_operation_A_on_data;
end
else if (time == 10'd2n && condition_B == 1'b1)
begin
        perform_operation_B_on_data;
end
else if (time == 10'd3n && data == n'dpqr)
begin
        perform_operation_C_on_data;
end
else
begin
        if (time == 10'd4n && data == n'dstu)
        begin
                perform_operation_D_on_data;
        end
        else
        begin
                perform_operation_E_on_data;
        end
end
…
```

Figure 1 Control Path implemented as Random Logic

Implementing control path so tightly bound to datapath has the following disadvantages:

1. It reduces the readability (read understandability) of the code. This seriously affects the review phase. The reviewer, thus, needs to spend lot of time deciphering the code, before verifying it against the specifications.
2. Exercising certain nodes in the control tree during functional simulation/verification becomes entirely data dependant and one ends up generating more patterns merely for higher coverage. This necessitates directed vectors to test these parts of the code.

Directed test vector generation requires lot of manual intervention and hence this consumes costly man-hours (may be man-months).

3. Observability of the control path during debugging becomes extremely difficult. Too many signals would be required to be probed. This makes the process complicated and could thereby result in more man-hours spent on this.

4.0 Hierarchical FSM for Control Path Implementation

Most of the non-trivial control logic can be implemented using FSMs. FSMs are an elegant and lucid way for implementing control logic. Most communication transceiver or signal processing ASICs use FSM for high level sequencing of operations. Often, sub-modules involve quite complex control logic by themselves. The control logic of these sub-modules is generally implemented as random logic tightly bound to the datapath. As mentioned before, this method has its own de-merits.

Figure 2 depicts a design with control path tightly bound to the datapath. The control logic of modules A and B are implemented bound to the respective datapath. The control path of the top module again is implemented using random logic and is bound to its datapath and the modules A and B.

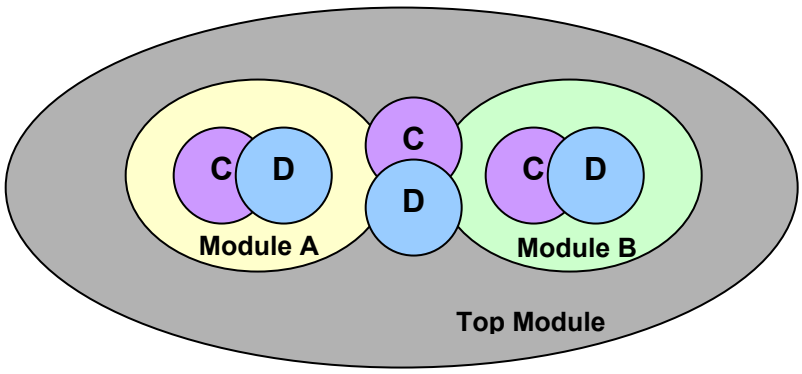

Figure 2 Control Path Tightly Bound to Datapath

In this paper, hierarchical FSMs are proposed as a solution to this problem. In this method, the non-trivial control path of all the sub-modules are implemented using finite state machines. These control FSMs for the sub-modules need to be implemented as separate modules. This method of control path implementation has the following advantage over implementing the same as random logic:

1. The control logic is more organized and grouped into various states. The signal levels in an FSM implementation depend on the current state of the state machine. This

enhances the readability of the code. Also this method helps significantly in the design/code review process as well.

2. Since the level of the various control signals depends on the state, the state encoding bits and the state transition provide valuable information for debugging. This is very helpful especially in debugging using FPGA prototypes, where the number of lines available on the board/FPGA limits the number of signals that can be probed.
3. The most significant advantage of this method is that it enables easier methods for control path verification. This is discussed in detail in section 5. Simulation tools generally identify the FSMs from their language syntax and constructs and they provide a very valuable coverage metric – the state coverage metric. This metric gives the percentage of the state and state transitions covered by the simulations.

Figure 3 depicts a small design with control path designed using hierarchical FSMs. The control logic of modules A and B are implemented through FSM and the corresponding datapath are implemented in separate modules. The control path and the datapath are integrated into modules A and B. The control logic of the top module is again implemented as an FSM (in a separate Verilog module). The top module datapath, control path and the sub-modules are then integrated.

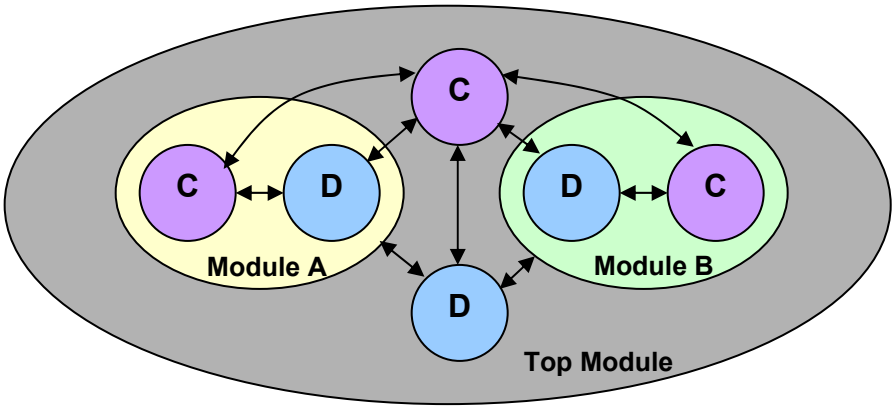

Figure 3 Control Path Implemented using Hierarchical FSMs

5.0 Control Path Verification

As mentioned in section 4, design of control path using hierarchical FSMs helps in achieving verification completeness faster. FSMs for the control path are implemented in separate modules as depicted in figure 3. For control path verification, the FSMs of the top module and the sub-modules alone can be isolated. This is depicted in figure 4. The control path alone can then be integrated and a test-bench prepared for this setup. The signals between the control path and the datapath can now be fed from the test-bench. In addition to the data, the test-bench would also feed the control signals to the top-level control module.

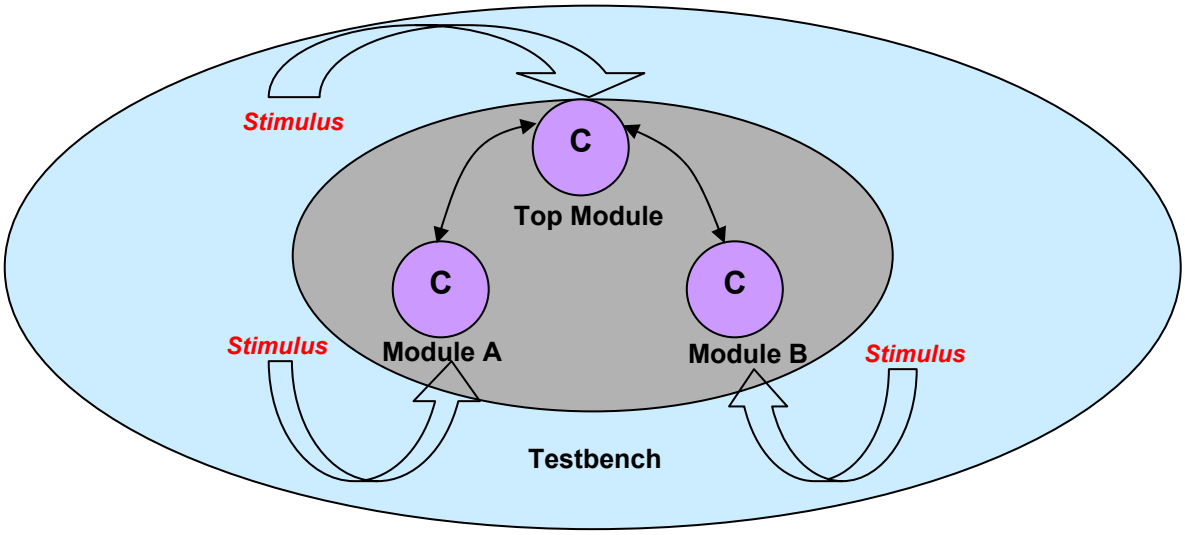

Figure 4 Test Setup for Control Path implemented through Hierarchical FSMs

6.0 Results

This method has been successfully used in some of the ASIC designs. It has been found that this method has aided in verification completeness faster. There has not been an instance where both the traditional random logic implementation of control path and the hierarchical FSMs were tried on the same system. But some analysis was done for 2 of the systems of comparable size and complexity, one implementing control path in a traditional approach and the other implementing hierarchical FSMs. The results were quite encouraging in favor of the proposed method.

7.0 Acknowledgements

I sincerely thank my colleagues Anasuya Pai Maroor, Kedar D Patki, Kiran Devanahalli, Sridhar SR and Varadaraj N Kamath for the help and motivation they provided in publishing this work and for their review comments and inputs that helped refine this paper and the presentation.

8.0 References

[1] Rindert Schutten, Tom Fitzpatrick, “Design for Verification – Blueprint for Productivity and Product Quality”, Synopsys Inc. Whitepaper, April 2003

[2] Frank F. Hsu, Elizabeth M. Rudnick, Janak H. Patel, “Enhancing High Level Control-Flow for Improved Testability”, ICCAD, 1996

[3] Chein-Nan Jimmy Liu, I-Ling Chen, Jing-Yang Jou, “An Efficient Design-for-Verification Technique for HDLs”

[4] Bilung Lee, Edward A. Lee, “Control Logic using Finite State Machines”, Ptolemy Mini-Conference, 1999.