Interviewing a Verification Engineer

by Akiva Michelson, Ace Verification

INTRODUCTION

A key challenge today is choosing the right staff for achieving excellent verification results. Indeed, the defining moment for most projects is when the staff is selected, since the right combination of skills and personality can lead to outstanding technical outcomes (while the wrong combination can lead to disaster). Verification engineers differ significantly from other engineers in terms of skill sets required for success. Due to the nature and breadth of verification tasks, a verification engineer needs to have excellent communication and interpersonal skills, a passion for verification, and the technical know-how to complete tasks in a highly dynamic environment. This article provides a basic interview framework for identifying a capable verification engineer who will work well with your team. Questions about previous projects, verification skills, debug skills and programming skills are described, as well as how to plan the content of the interview, what to look for in the answers, and what traits are most important in a prospective candidate.

THE INTERVIEW FRAMEWORK

The interview framework has three major sections: the project, technical skills, and methodology. The project section aims to explore the experience and personality of the interviewee, the technical section measures both verification language and technical skills in related tasks, and the methodology section measures verification comprehension. This framework can span one or more interviews.

PROJECT:

When interviewing a prospective verification engineer, have him explain his most recent completed project. The following questions can help relate this previous work experience to the engineer's probable fit in your project.

- **Understanding the overall project**
  Verification engineers must possess the ability to explain complex processes in a simple and succinct manner. The verification engineer needs to know how to explain such processes in a way that you, the interviewer, will understand. That is, he will need to take cues from you and give descriptions in a way that will allow you to understand both the overall project and his part in it.

  If you are not able to understand the project or his role in it, then this is not the person you want explaining bugs or discrepancies to your design engineers. Cut the interview short and stop here. However, if you have a clear knowledge of the size, scope, and content of the project, then proceed.

- **Understanding the engineer's role in the project**
  As the interviewee explains the project, try to probe on the scope of his responsibilities. Did he define the methodology? Did he inherit a working verification environment? With whom did he review his design code? Who helped him when he came across problems? If relevant, what helped him collaborate with other verification teams within the organization?

  Here you are trying to gauge both the independence and teamwork skills of the engineer. On the one hand, you want someone who can take ownership of a complete verification environment and manage his own work from start to finish, but you also want an engineer who is not shy about asking for help and reviewing his work with others. Part of working on a verification team is knowing how to align to the team's methodology and how to work within a defined framework. It's important to discern that the engineer will be able to work smoothly within your company's framework.

- **Understanding the difficulties of the project** [1]
  Ask for examples and anecdotes of specific problems the engineer and his team encountered on their projects. Inquire about the solutions pursued and how effective they ultimately were in resolving the problem.

Here we want to understand the difficulties and how the engineer resolved or circumvented them. A good engineer should be able to describe at least three problems and their practical solutions. An engineer who works on perfect projects and never has difficulties hasn't done verification. A good verification engineer will be able to identify problem areas and think of practical solutions, some of which he has implemented in the past. If all the proposed solutions center on how others should change, that should raise red flags about how this engineer works with other people.

TECHNICAL ASSESSMENT[2]:

SystemVerilog/E comprehension

a. Why is a verification language necessary?
See that the engineer knows the components and use of a verification language.

b. <Add specific language questions here>
See if the engineer knows the right place to use the right constructs. Make sure that the engineer has explored beyond template-type coding.

c. Debug skills

- Contradictions – How to debug and how to avoid
- Action not occurring at the right timing - How to debug
- Simulation Crash - How to debug
- Regressions - Have the engineer describe the typical regression and types of failures that occur and how to organize and debug failures

Answers to these four types of questions should help you understand how well the engineer debugs and solves problems, how well he uses the built-in debug tools, and his thought process in thinking through debugging.

d. Libraries - eRM, UVM

- Understand the engineer's experience with verification libraries
- What's happening when you write a sequence
- Ask for the engineer's opinion of the libraries. Make sure he can think for himself, and support his position.

Technical questions often reveal gaps in knowledge, and it is important to evaluate whether these gaps are due to lack of professionalism and training or rather to nuances of how verification tasks are divided in different companies or on different types of projects.[3] The engineer should be open and forthcoming about areas where he lacks knowledge.

Broader skill set

The broader skills to look for in a prospective verification engineer depend on your organization's needs and the skill set you already have on the team. Below is a small sample of skills that, though they lie outside the standard abilities of a verification engineer, can be useful to know and might give the interviewee who possesses them an advantage.

- Unix – Can the engineer install a verification tool, resolve a problem if the disk is full, "tar" a directory, or log in to a remote server in Antarctica and work without a gui or favorite editor?
- Shell commands (| . sed, awk) – Can he manipulate data/results easily?
- Shell scripts (csh, sh) – Can he manage on his own when existing scripts are insufficient?
- Perl – Can he write and/or modify existing Perl scripts to process tables into verification code, or regression results into reports?
- Source Control / LSF / bug tracking – Can he work with tools appropriately? Does he have experience in merging two separate branches?

It is fine for the engineer not to possess some of the skills mentioned above, as long as he is upfront about it and you are confident he will be able to adjust to your team quickly.

Programming skills

Have the interviewee write pseudo-code of a simple program, preferably something with a number of corner cases, and logic that involves multiple expressions. Does he write clean code?[4] Is it readable and correctly written? There should be no time limit for this part of the interview.

Grade this section on the output. Is it logically correct? Would you like to see this type of code in your environments? Is the interviewee open to your comments? Do not grade on the time it takes to write or rewrite the code, just look at the final product. Given that the time it takes to write code is such a small part of verification, someone who takes an hour and gets it right is preferable to someone who takes five minutes and gets it wrong.

METHODOLOGY

The following section aims to discover verification comprehension, or how well the verification engineer can come up with the best solutions for non-textbook projects. Here are some of the questions you can ask to ascertain an engineer's thinking about verification.

- When is use of random good/bad?
- What are the costs/benefits of white box vs. black box testing?
- When should functional coverage be used/skipped?
- Where do you cut when your time is limited?
- How do some FPGA teams develop entire working systems with virtually no verification? Is verifying a prototype the same as verifying a design slated for production?
- Is there truly a complex chip with 0 bugs?

In methodology there are no completely right or wrong answers. Diverse opinions are welcome if they can be supported. In these questions you should be alert for standard answers. Are they just repeating what the local EDA vendor has been saying, or have they actually thought through these questions and reached independent conclusions? These questions are key to differentiating between verification engineers and *verification followers*. Verification engineers will get excited by the discussion; they will have supported opinions and useful insight. The verification followers will just tell you what they have done in the past, or what they believe you want to hear. While it is important to hire engineers who will align to your company's methodology, an engineer is always preferable to a follower.

CONCLUDING THE INTERVIEW

The questions and discussions above serve to get acquainted with the verification engineer, his breadth and depth of knowledge, and his ability to use that knowledge to tackle new challenges creatively. After recording and evaluating the answers to the questions above, the interviewer must also ask himself: Would I enjoy working with this person every day? While this question is very subjective, verification is an extremely interactive profession which requires both technical and interpersonal skills. The verification engineer needs to interact with multiple people to get even the most basic design verified. As such, while technical skills can be mastered, the finesse required when telling an engineer his code has a 'bug' or the gentle persuasion necessary when trying to attain the assistance of a very busy coworker are essential personality traits for making verification projects flow smoothly.

SUMMARY

Verification requires a diverse set of skills. This article has provided a framework for identifying technical skills, methodology comprehension, and project experience. Using part or all of the framework will enable the interviewer to identify the qualities required for his project in prospective engineers.

All prospective verification engineers are welcome to read this article and prepare before the interview. Interviewers should also consider sending the article to prospective employees prior to the interview so they can understand the framework, the purpose of the questions asked, and see what they can accomplish when they prepare in advance for the interview.

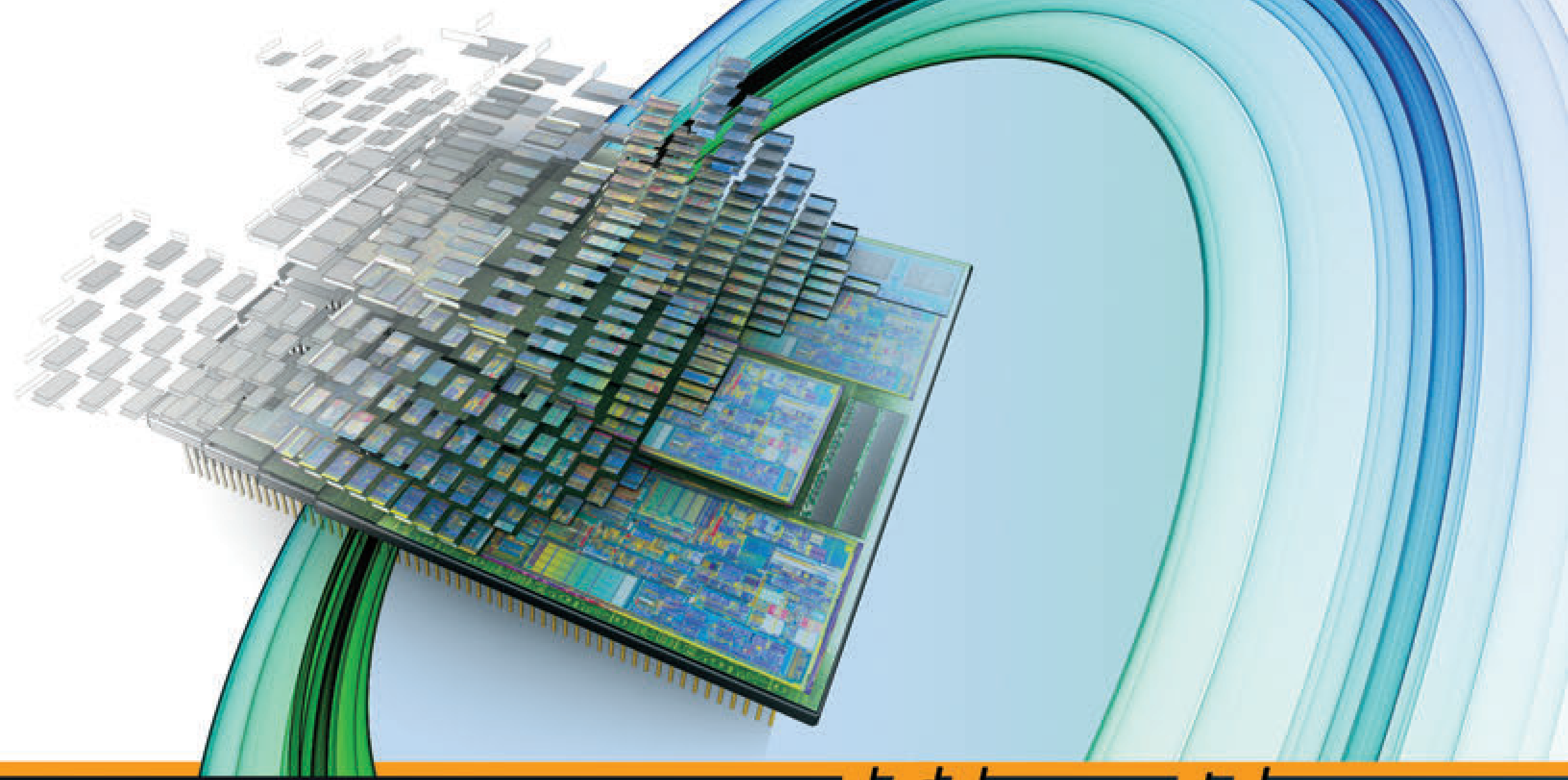
END NOTES

1. Each project poses a different set of challenges, which can range from purely technical issues such as reference model integration to purely interpersonal issues such as personality conflicts between designers.

2. This part of the interview must be given only by an experienced verification engineer.

3. For every type of design project (such as a CPU, networking, PHY, or SoC design) there are radically different sets of verification challenges. Even the most adept engineer experienced in one field may not have specific skills or knowledge commonplace in another.

4. Clean code is code that is easy to read and self-explanatory. In clean code, variables and function names are carefully chosen, comments are used to improve readability, indentation and parentheses make the code readable and help control the flow in clear and simple fashion.



verification HORIZONS

Editor: Tom Fitzpatrick
Program Manager: Rebecca Granquist

Wilsonville Worldwide Headquarters
8005 SW Boeckman Rd.
Wilsonville, OR 97070-7777
Phone: 503-685-7000

To subscribe visit:
www.mentor.com/horizons

To view our blog visit:
VERIFICATIONHORIZONSBLOG.COM

Mentor Graphics®

www.mentor.com