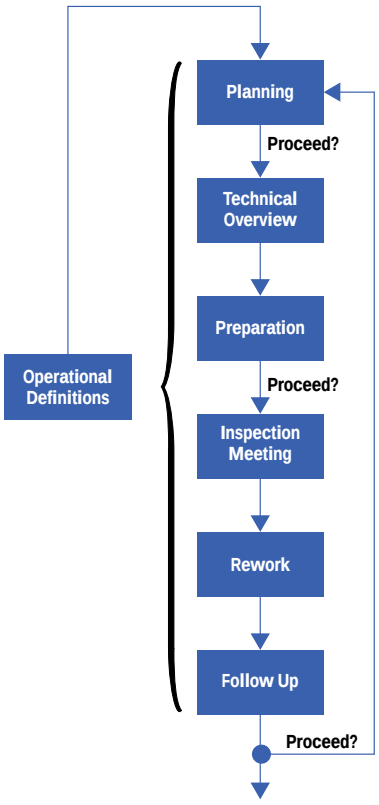
# Applying the Code Inspection Process to Hardware Descriptions

The code inspection process from the software world has been applied to Verilog HDL (hardware description language) code. This paper explains the code inspection process and the roles and responsibilities of the participants. It explores the special challenges of inspecting HDL, the types of findings made, and the lessons learned from using the process for a year.

**by Joseph J. Gilray**

The primary goal of the code inspection process is to maximize the quality of the code produced by an organization. A secondary benefit of the process is that it allows members of the development teams to share best practices. The code inspection process revolves around a formal inspection meeting. The process calls for the development of operational definitions, planning, a technical overview, preparation for the meeting, rework after the meeting, and follow-up. Fig. 1 illustrates the relationships between the steps. The steps themselves are described in the sections that follow. As shown in Fig. 1, the operational definitions affect all stages in the inspection process. Between some of the stages in the inspection process decisions whether or not to continue need to be made. These decisions are indicated on the figure by "Proceed?".

***Fig. 1.*** *the code inspection process.*




The code inspection process as implemented at the HP Integrated Circuits Business Division in Corvallis, Oregon (ICBD Corvallis) contains several roles: process manager, moderator, author, paraphraser (reader), scribe, and inspector. There is only one permanent role, that of inspection process manager. The remaining roles are filled for each inspection. The subsections below call out the general responsibilities and duties of each role. Specific tasks are called out in the description of each process stage later in the paper. HP's software quality engineering department has published checklists for each role that can be very useful when getting started with the process.

**Process Manager.** Ensures that best practices are spread among the designers of the organization or project. Tasks include developing and publishing operational definitions (described in the next section), disseminating best practices and common defects, and acting as an advocate for the HDL code inspection process. This last item cannot be overemphasized. The process manager must ensure that priority is given to inspections even in the face of mounting schedule pressure on the design team. It is also important that the process manager make clear that the specific results (defects found) of the inspections will not be made available to management or any other party. The inspection process can only succeed in an environment where the members of the design team feel secure in opening their code to review. So that management sees the value of the process, the process manager should keep general results of the overall inspection process such as the number and type of defects found, the time spent, lines of code inspected, and most important, best practices shared. These process statistics are very useful, but the grass roots support that develops for the inspection process will be the real indicator of its value.

**Moderator.** Manages each step of the process for a given inspection. Ensures that participants are prepared and that requirements are met.

**Author.** Prepares the HDL for inspection. Creates supplementary documentation (such as block diagrams) as necessary to explain the purpose of the code. Open to suggestions and defects. Reworks the HDL as necessary.

**Paraphraser.** Familiar with guidelines and best practices. Able to explain the HDL code during the inspection meeting.

**Scribe.** Logs defects and enhancements found during the meeting.

**Inspector.** Reads and understands the HDL. Notes any defects, comments, or enhancements before the meeting. Every person involved in the meeting participates as an inspector.

## Development of Operational Definitions

An operational definition is simply a standard. Before an inspection takes place a core set of operational definitions should be in place and recognized by the design team. They are developed from conventions, guidelines, industry standards, and recognized best practices. For HDL code inspections at ICBD Corvallis, we adopted the simplest set of operational definitions that we felt were adequate to guide the process:

- Coding style standards. Although no explicit HDL coding standard was selected, we developed a standard HDL module header (Fig. 2).

***Fig. 2.*** *Standard Verilog HDL module header adopted for code inspections.*

```
// File name
// Module name(s)
// Author name(s)
// Revision log
// File description (why are these modules grouped together)
// . . .
// Module name (for each module)
// Module description
// Signal descriptions (these include all HDL signals, including wires)
//     For each signal specify:
//         – type
//         – purpose
//         – values/states description
//         – invariants (such as tristate nodes that are always driven)
//         – special loading conditions
//         – value at reset
//         – overflow/wraparound conditions (e.g. for counters)
```

- Definition of a defect. We defined a defect as any deviation from the module specification as presented in the technical overview meeting (see below) and the HDL module header.
- Defect severity codes. We applied a simple defect severity scale based on HP's internal Defect Tracking System (DTS), as shown in Table I.

**Table I**
**Defect Severities**

| Name | ID | Description |
|---|---|---|
| Critical | 9 | Defect will lead to unworkable or grossly inefficient design. |
| Serious | 7 | Defect will lead to a large deviation from the specification or to a design that is unreliable or very inefficient. |
| Major | 5 | Defect will lead to a deviation from the specification or to a design that is inefficient. |
| Minor | 3 | Defect will lead to a minor deviation from the specification or to a design that is slightly inefficient. Also used when code is in serious need of comments to be maintainable. |
| Wibni | 1 | "Wouldn't it be nice if...?" This ID is used for enhancement requests, which are typically changes in coding style or requests for clarifying comments in the code. |

- Defect logging standards. We started out using inspection data summary sheets provided by HP's software quality engineering department, but after a few inspections we found that an open-format inspection process and defect logs worked better.
- Target-based best practices. ICBD Corvallis developed a set of Verilog HDL coding guidelines to ensure reliable, high-quality synthesis results. These guidelines include sections on clocking strategies, block structure, latches and registers, state machines, design for test, ensuring consistent behavioral and structural simulation results, and issues specific to Synopsys synthesis tools, which are used extensively by HP. This document provided valuable input to the HDL code inspection process and itself benefitted from the practices shared during the inspections.
- Inspection entry criteria. The inspection entry criteria were that the HDL had to be functionally correct in behavioral simulation and had to be of small-to-moderate size (100 to 700 noncomment Verilog HDL source statements).
- Inspection exit criterion. We did not develop a formal inspection exit criterion. Instead, the moderator was given the responsibility of ensuring that rework was satisfactorily completed for each piece of HDL inspected.

## Planning

When a designer feels that a piece of HDL code is a good candidate for inspection, the designer asks another designer to act as moderator. Together they review the HDL to be inspected to ensure that it meets the entry criteria, especially that the amount of HDL to be inspected is appropriate. In addition, they review any supplementary documentation such as module specifications or block diagrams and discuss what will need to be presented at the technical overview meeting. The moderator, with help from the author, assembles the rest of the inspection team: a paraphraser (reader), a scribe, and up to three additional inspectors. It is the moderator's responsibility to schedule the technical overview meeting and to ensure that the inspection team members are prepared to meet their responsibilities. The moderator should treat the meetings and preparation as a very important requirement for each participant. Every person involved must be prepared—at a code inspection, no one is just an observer.

## Technical Overview

The technical overview meeting should last no more than 90 minutes. Its primary purpose is to allow the author to outline the module(s) to be inspected and to answer questions. The roles are formally assigned during this meeting and the moderator should ensure that all participants understand the roles assigned to them. If there are inexperienced inspection team members, the moderator should take time to explain the operational definitions and to pass out responsibility checklists for each role. Finally, any supplementary documentation and the HDL code itself are distributed to the team. The code should be printed with line numbers so that during the inspection meeting all team members can more easily follow the discussion.

## Preparation

Each member of the inspection team should spend from two to four hours reading over the HDL. Team members should mark possible defects on their copies of the code. Team members should freely discuss the code among themselves but not in a wider context, to protect the privacy of the author. The team should be given at least a week to look over the HDL. During this time the moderator should schedule the inspection meeting. Before the meeting the moderator should ensure that all team members are prepared and can participate in the meeting before allowing the inspection to proceed.

## Inspection Meeting

The inspection meeting is the heart of the process. The moderator must reserve a quiet room for a sufficient amount of time. Typically inspection meetings take from two to three hours. The moderator is also responsible for keeping the meeting on

track so the code can be completely inspected in the time allowed. To start the meeting the scribe should record the amount of preparation time required of each participant. The paraphraser should announce the order in which the code will be inspected. Typically this is top-down or bottom-up. The paraphraser explains each block of code and allows time for each inspector to discuss possible defects or enhancements to that code.

The goals of the meeting may vary somewhat from organization to organization, but typically the major goals are to find defects in the code under inspection and to share best practices among the members of the design or coding team. In our process, we encouraged discussion of any defect or enhancement. Although this does not strictly adhere to the traditional software inspection process, we felt the benefits (improved coding, simulation, and synthesis practices) justified the time spent.[1]

The moderator must ensure that any defect or enhancement is recorded by the scribe and that the inspection team agrees to the severity assigned to each item. To keep the group on track, the moderator should not allow long discussions of the severity of any defect. Where no agreement can be reached, the moderator should assign a severity. If the assignment of a severity code becomes a stumbling block to progress in several meetings, a simpler major/minor severity classification can be adopted as an operational definition.[2] The moderator should keep track of any best practices that come up during the meeting that are not already part of the operational definitions and note any questions raised about related design processes and tools.

## Rework

After the meeting the scribe gives the defect log to the author (and only to the author). It is the author's responsibility to modify the HDL code as appropriate. If the author or the moderator feels that the HDL should be reinspected, another meeting can be scheduled (this should be very rare, and should proceed with a different set of participants in all roles other than the author).

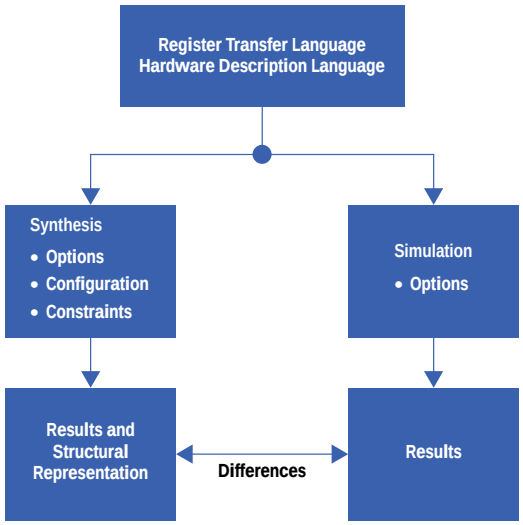
## Follow-up

After each inspection the moderator should investigate any questions that were brought up about design processes and tools, such as simulation and synthesis. The results of the investigation along with any new best practices should be published for the design teams. The moderator and process manager should also review the operational definitions and update them. Finally, the process manager should update the overall inspection process statistics.

## HDL Issues

When inspecting code written in a high-level software language, normally there is a single target compiler and platform. We found that a major difficulty with inspecting code written in a hardware description language was deciding on a target on which to focus. HDL is traditionally targeted to both simulation and synthesis (among a growing list of HDL source-level tools). We started by trying to inspect HDL without thinking in terms of a specific target. In theory, it might be possible to inspect HDL code as an abstract description. In practice, it was nearly impossible. Both the expected simulation results and the actual implementation created by the synthesis process were always on the minds of the inspectors (see Fig. 3). Furthermore, operational definitions such as defect severity are invariably developed and interpreted with reference to a target.

***Fig. 3.*** *HDL is targeted to both simulation and synthesis.*



By the time we had done several inspections it was evident that the most common practices being shared in the meetings were related to register transfer language (RTL) coding for synthesis. Since the synthesis tools are not as mature as either compilers for high-level languages in the software domain or simulators in the hardware domain, we spent a good deal of time discussing what structural elements the synthesis tools would create from the RTL-level HDL code given a set of constraints and synthesis options. This seemed natural given the complexity of the synthesis tools. At times the inspection meetings focused more on the synthesis tools than on the HDL. When writing HDL for synthesis the types of complexities involved are more akin to porting a complex piece of software between frameworks than between compilers. Therefore, it is inevitable that the inspection meetings devote a good deal of time to synthesis. As use of very complex source-level tools such as behavioral synthesis tools becomes widespread this effect will become more pronounced. In fact, one benefit of the HDL inspection process is to share information about the tools used during design creation. This happens less frequently in a traditional software inspection where the compiler is less configurable and better understood. But where the software is targeted at many platforms, this type of discussion occurs in the software domain as well.

Another difference we found between HDL code inspections and software inspections was that often there were questions that couldn't be satisfactorily answered during the inspection, such as "What will the synthesizer produce from the following code (e.g., mixed addition and subtraction of registers with differing widths)?" It was up to the moderator to follow up with the author (or another inspector) on questions that couldn't be answered in the meeting and to write up a response for the design team and for possible inclusion in the best practices guidelines.

Table II indicates the kinds of topics that came up during the inspections and their approximate frequency.

**Table II**
**HDL Inspection Topics**

| Frequency | Topic |
|---|---|
| 35% | HDL coding style, standards, and guidelines (e.g., when to use blocking and nonblocking assignments, etc.) |
| 30% | Structures produced by synthesis tools (HDL compiler, design compiler, finite state machine compiler) |
| 10% | Differences between simulation results and synthesis results |
| 10% | HDL efficiency considerations (e.g., inference of unnecessary latches, use of extra clock cycles) |
| 10% | HDL documentation |
| 5% | HDL block structure |

As more HDL inspections were performed, the number of experienced inspectors grew and the guidelines for creation of HDL for synthesis, which had been created by synthesis users in the lab, became widely disseminated and discussed. Again, one of the primary benefits of the HDL code inspection process is the spread of best practices among the larger group of designers.

## Lessons

As the use of HDL increased in our lab, we noted a need for tools to improve the quality of the HDL produced by the design teams. The lack of HDL source-level tools such as code complexity analyzers, lint (a syntax checker), and others led us to choose a less automated approach. Our first effort at improving the quality of HDL was to develop an HDL code inspection process based on the inspections done for software written in high-level languages.

The process that evolved for inspecting HDL in our lab incorporates elements of both a formal code inspection process and a structured walkthrough process. Although we gave importance to the technical overview meeting, it wasn't always held, especially if inspection team members were offsite. Furthermore, both the rework and the follow-up steps were left to the moderator and author and checked only informally by the process manager.

Early in the adoption of the process we used a set of responsibility checklists for each role. As time went on we found that these were not strictly necessary but did engender a feeling of formality. It is important that the participants take the process seriously to ensure that the time spent on it is not wasted.

Over time we came to realize the importance of the technical overview meeting. If it is impossible for the author to attend the meeting (we ran into several cases where the author was from another site and unable to attend a technical overview) then someone else on the inspection team should take the author's place for the meeting. In cases where we skipped the meeting, the preparation time for each participant increased dramatically. In one case the inspection required 6 to 10 hours of

preparation time. Though the code was fairly long at 900 lines of HDL, this was an unreasonable amount of time to expect from each reviewer and could have been reduced by half had there been a one-hour technical overview held.

In our experience, the most significant benefit of the HDL inspection process was to spread HDL, simulation, and synthesis best practices among the design teams. Not only did the process encourage interaction between various teams within ICBD, but several design teams in HP entities outside of ICBD brought code to us for inspection. To ensure that this benefit is realized it is very important that the process manager and the moderators take the time to publish the guidelines that are developed during each inspection. As designers become proficient at creating HDL and knowledgeable of synthesis and simulation best practices, and as HDL coding guidelines become well-established in an organization, the need to do inspections to spread best practices decreases.

We found relatively few major defects in the HDL code that was inspected, probably because the code was all at the RTL level and simulated and synthesized before inspection. Studies have indicated that the inspection process gives the best results when applied at a high level of abstraction. I contend that we will find more defects if we apply the process to module specifications or to behavioral HDL. If the target chosen is complex (as behavioral synthesis tools currently are) the tendency for the process to focus on the tool instead of the code will also be more pronounced. Even so, applying the inspection process to higher-level abstractions may be a logical next step. Doolan wrote, "As people become aware of the tremendous benefits of the inspection process, there is an increasing desire to apply it to other software items, such as user documentation ... inspection breeds inspection."[2]

## Summary

Reference 3 describes one ICBD Corvallis project that used the HDL inspection process (however, inspections are not discussed in reference 3).

The code inspection process can be applied successfully to hardware descriptions if the following conditions are met:

- A simple set of operational definitions is developed for the process.
- Engineers are willing to open their code to inspection and the process is viewed by the design community as beneficial and important.
- Management gives project teams adequate time to perform inspections.
- Best practices and guidelines are recorded and updated.

For project teams just starting to use hardware description languages in the design process, code inspections can play a vital role in ensuring high-quality HDL. At ICBD Corvallis, we found that the inspection process works extremely efficiently in spreading best practices for HDL coding, simulation, and synthesis.

## References


1. T. DeMarco, *Controlling Software Projects*, 1982, pp. 220-232.
2. E.P. Doolan, "Experience with Fagan's Inspection Method," *Software—Practice and Experience*, February 1992, pp. 173-182.
3. J.D. McDougal and W.E. Young, "Shortening the Time to Volume Production for High-Performance Standard Cell ASICs," *Hewlett-Packard Journal*, Vol. 46, no. 1, February 1995, pp. 91-96.



▶ Go to *Article 9*
▶ Go to Table of Contents
▶ Go to HP Journal Home Page