

Automatic Creation Of VHDL Testbench

Erh-Chiao Charles Wang
NCR Corporation, Torrey Pines Development Center
11010 Torreyana Road
San Diego, CA 92121
E-mail: charles.wang@TorreyPinesCA.ncr.com

Abstract

VHDL testbench creation is considered a tedious task which should be automated as much as possible. For each low level VHDL module, no matter which level of abstraction it is, the designer needs to verify that it behaves properly. In addition, for each higher level module which has sub-modules in it, the designer needs to verify its functionality. As the target system gets more complex, with several level of hierarchies, there are more chances to find bugs in the existing design, even if each low level module works correctly individually. In this case, the designers need to extract part of the system, simulate against it, and trace down to where the problems are. No matter in what situation shown above, it is not practical to feed in vectors by hand at the simulation run time. Creating a VHDL testbench for a target DUT (Design Under Test) is a typical approach. The designer has more flexibility to deal with input vectors, output results, and the design itself. More importantly, the maintainability of the design testbench will benefit the whole design process a great deal over the long term. However, as the designs get more complex, the creation of a testbench is usually as tough as the design itself. To reduce the effort of VHDL testbench creation is the motivation for this work.

At NCR-TPDC, we have investigated different approaches to create the VHDL testbench automatically. This presentation will discuss what we need as inputs in order to create a VHDL testbench. The key issue is how to feed stimuli into the design efficiently. Three different approaches to perform the stimuli feeding job will be presented. Finally, the fundamental idea of the testbench environment that is created by the best approach will be presented as a whole so that it will be understood why the testbench creation can be automated and updated easily. Assuming the test vectors are available, the final result will let the designers turn their VHDL design into a testbench in a push-button manner and concentrate on the design.

There is no serious theory behind this work. The EDA tool vendors might consider including this testbench creation feature in their tools, if they decided on a way to deal with the various stimuli format problem, etc. However, the work presented here might give each VHDL user some inspiration to incorporate this feature into their specific VHDL environment more easily.

Automatic Creation Of VHDL Testbench

VHDL International Users' Forum
October 21, 1992
Washington D.C.

Erh-Chiao Charles Wang
NCR Corporation, Torrey Pines Development Center
11010 Torreyana Road
San Diego, CA 92121
(619) 597-3740
(619) 597-3740 (FAX)
E-Mail: charles.wang@TorreyPinesCA.NCR.COM

Erh-Chiao Charles Wang



Page # 1

- Verify your VHDL design - Simulation (Three common methods)

- Assign the stimuli while the simulator is invoked.

```
% vhdlsimulator DESIGN
Some Brand VHDL Simulator V.X.x -- 1992
# cd DESIGN; trace SIGNAL_1, SIGNAL_2;
# assign '1' SIGNAL_1; assign '0' SIGNAL_2;
# run 20
```

- Include a big simulation script that performs the tests

```
% vhdlsimulator -include run.script DESIGN
Some Brand VHDL Simulator V.X.x -- 1992
-- running the script in the included file
```

Erh-Chiao Charles Wang



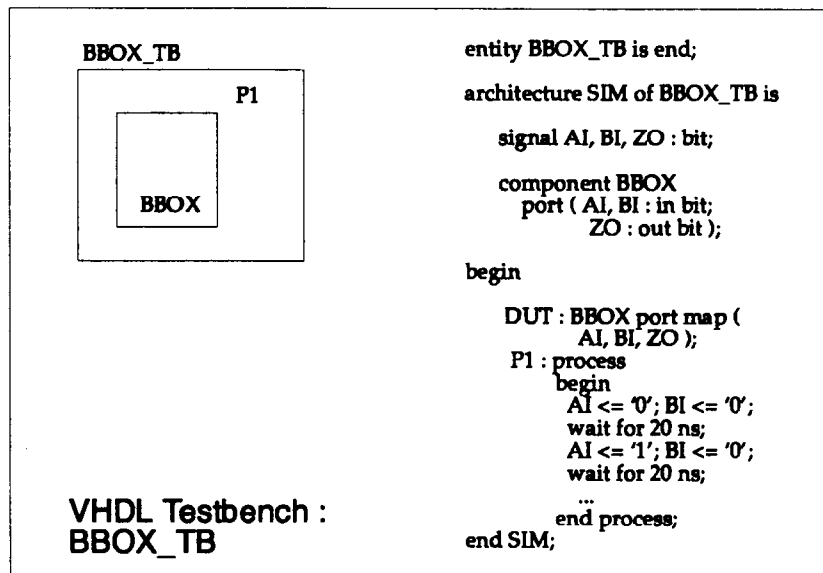
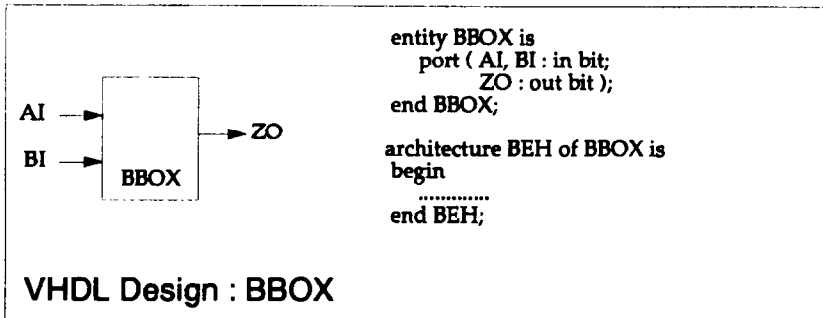
Page # 2

- Create a VHDL testbench that perform the tests itself.

```

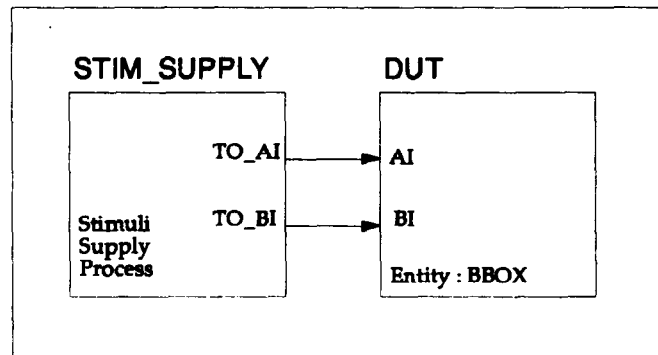
% vhdlsimulator DESIGN_TESTBENCH
    Some Brand VHDL Simulator V.X.x -- 1992
    -- running the testbench to test the DESIGN
  
```

- Example: VHDL Design And Testbench



- Make the stimuli supply process an entity - a test engine

BBOX_TESTBENCH



- At first glance, what information is needed to create such a testbench?
 - Known parts: DUT, stimuli
 - Unknown parts: Stimuli supply module, Testbench.
 - Need: entity declaration of DUT to build the structure shown above, stimuli to fill in the Stimuli supply module.
- Major issue - stimuli input scheme
 - What stimuli?
 - Different tools have different stimuli format.
 - Each stimuli format usually contains more information than we need. (Header, Comment, Alias,...,etc.)

- **Approach One: The most straight forward**
 - **Extract events and time stamps from stimuli file, and then write out directly in VHDL.**

```
P1: process
begin
  AI <= '1'; BI <= '1';
  wait for 20 ns;
  AI <= '0'; BI <= '1';
  wait for 20 ns;
  -- lines deleted....
end process;
```

- **Advantage: Less thought, easier to write the program.**
- **Disadvantage: The size of the stimuli supply module is proportional to the size of stimuli file. When the design is bigger, the stimuli file is larger, it takes longer to compile the stimuli supply module. When the stimuli file gets extremely large, it might not compile at all.**

- **Approach Two: Read in the stimuli file from the stimuli supply module.**

- **Advantage:**

- 1, The size of the stimuli supply module is fixed.
- 2, It is not necessary to know the contents of the stimuli file to create the testbench. Minor restriction applies.*

- **Disadvantages:**

- 1, It is not easy to parse a stimuli file using VHDL.
- 2, It is more difficult to write a program that can generate a stimuli parser in VHDL automatically.
- 3, It slows down the simulation.

- **Approach Three: Convert the stimuli file to an intermediate file that contains only the necessary information and then use the idea of Approach Two.**

- **Advantages:**

- 1, The size of the stimuli supply module is fixed.
- 2, It is not necessary to know the contents of the stimuli file to create the testbench. Minor restriction applies.*
- 3, Move the stimuli parsing part to the testbench creation program.

* Without knowing the stimuli first, if you have bus ports, you cannot assign single bit of a bus in your stimuli. You must assign the whole vector to a bus, even if most of the bits are unchanged.

- Disadvantage:

1, More thought means more time and more difficult to implement the testbench creation program. But it is a one time effort and definitely worthwhile.

- Approach Three Example:

```
entity STIM_SUPPLY is
port ( TO_AI, TO_BI : out bit );
end STIM_SUPPLY;

architecture SIM of STIM_SUPPLY is
FILE stim : TEXT is in "BBOX.vstim";
-- lines deleted...
```

```
while not (ENDFILE(stim)) LOOP
    READLINE(stim, inline1);
    READ(inline1, timeinc);
    READ(inline1, sigindex);
    READ(inline2, sigval);
-- lines deleted...
case sigindex is
    when 0 => TO_AI <= STR2BIT(inline2( 1
to inline2'LENGTH));
    when 1 => TO_BI <= STR2BIT(inline2( 1
to inline2'LENGTH));
    when others => assert (FALSE) report
"No such signal!"
-- lines deleted...
```

- Testbench creation in practice - i2vhdl_{sim}
 - Input 1: Read in DUT entity information from MCC VHDL compiled database.

- Output 1.1: Stim_supply.vhdl

The ports are all in output mode which are associated with all of the in and inout ports in the DUT. The architecture is a process that reads in the simplified stimuli file.

- Output 1.2: Testbench.vhdl

Non-port entity which instantiates two components: DUT and Stim_supply.

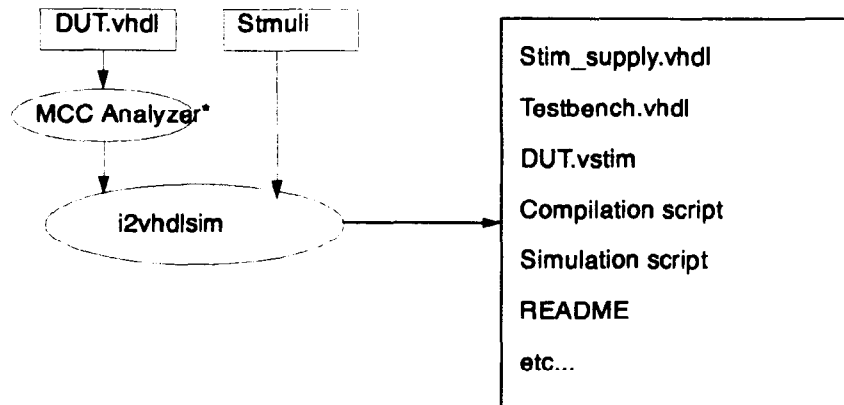
- Input 2: stimuli file

- Output 2.1 : DUT.vstim

This is the simplified version of the stimuli file. It is read in by the process in Stim_supply.vhdl.

- Other output files: User's README file, compilation script, simulation script, simulation auxiliary files, etc.

■ Flow Chart of The Process



* Anything that can extract port information from DUT.vhdl.

■ Conclusion

- In most cases, the designers can turn the VHDL testbench creation into a push-button process.
- In some special cases, the designers only need to modify the simulation script to fulfill the special needs.
- In the worst case, the designers have the generated testbench as a template and it would save a lot of time to tailor it instead of creating one from scratch.