Innovative Technological Narratives
Leveraging the Idea of Authenticity in a Human Being

Anna M. Ravitzki
Uri Feigin
Hagai Arbel
A New Way to Debug

• Vtool is a diverse team of experts – Philosophers, sociologists, engineers, and designers.

• This is how we solve the diversified problem called: Bugs

• Just like the living that fight the dead.

• We made a revolutionary machine, with the purpose of creating an existential revolution.

• We call it Cogita. And it is the new way to debug – it works!
What Debug Really Is

- Verification is a technological tool, for verifying technological innovations using a practical system.
- Without verification, there is no technology.
- Debug is the process of “identifying and removing errors”.
- Debug consists of reading results, analyzing them, and concluding the root cause.
- Specifically, in most cases, debug uses simulation log files.
26656 # UVM_INFO [monitor_axi_write_interface] @ 39530.0ns: => SKIP WRITE - curr_addr = 0xf383333f, wdata[0] = 0x0 (beat_counter: 71)
26658 # UVM_INFO [monitor_axi_write_interface] @ 39530.0ns: => SKIP WRITE - curr_addr = 0xf3833340, wdata[1] = 0x0 (beat_counter: 71)
26659 # UVM_INFO [monitor_axi_write_interface] @ 39530.0ns: => SKIP WRITE - curr_addr = 0xf3833341, wdata[2] = 0x0 (beat_counter: 71)
26660 # UVM_INFO [monitor_axi_write_interface] @ 39530.0ns: => WRITE - curr_addr = 0xf3833343, wdata[4] = 0x3b (beat_counter: 71)
26661 # UVM_INFO [monitor_axi_write_interface] @ 39530.0ns: => *** curr_write_addr ==== f3833194
26662 # UVM_INFO [monitor_axi_write_interface] @ 39530.0ns: => WRITE - curr_addr = 0xf3833344, wdata[5] = 0x75 (beat_counter: 71)
26663 # UVM_INFO [monitor_axi_write_interface] @ 39530.0ns: => *** curr_write_addr ==== f3833195
26664 # UVM_INFO [monitor_axi_write_interface] @ 39530.0ns: => *** curr_write_addr ==== f3833195
26665 # UVM_INFO [monitor_axi_write_interface] @ 39530.0ns: => *** curr_write_addr ==== f3833195
26666 # UVM_INFO [monitor_axi_write_interface] @ 39530.0ns: => *** curr_write_addr ==== f3833195
26667 # UVM_INFO [monitor_axi_write_interface] @ 39530.0ns: => *** curr_write_addr ==== f3833195
26668 # UVM_INFO [monitor_axi_write_interface] @ 39530.0ns: => *** curr_write_addr ==== f3833195
26669 # UVM_INFO [monitor_axi_write_interface] @ 39530.0ns: => *** curr_write_addr ==== f3833195
26670 # UVM_INFO [monitor_axi_write_interface] @ 39530.0ns: => *** curr_write_addr ==== f3833195
26671 # UVM_INFO [monitor_axi_write_interface] @ 39530.0ns: => *** curr_write_addr ==== f3833195
26672 # UVM_INFO [monitor_axi_write_interface] @ 39530.0ns: => *** curr_write_addr ==== f3833195
26673 # UVM_INFO [monitor_axi_write_interface] @ 39530.0ns: => *** curr_write_addr ==== f3833195
26674 # UVM_INFO [monitor_axi_write_interface] @ 39530.0ns: => *** curr_write_addr ==== f3833195
26675 # UVM_INFO [monitor_axi_write_interface] @ 39530.0ns: => *** curr_write_addr ==== f3833195
26676 # UVM_INFO [monitor_axi_write_interface] @ 39530.0ns: => *** curr_write_addr ==== f3833195
26677 # UVM_INFO [monitor_axi_write_interface] @ 39530.0ns: => *** curr_write_addr ==== f3833195
26678 # UVM_INFO [monitor_axi_write_interface] @ 39530.0ns: => *** curr_write_addr ==== f3833195
26679 # UVM_INFO [monitor_axi_write_interface] @ 39530.0ns: => *** curr_write_addr ==== f3833195
26680 # UVM_INFO [monitor_axi_write_interface] @ 39530.0ns: => *** curr_write_addr ==== f3833195
26681 # UVM_INFO [monitor_axi_write_interface] @ 39530.0ns: => *** curr_write_addr ==== f3833195
26682 # UVM_INFO [monitor_axi_write_interface] @ 39530.0ns: => *** curr_write_addr ==== f3833195
26683 # UVM_INFO [monitor_axi_write_interface] @ 39530.0ns: => *** curr_write_addr ==== f3833195
26684 # UVM_INFO [monitor_axi_write_interface] @ 39530.0ns: => *** curr_write_addr ==== f3833195
26685 # UVM_INFO [monitor_axi_write_interface] @ 39530.0ns: => *** curr_write_addr ==== f3833195
26686 # UVM_INFO [monitor_axi_write_interface] @ 39530.0ns: => *** curr_write_addr ==== f3833195
26687 # UVM_INFO [monitor_axi_write_interface] @ 39530.0ns: => *** curr_write_addr ==== f3833195
26688 # UVM_INFO [write_sys_axi_r_ap] @ 39640.0ns: => Entered in write_sys_axi_r_ap
26689 # UVM_INFO [DEBUG] @ 39640.0ns: => SYS_AXI_READ_RESPONSE received:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Size</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>rd_trans</td>
<td>axi_tnx</td>
<td>-</td>
<td>@39302</td>
</tr>
<tr>
<td>id</td>
<td>integral</td>
<td>15</td>
<td>'h0</td>
</tr>
<tr>
<td>direction</td>
<td>direction t</td>
<td>32</td>
<td>READ</td>
</tr>
</tbody>
</table>

© Accellera Systems Initiative
Log Files

● A log file may contain all the information that is required for understanding what happened in the test.

● But,
  1. It includes too much information.
  2. It is hard to search through.
  3. It is hard to remember.

● Log files are not ideal for humans to comprehend.
The London Tube - 19th Century
The London Tube Map - 1926
The London Tube Map - 1931
Revolutionizing How We Perceive a Log File

● We need to find a way to visualize test result, without the clutter of “geography”.
● We must maintain the details around a bug scenario, but without “compromising the suburbs”.
● Only one data path, one class, should be highlighted.
● Fisheye lens.
Cogita

- Cogita automatically parses huge simulation log files.
- Cogita visualizes them in one clear navigational image.
- Cogita offers innovative tools that drastically reduce debug time.
- Cogita is content-agnostic, and able to read any log file in any verification environment, not limited to UVM methodology, SystemVerilog, or others.
- Not limited to verification at all.
Before and After
How Cogita Works

- Smart parser reads any log format.
- Messages from the log are presented on a timeline (much like a wave viewer).
- The query builder helps to search, and presents only the relevant info at any given moment.
- Visual comparisons enable to spot scenarios in an instance.
- We appeal to the natural human capacity for processing visual images.
Example - Memory Controller with Multiple AXI Ports
1. Entire Log Scenario with Error Highlighted
2. Looking for “BLK0_BNK3” Events
Test Case
Endless Analysis Potential

- Cogita combines a multidimensional database, into one clear image.
- Log messages, errors, waves, simulation results.
- Cogita serves as a platform for root-cause-analysis algorithms, presented in a way that humans can quickly comprehend.
Other Analysis Capabilities
Cogita Advantages

- Drastically reduces detection time of complex bug scenarios.
- Tweaking test scenarios, to reach an expected outcome.
- Better understanding of teamwork scenarios.
Summary

• We appeal to the natural human capacity for processing images, not text.
• The log exists as a dia-log, instead of the addiction to logic rhythm (a-log-arithm).
• Cogita is the new way to debug.
Thank You
Questions?