Taking Design Automation to the next level with User Experience Design

Jamie Lai, Bodo Hoppe
IBM R&D Germany GmbH
The Team

Jamie Lai
UX designer, IBM Z Systems
jamie.lai@ibm.com

Bodo Hoppe
Distinguished engineer, IBM Z Hardware
bohopp@de.ibm.com
The IBM Telum Processor

> 5 GHz frequency

> 22 Billion Transistors on a module

> 19 Miles of wires on a module
Development of the IBM Telum Processor

1,500,000,000,000,000 Simulation Cycles in software Simulation

> 500K unique discrete coverage events in the design

+ Millions of cross-product events
Logic Designers and Verification Engineers spend a huge amount of time on

- Defining coverage
- Implementing coverage
- Debugging coverage
- Analyzing coverage holes
- Hitting important coverage events
Fast time to market

Concept
Specification
Logic Design
Block Simulation
Processor Core Simulation
Physical Design
System Simulation

Tape-Out
Power-On

Concept
Specification
Logic Design
Block Simulation
Processor Core Simulation
Physical Design
System Simulation

Tape-Out
Power-On
Potential of Functional Coverage

- Faster time-to-market
- Higher Quality
- Less (Compute) Cost
We use various functional coverage types

- Interface coverage
- (Micro-)Architectural cross product coverage
- Configuration coverage
- Discrete design events
- Testbench coverage
We have (some) collaboration

- Mark as waived/deferred/impossible
- Analyze
- Prioritize
- Comment
We have the technology

- Template Aware Coverage
- Coverage Driven Generation
- Finding unhittable events
- Coverage hole analysis
- Grading events & test templates
- Identifying aged out events

But...

- Many months are still spent on coverage closure
  - In the critical path for tape-out
- We’re still not seeing the forest because of all the trees
  - Missing potential bugs
- Still wasting compute cycles
Let’s automate more ...

... and we can also ...
The User

Collaboration

Functional Coverage Types

Automation
Our tech and our tools can **assist**, but they cannot **replace**
How do we use the tools and tech we have in a way that will **enhance and augment** the user’s abilities?
Introducing... UX design!
“Design a vase”
“Design a better way for someone to enjoy flowers”
We look to these ways of questioning our problem spaces, to **open up our creativity** and scope, and to ask the **Right Questions**
Enterprise design thinking: The loop

Observe  Reflect  Make
UX design: An umbrella term

Observe

Reflect

Make

UX research
UX design: An umbrella term

Observe  Reflect  Make

UX design  Systems design  Service design  Product design
UX design: An umbrella term

Observe  Reflect  Make

UI design
Visual design
Content design
The original scope

- Improve the coverage data analysis experience
- Show user actions like waiving, deferring, marking impossible...
"What's the point?"

"This is just a facelift"

"We already have something similar"
What we were doing: treating the symptom of functional coverage
A quick reminder...

https://scrolller.com/ui-vs-ux-3ly0tb43v4
What we should’ve been doing: investigating the **underlying issue**...
Why are there so many irrelevant and confusing events in the first place?
“Design a way to improve the coverage data analysis experience”
“Design a way to improve the coverage data analysis experience”
“Design a way to close the design of the microprocessor faster”
Feature definition
Feature implementation
Coverage analysis
Coverage closure
Floral fantasy at Gardens by the Bay, Singapore
by: @unique_singapore [IG]
Observe

Reflect

Make
User research questions

• What is the **overall process**, from feature definition to coverage closure?

• Where are the **collaboration touchpoints** between VE and LD?

• What are the **pain points** in this collaboration process?
Design toolbox: Personas

Eddie
Logic Designer

Works with architects to define the new features of the chip
Creates the logic design of the hardware
Defines spaces that need verification, using things called “events”

Jobs to be done?
Collaboration?
Pain points?

Vi
Verification Engineer

Creates test cases in C++ that stress the design of the chip
Looking to improve their test bench to hit all important coverage events to find all bugs in the HW
Sorts through all the events generated by the Verification Cockpit (VC) to look for the most interesting events
Research

Step 1: Observe

What does the user think/do?

Step 2: Gather insights

What common attitudes and behaviors do our users collectively have?

Step 3: Synthesize into big ideas

What are the main ideas we can gather from this?
EDT toolkit: As-is scenario
## Designer toolkit: As-is scenario overview

<table>
<thead>
<tr>
<th>Phase 1 Feature definition</th>
<th>Phase 2 Feature implementation Simulation bring-up</th>
<th>Phase 3 Coverage analysis</th>
<th>Phase 4 Coverage closure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main feature and design are defined here</td>
<td>Designs are created, simulation testing begins</td>
<td>Analysis of the testbench and discrete events begins</td>
<td>Coverage must meet the expected quality requirements</td>
</tr>
<tr>
<td>Eddie draws up the specs and designs with the architects</td>
<td>Eddie begins coding his designs</td>
<td>Eddie and Vi collaborate to analyze the results.</td>
<td>Vi writes special cases and continues to improve the testbench until the coverage percentage is high enough</td>
</tr>
<tr>
<td>Vi takes a back seat</td>
<td>Vi looks at the design specs in order to begin crafting her testbench scenarios</td>
<td></td>
<td>Eddie improves his designs</td>
</tr>
</tbody>
</table>
**Designer toolkit: As-is scenario overview**

**Phase 1: Feature definition**
- Main feature and design are defined here
- Eddie draws up the specs and designs with the architects

**Phase 2: Feature implementation**
- Design is created, simulation testing begins
- Eddie begins coding his designs
- Vi looks at the design specs in order to start crafting her testbench scenarios

**Phase 3: Coverage analysis**
- Analysis of the testbench and discrete events begins
- Eddie and Vi collaborate to analyze the results
- Analysis of the testbench and discrete events begins

**Phase 4: Coverage closure**
- Coverage must meet the expected quality requirements
- Vi writes special cases and continues to improve the testbench until the coverage percentage is high enough
- Eddie improves his designs

---

Eddie is often asked to deliver both good design and good coverage.

Vi does not participate in design scoping.

Because Eddie must focus on his designs, this often results in the creation of improperly documented coverage.

Vi is blocked from coverage until after the implementation of the designs by Eddie.

Vi, in turn, struggles with understanding the intent of events coded by Eddie.

There is no central repository for proper collaboration.

Vi has no set definition of “done.”

Eddie discovers bugs super late in the implementation phase because they were not caught by the testbench earlier.
Designer toolkit: As-is scenario overview

**Phase 1**
Feature definition
- Main feature and design are defined here
- Eddie draws up the specs and designs with the architects
- Vi takes a back seat
- Designs are created, simulation testing begins
- Eddie begins coding his designs
- Vi looks at the design specs in order to begin crafting her testbench scenarios

**Phase 2**
Feature implementation Simulation bring-up
- Analysis of the testbench and discrete events begins
- Eddie and Vi collaborate to analyze the results.
- Coverage must meet the expected quality requirements
- Vi writes special cases and continues to improve the testbench until the coverage percentage is high enough
- Eddie improves his designs

**Phase 3**
Coverage analysis

**Phase 4**
Coverage closure
- Coverage must meet the expected quality requirements
- Vi has no set definition of “done”
- Eddie discovers bugs super late in the implementation phase because they were not caught by the testbench earlier
- Vi struggles with understanding the intent of events coded by Eddie

Eddie is often asked to deliver both good design and good coverage.

Vi does not participate in design scoping.

Because Eddie must focus on his designs, this often results in the creation of improperly documented coverage.

Vi is blocked from coverage until after the implementation of the designs by Eddie.

There is no central repository for proper collaboration.
EDT toolkit: To-be scenario
How might we activate the users through collaboration?

Pain points

- Eddie often asked to deliver both good design and good coverage
- Vi does not participate in design scoping

To-be scenario

- Eddie defines the events without implementation, along with the scope of the stories
- Vi collaborates earlier to understand the scope and maps the events to the features created
How might we **activate** the users through collaboration?

**Pain points**

• Because Eddie must focus on his designs, this often results in the creation of improperly documented coverage

• Vi struggles to understand the coverage created by Eddie

**To-be scenario**

• Eddie **provides the details** of the coverage space and the expected results to Vi

• Vi **implements the coverage** defined by Eddie in her testbench in order to create thorough and well-architected coverage
How might we activate the users through collaboration?

**Pain points**

- There is no central repository for proper collaboration

**To-be scenario**

- All past tooling and features are integrated into 1 collaborative-centric platform
- The experience and the views on this platform are standardized for easier access by both personas
Design a prototype

Observe

Reflect

Make
Overview

To-be scenario: All past tooling and features are integrated into 1 collaborative-centric platform

- Implemented with a standardized design system
- The user can see important metadata in order to provide the necessary context of the event
- Advanced filtering
Coverage details

To-be scenario: Eddie provides the details of the coverage space and the expected results to Vi

• A new “more info” page gives more detailed info about the coverage point
• This is where Vi would find information about coverage, fleshed out by Eddie
• History/log of events and comments can be found here
Request Coverage

To-be scenario: Eddie defines the events without implementation, along with the scope of the stories

• Eddie is able to request coverage and give a detailed blueprint of how it should be implemented
• Vi will take this blueprint to implement coverage into her testbench
Enabling Collaboration using Python

- Python is a common language
- Open Source Development
- System Verilog Constructs well proven over years

```python
@vsc.covergroup
class my_weak_pht_cnt_cg(object):
    def __init__(self):
        self.with_sample(
            a = uint8_t(5)
            b = uint8_t(5)
        )
        self.cp1 = vsc.coverpoint(self.a, bins=dict(
            A = vsc.bin_array([],1,2,3,4)),
            illegal_bins=dict(
                illegal_Val = vsc.bin(5)
            )
        )
        self.cp2 = vsc.coverpoint(self.b, bins=dict(
            A = vsc.bin_array([],[5,31])
        )
    self.cp1x2 = vsc.cross([self.cp1, self.cp2])

Credits to Matt Balance’ PyVSC library:
```
Implement a e2e prototype
Develop an end-2-end prototype in 2 weeks

Define
Implement
Collect
Analyze
Our client – starts tests, submits hit data, submits event info on trigger

Calc1 example

Simulator (MESA)
Summary

Enterprise Design Thinking is key

✓ With the user in the loop

UX design is key to

✓ Deeply understand the developers‘ needs
✓ taking a holistic view at the process
✓ fast prototyping
✓ Drive usability testing
Conclusions

- UX design and Prototyping allow fast iterations
- Can change the way how we work! better better together
- Allows the developer to take full advantage of the available technology
- Can significantly improve the overall development effort and schedule
Questions?

Jamie Lai
UX designer, IBM Z Systems
jamie.lai@ibm.com

Bodo Hoppe
Distinguished engineer, IBM Z Hardware
bohopp@de.ibm.com