Microprocessor Design Bootcamp: Verification 101

December 19, 2025

Why Verification is Important?

Why Verification is Important?

Logarithmic Cost of Defect Escapes

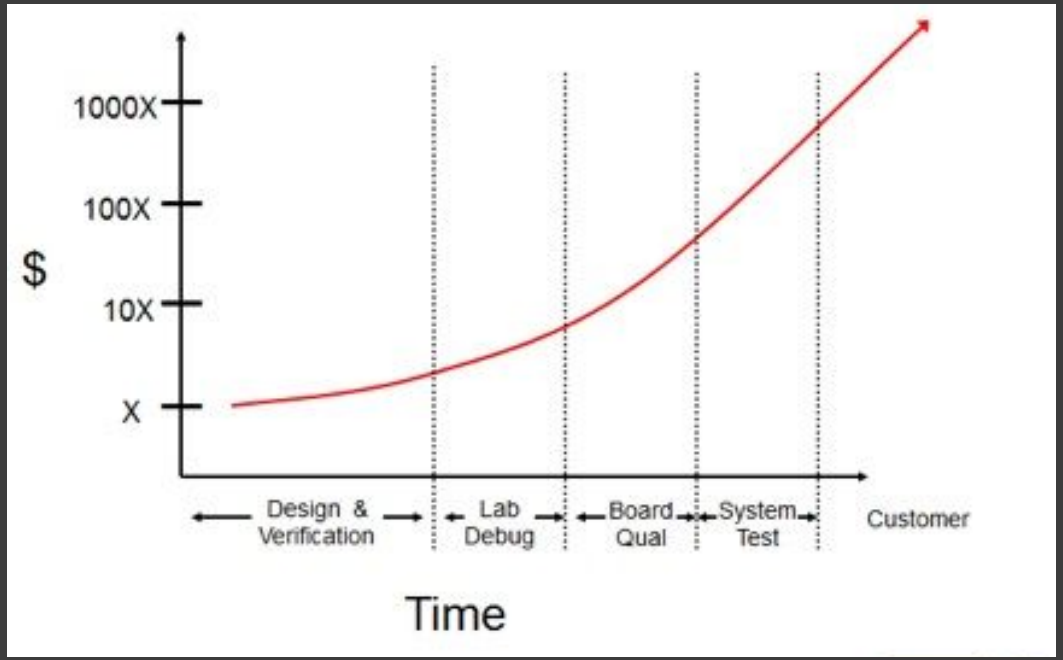


Image source: https://www.intrinsix.com/blog/bug-tracking-for-soc-asic-design-verification-projects

Why Verification is Important?

Logarithmic Cost of Defect Escapes

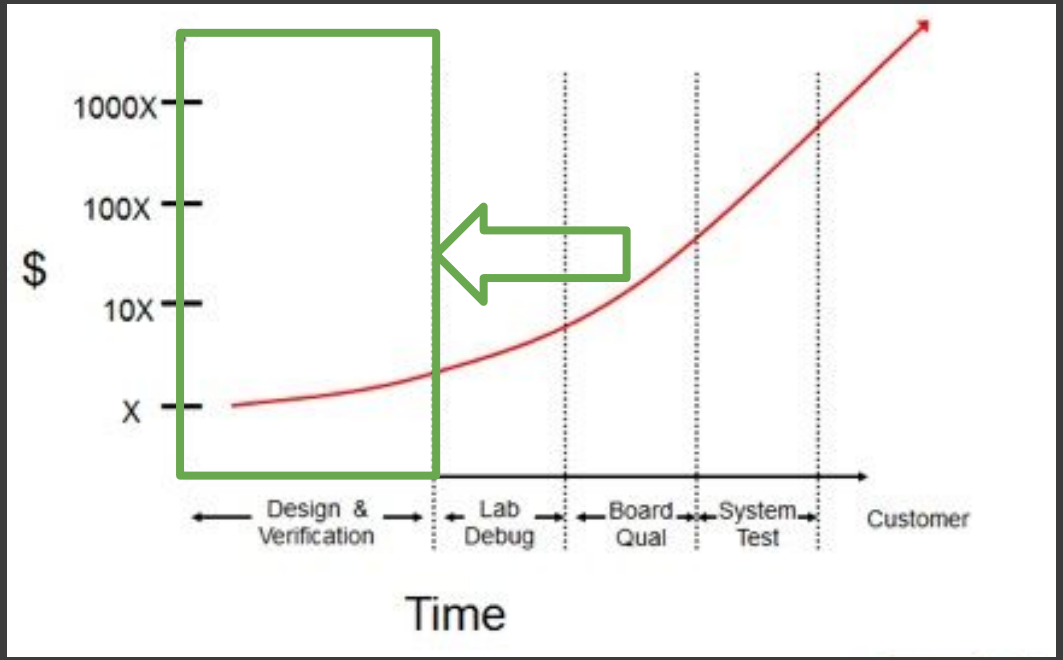


Image source: https://www.intrinsix.com/blog/bug-tracking-for-soc-asic-design-verification-projects

Why Verification is Important?

A bug costed Samsung $17 billion

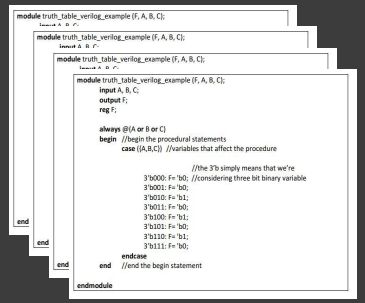
https://www.synopsys.com/blogs/software-security/cost-to-fix-bugs-during-each-sdlc-phase/

A bug can cost lives

How microprocessors are verified?

How microprocessors are verified?

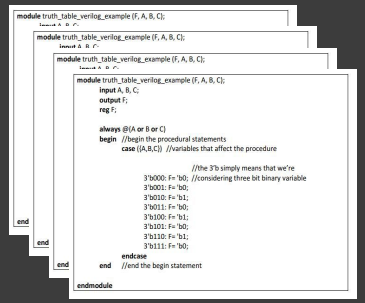
RTL (Verilog/VHDL/...)



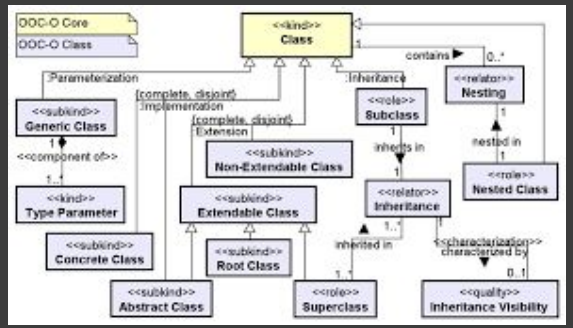
```
module truth_table_verilog_example (F, A, B, C);
    input A, B, C;
    output F;
    reg F;

    always @(A or B or C)
    begin   //begin the procedural statements
        case ({A,B,C})  //variables that affect the procedure

                                //the 3'b simply means that we're
            3'b000: F= 'b0;  //considering three bit binary variable
            3'b001: F= 'b0;
            3'b010: F= 'b1;
            3'b011: F= 'b0;
            3'b100: F= 'b1;
            3'b101: F= 'b0;
            3'b110: F= 'b1;
            3'b111: F= 'b0;
        endcase
    end   //end the begin statement

endmodule
```

How microprocessors are verified?

RTL (Verilog/VHDL/...)

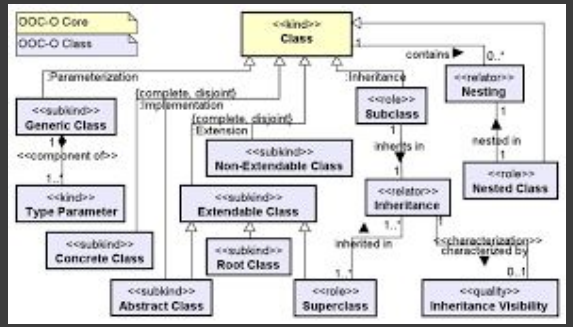


```
module truth_table_verilog_example (F, A, B, C);
    input A, B, C;
    output F;
    reg F;

    always @(A or B or C)
    begin   //begin the procedural statements
            case ({A,B,C})  //variables that affect the procedure

                                    //the 3'b simply means that we're
                    3'b000: F= 'b0;  //considering three bit binary variable
                    3'b001: F= 'b0;
                    3'b010: F= 'b1;
                    3'b011: F= 'b0;
                    3'b100: F= 'b1;
                    3'b101: F= 'b0;
                    3'b110: F= 'b1;
                    3'b111: F= 'b0;
            endcase
    end     //end the begin statement

endmodule
```



How microprocessors are verified?

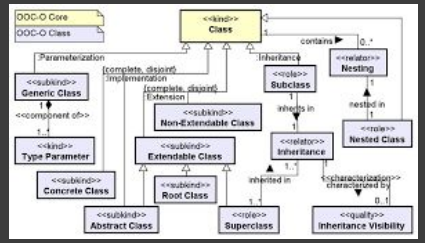
Design Under Test (DUT)



How microprocessors are verified?

Testbench

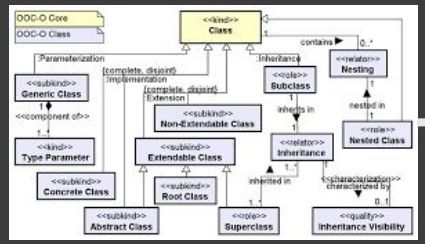
DUT



How microprocessors are verified?

Testbench

DUT

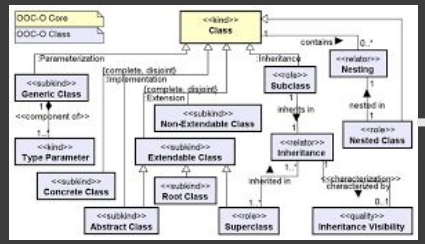


Memory model

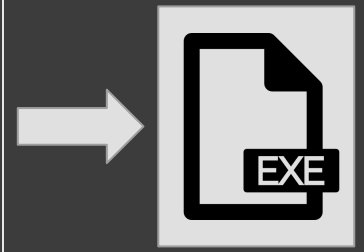
How microprocessors are verified?

Testbench

DUT



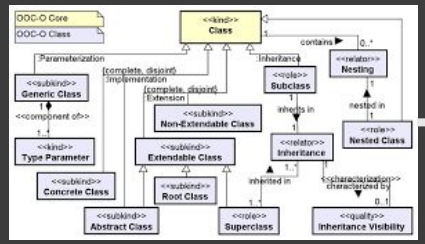
Memory model



How microprocessors are verified?

Testbench

DUT



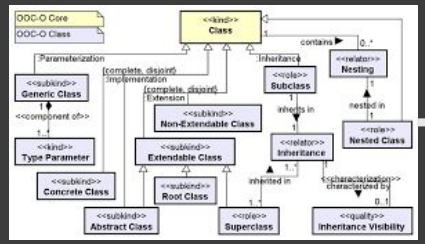
Memory model

Which tests to run?

How microprocessors are verified?

Testbench

DUT



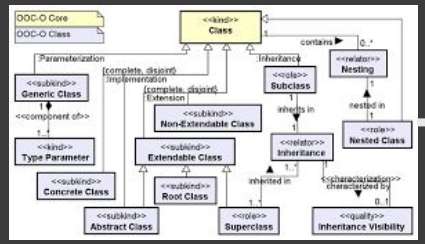
Memory model

Unit Tests

How microprocessors are verified?

Testbench

DUT



Memory model

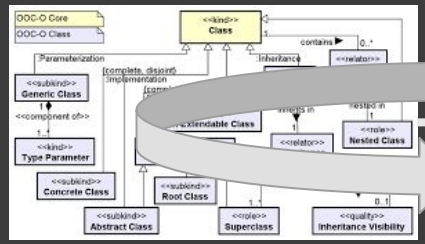
3x3

3x3 = 9

How microprocessors are verified?

Testbench

DUT

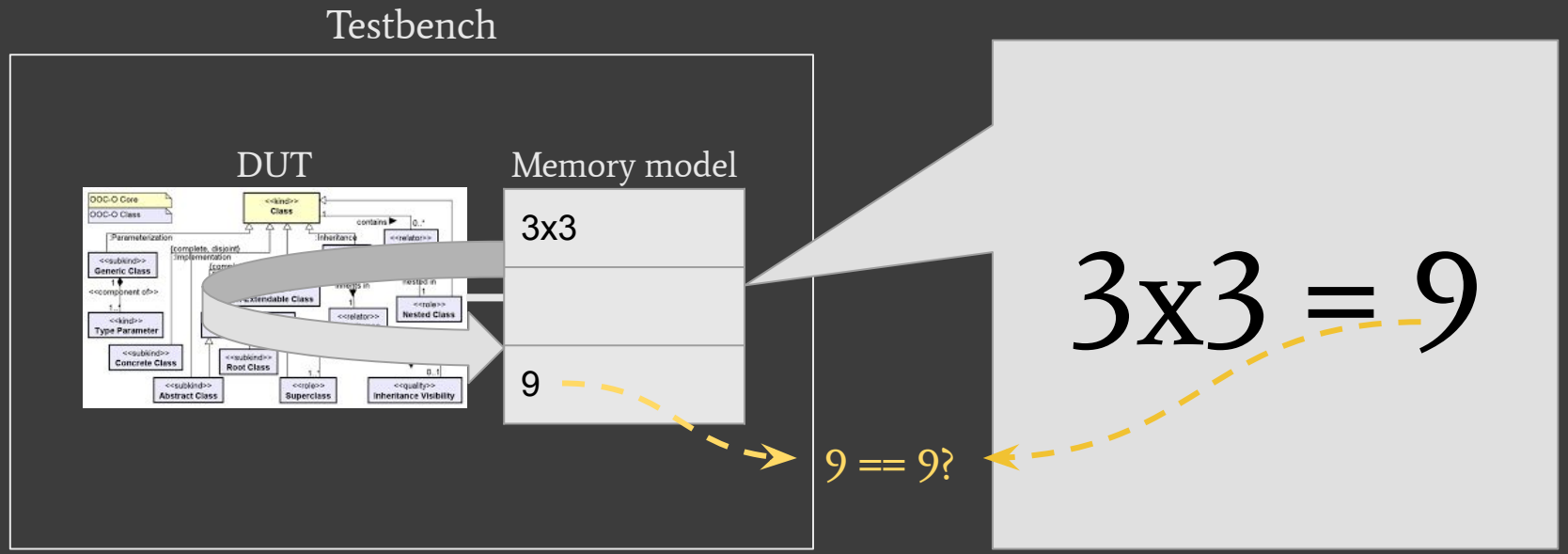


Memory model

3x3
9

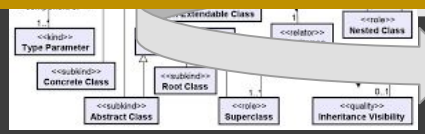
3x3 = 9

How microprocessors are verified?



How microprocessors are verified?

Testbench



9

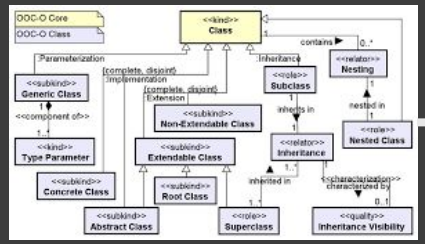
Unit tests are not enough

3x3 = 9

How microprocessors are verified?

Testbench

DUT



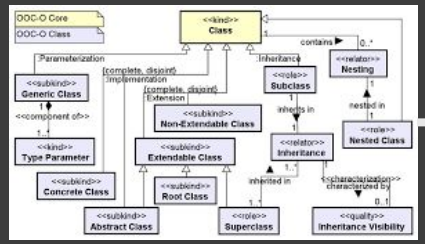
Memory model

Real apps, boot Linux

How microprocessors are verified?

Testbench

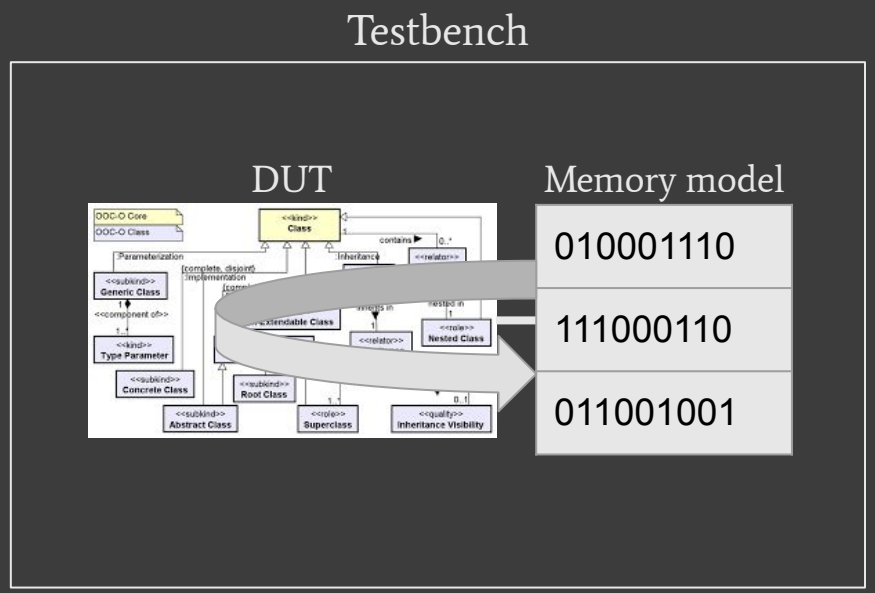
DUT



Memory model

Randomly Generated Test

How microprocessors are verified?



How do we know if the test passed or failed?

Comparison with the “golden” model

How microprocessors are verified?

The golden model:

- very-high level abstraction of the processor

How microprocessors are verified?

The golden model:

- very-high level abstraction of the processor
- is independent software

How microprocessors are verified?

The golden model:

- very-high level abstraction of the processor
- is independent software
- is simple, has no implementation details

How microprocessors are verified?

The golden model:

- very-high level abstraction of the processor
- is independent software
- is simple, has no implementation details
- changes architectural state on the instruction level granularity

How microprocessors are verified?

The comparison with the golden model can be implemented:

1) by end-of-simulation architectural state comparison

How microprocessors are verified?

The comparison with the golden model can be implemented:

1) by end-of-simulation architectural state comparison
2) by trace comparison

How microprocessors are verified?

The comparison with the golden model can be implemented:

1) by end-of-simulation architectural state comparison
2) by trace comparison
3) by co-simulation

How microprocessors are verified?

Testbench with
DUT and memory

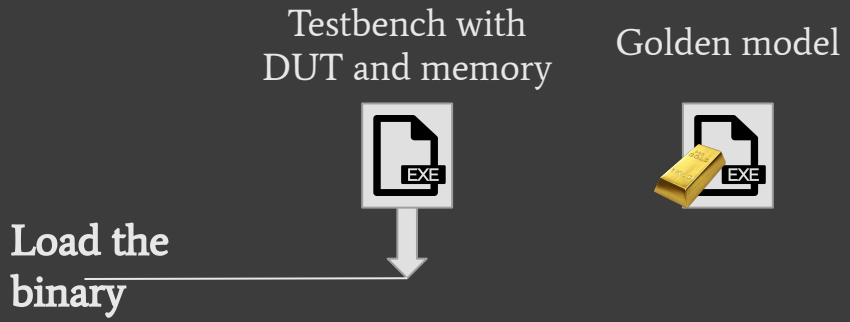
Golden model

How microprocessors are verified?

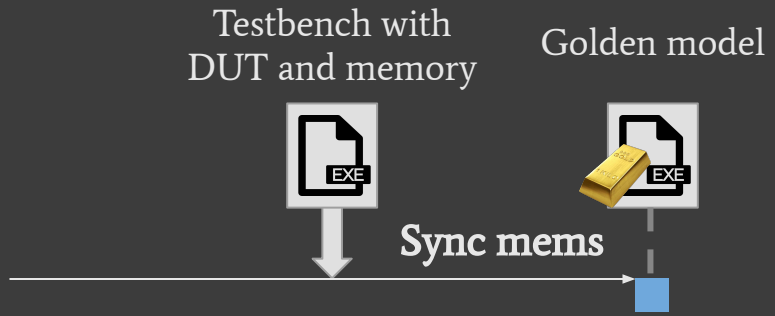
Testbench with
DUT and memory

Golden model

How microprocessors are verified?



How microprocessors are verified?

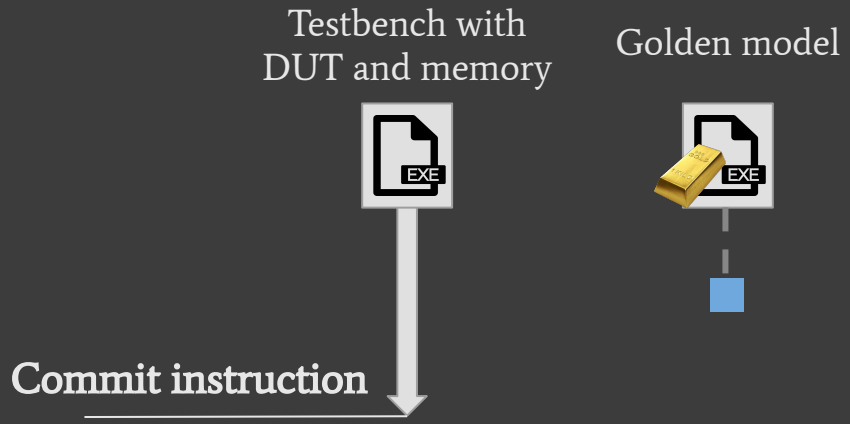


How microprocessors are verified?

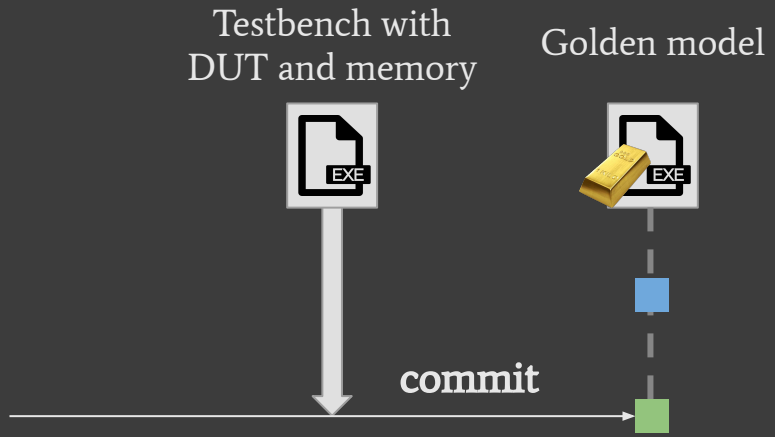
Testbench with
DUT and memory

Golden model

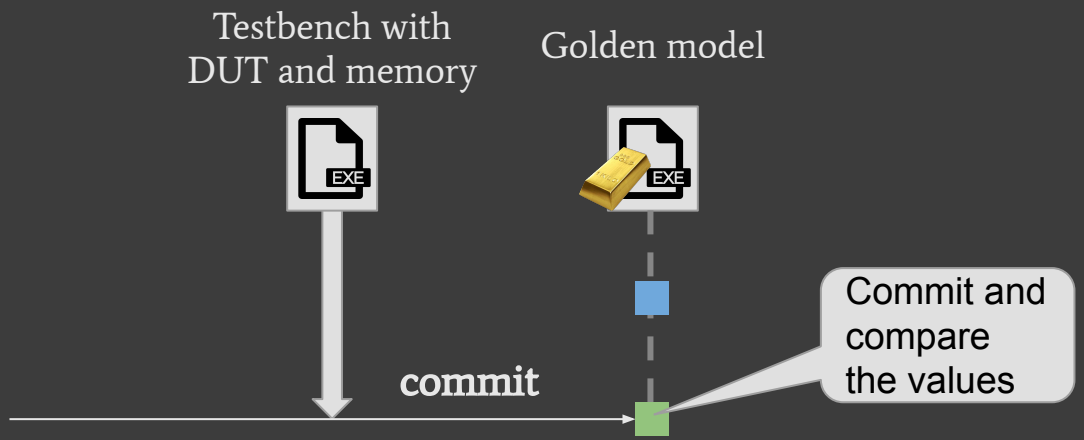
How microprocessors are verified?



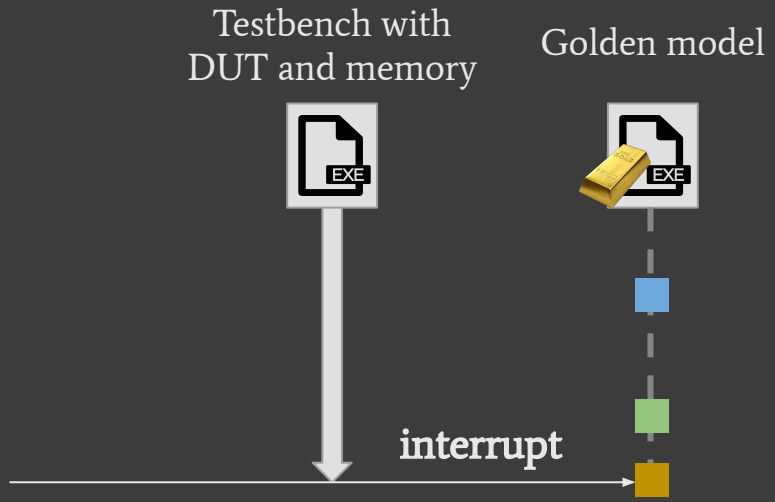
How microprocessors are verified?



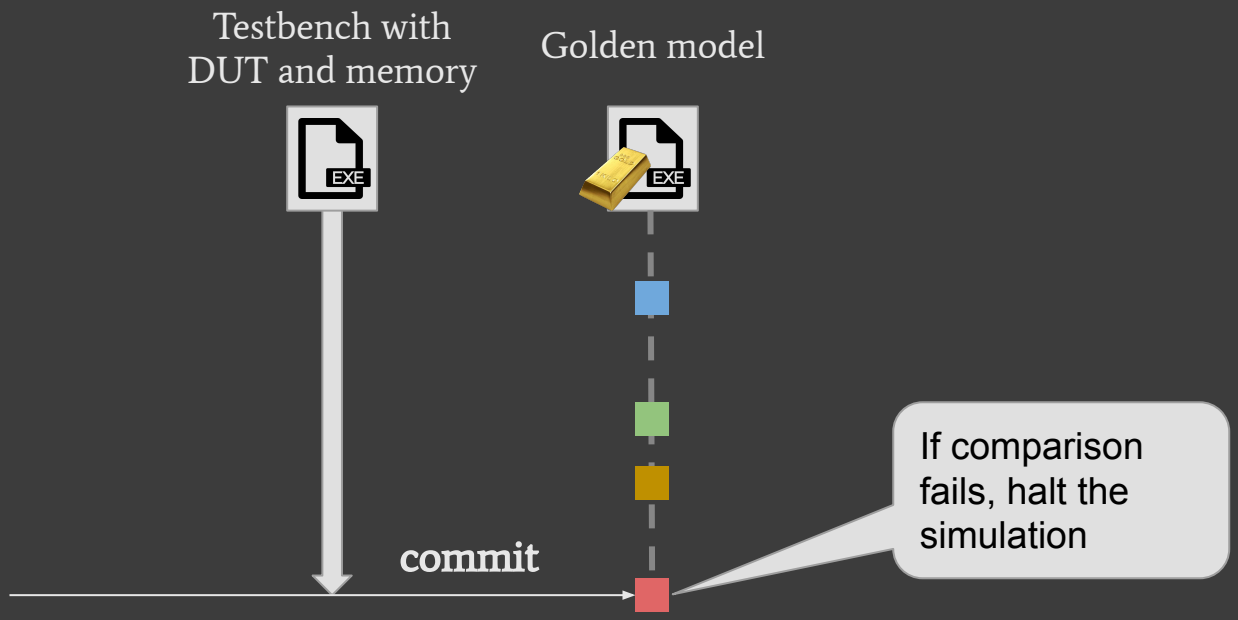
How microprocessors are verified?



How microprocessors are verified?



How microprocessors are verified?



How much to simulate?

How microprocessors are verified?

To measure quality of the simulation, we use these metrics:

- code coverage metrics

How microprocessors are verified?

To measure quality of the simulation, we use these metrics:

- code coverage metrics
- circuit specific coverage metrics

How microprocessors are verified?

To measure quality of the simulation, we use these metrics:

- code coverage metrics
- circuit specific coverage metrics
- **functional coverage metrics**

How microprocessors are verified?

To measure quality of the simulation, we use these metrics:

- code coverage metrics
- functional coverage metrics
- circuit specific coverage metrics
- time since the last bug was found

How microprocessors are verified?

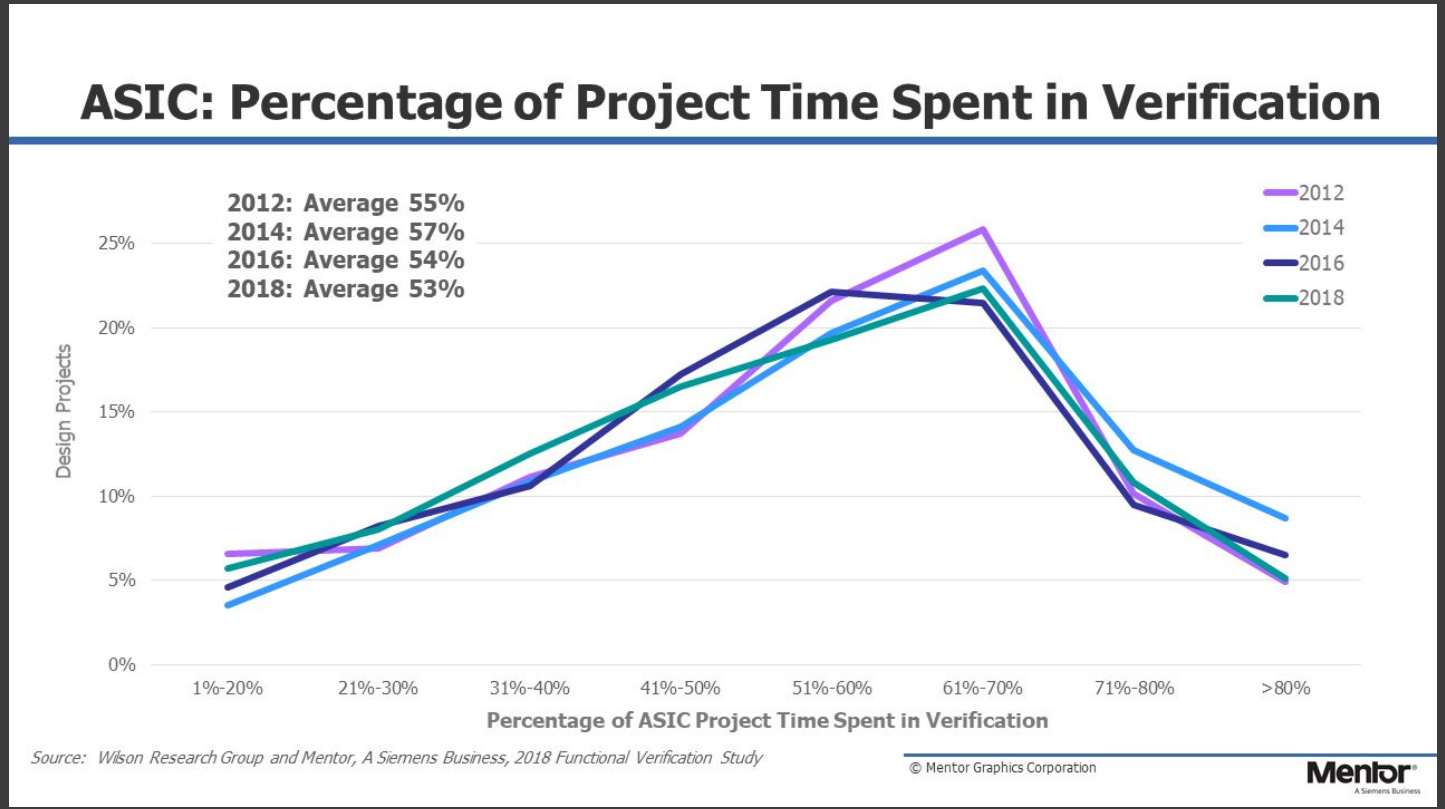
To measure quality of the simulation, we use these metrics:

- code coverage metrics
- functional coverage metrics
- circuit specific coverage metrics
- time since the last bug was found

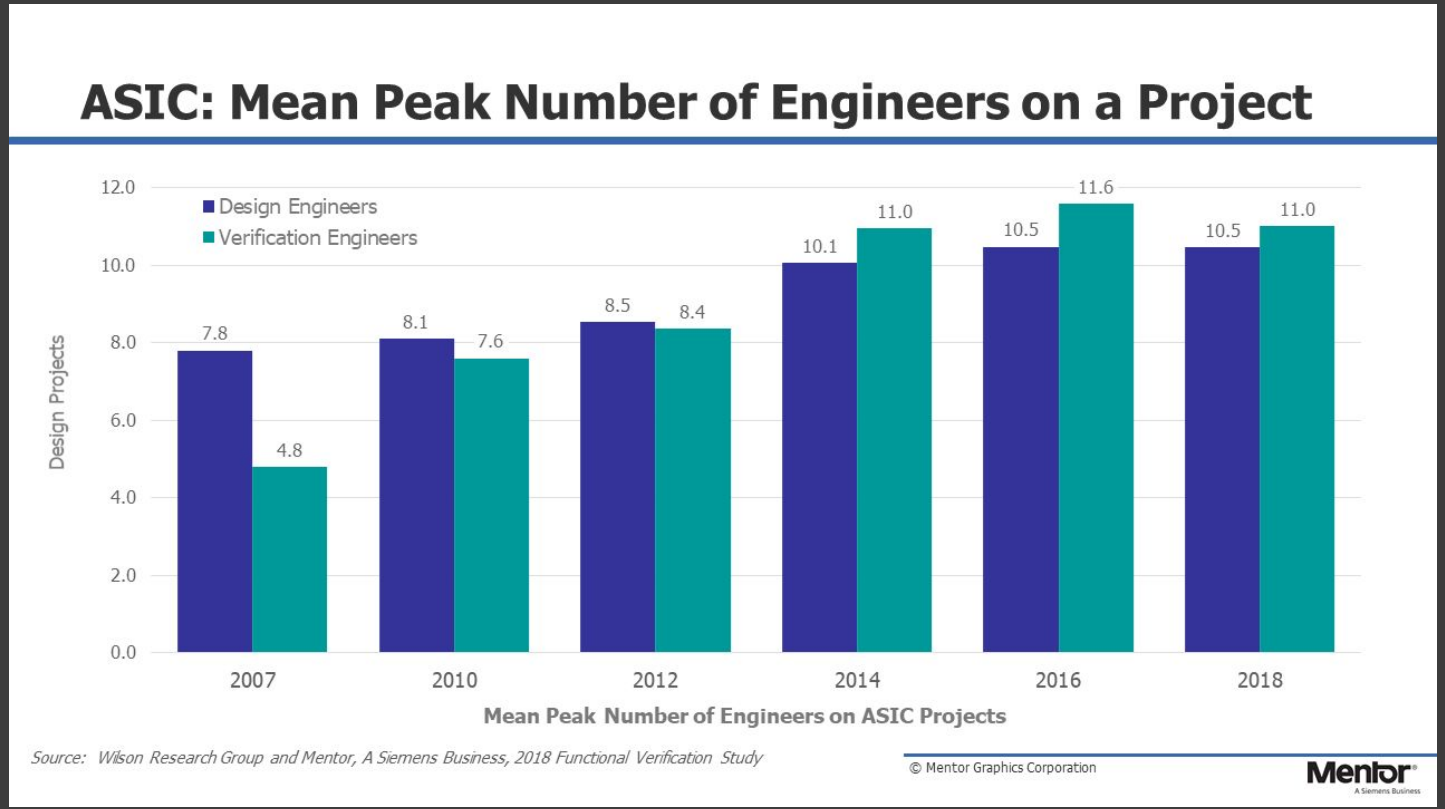
- All of the above are proxy metrics

Verification takes long time and enormous engineering effort

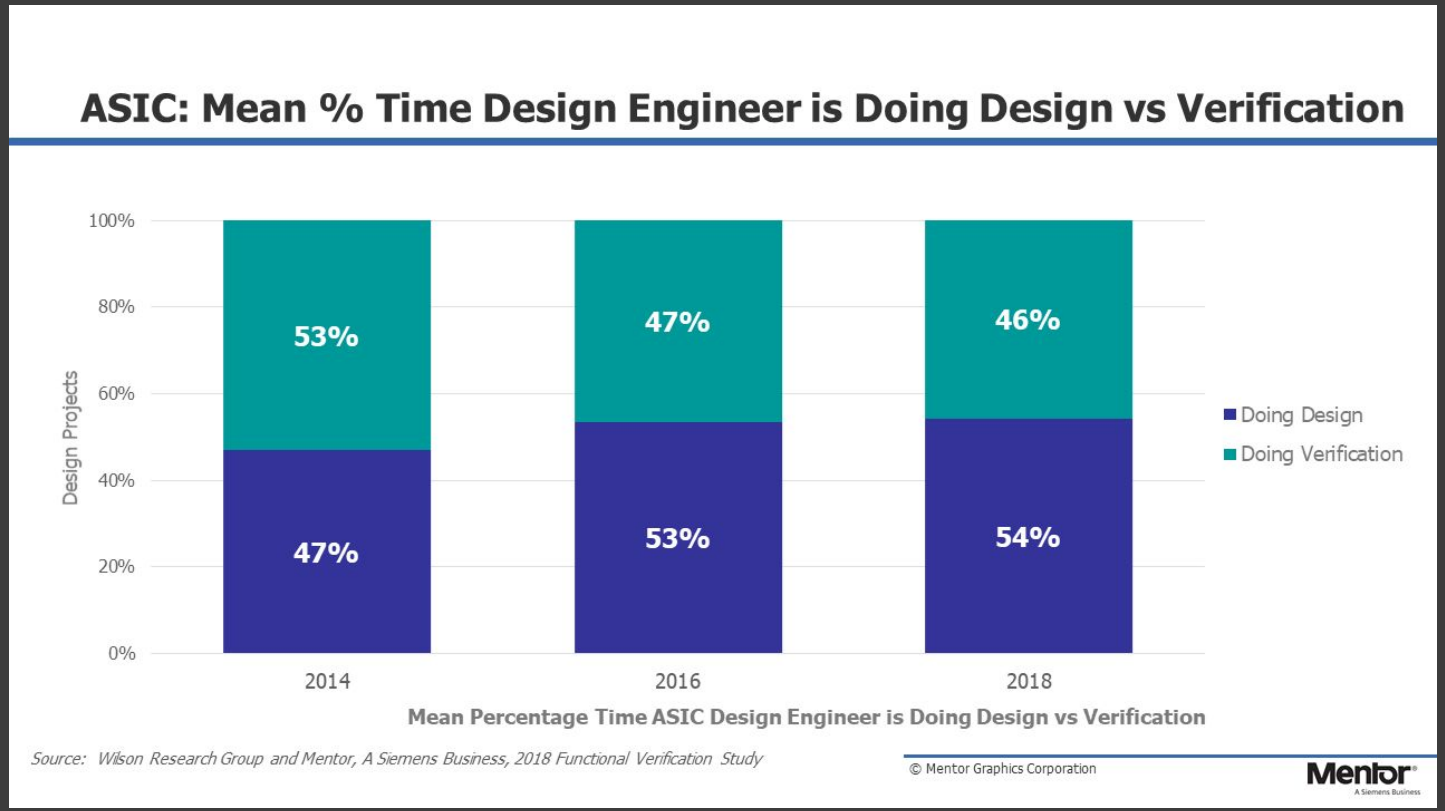
Problem statement



Problem statement

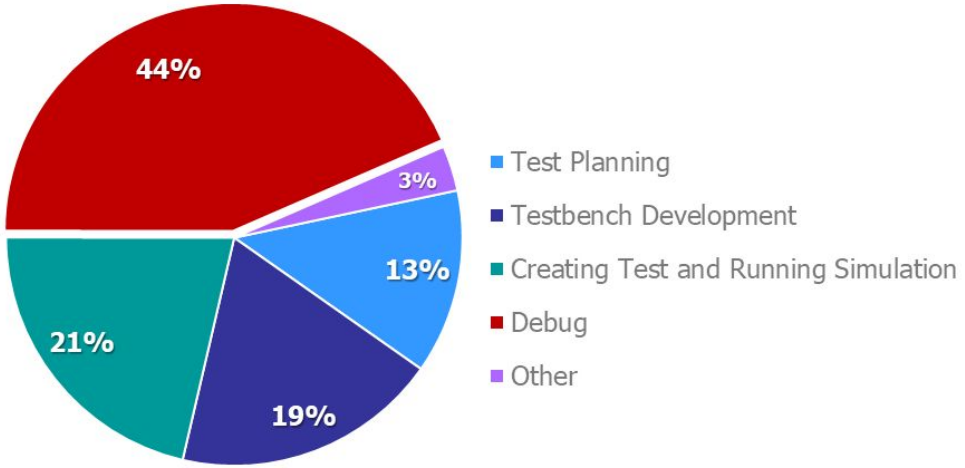


Problem statement



Problem statement

ASIC: Where Verification Engineers Spend Their Time

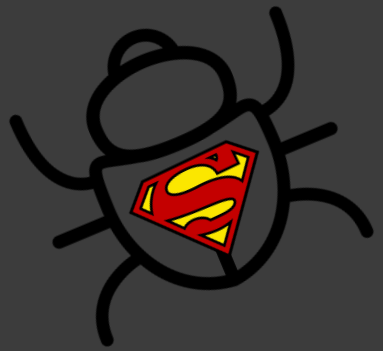


Source: Wilson Research Group and Mentor, A Siemens Business, 2018 Functional Verification Study

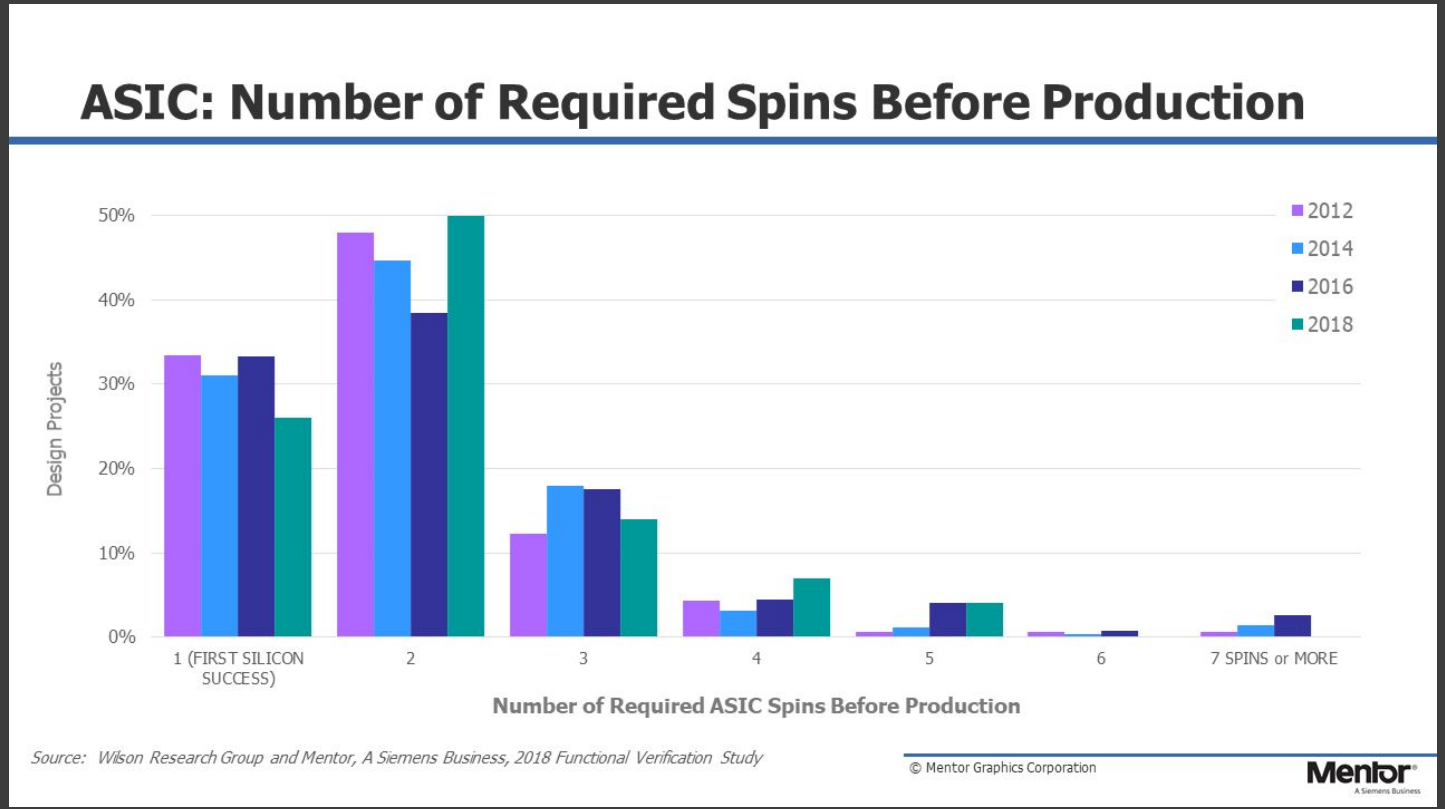
Mentor
A Siemens Business

Despite so much time and effort, bugs sneak into silicon

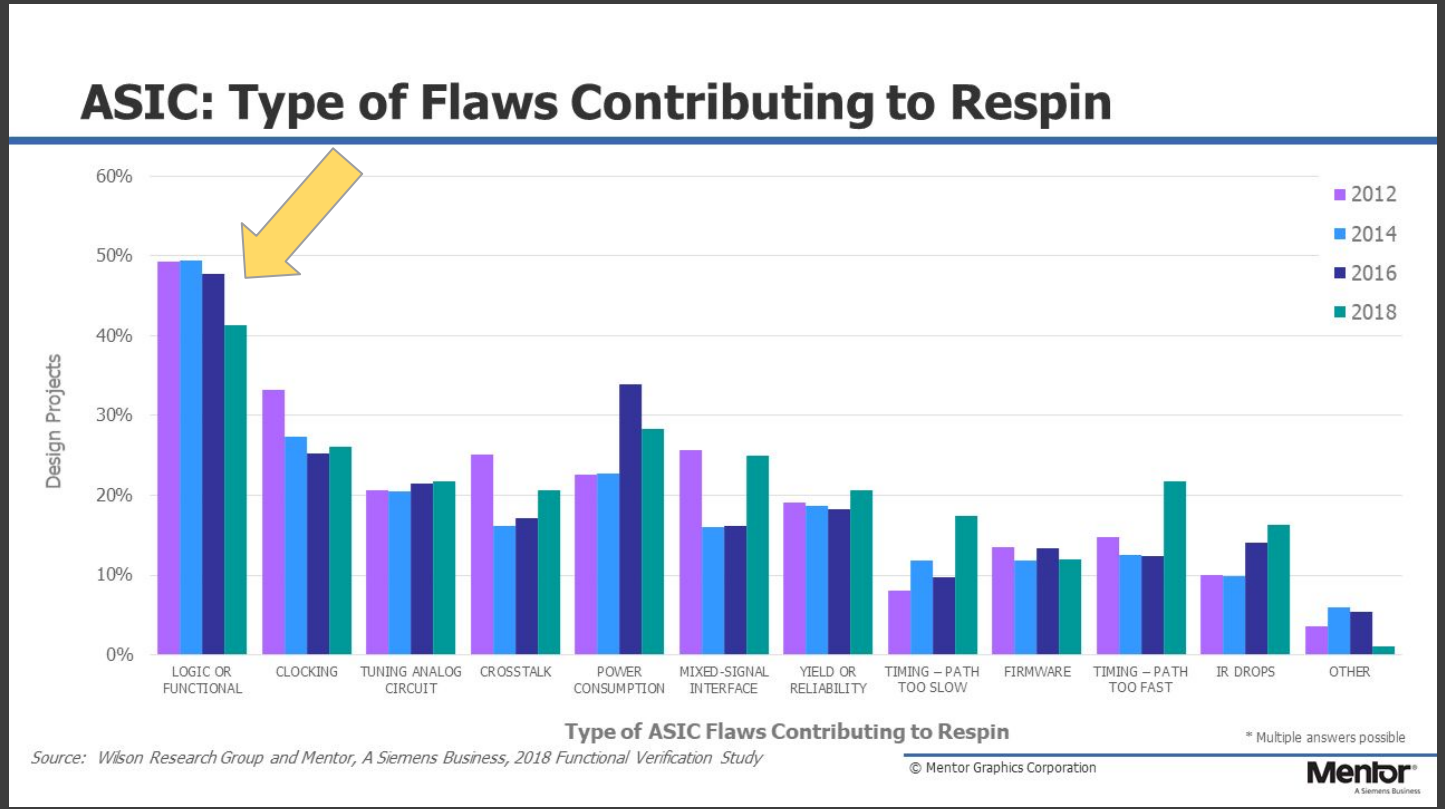
Simulation resistant “superbugs”



Problem statement



Problem statement



Problem statement

To sum up the challenges:

1) verification is time consuming
2) there are bugs that are not detected by simulation

Now let's build the golden model for sCPU!

sCPU Verilog RTL

sCPU golden model