EmLogic

The lack of an important verification knowledge

Verification Futures Conference
8 June 2022

(by Espen Tallaksen, CEO EmLogic)

Also a lack of Design knowledge

The title
'The lack of an important **Verification** knowledge'

is even more important if there is a corresponding
'Lack of the corresponding **Design** knowledge'

*This is indeed - A Design **AND** Verification challenge*

Cycle related corner cases

Cycle related corner cases

Occur when the cycle in which an event happens is not fixed – and this event is related to another event – fixed or not

Typically - events in different FSMs (explicit or implicit) that affect each other or affect the same objects

- Any two or more events combining into an FSM
  - E.g. Interrupt or DMA controller receiving two triggers 0-N cycles apart
- **Any** communication interface:
  - e.g. Read-out vs New data entry
- Lots of other scenarios
  - And very often scenarios not seen by the designer as a corner case…

Communication interface example (1)

```
if (rx_data_reg read from cpu) then
   <something>
elsif (new byte received from rx) then
   <something>
end if;
```

If-then-else
for non-exclusive actions
e.g. inside single process in UART:

Bug if both are true in the same cycle

Different coding style
- Same scenario:

```
if (rx_data_reg read from cpu) then
   <something>
end if;
.....
.....
if (new byte received from rx) then
   <something>
end if;
```

Bug if <something> is

a) go to new state, or

b) assigning to same object

if both ifs are true in the same cycle

Communication interface example (2)

```
if (rx_data_reg read from cpu) then
   <something>
elsif (new byte received from rx) then
   <something>
end if;
```

If-then-else
for non-exclusive actions
e.g. inside single process in UART:

For clock @ 100 MHz and bit rate @ 100 kHz (→ byte @ 10 kHz)

- Probability of bug = 1:10.000 per byte
- Probability of detection in sequential simulation:
  - For one single byte: 1:10.000
  - For thousands of bytes: 1:10.000 – in lots of testbenches
    > 1:10.000 – in quite a few testbenches but not necessarily much better…
- Probability of detection in the lab: Depends on setup.
- Probability of bug in final product : **High!**

How do we avoid corner cases?

- Most corner cases are given by specification
  - These cannot be avoided in the design
- Some corner cases are added by implementation
  - Some of these cannot be avoided
  - Some can definitely be avoided

A corner case is not a problem in itself.

It is only a problem when the design doesn't work for this case.

AND this is detected late in the FPGA development

OR even worse - not detected until delivered

Detection – for the communication interface example

Communication interface example (2)

```
if (rx_data_reg read from cpu) then
   <something>
elsif (new byte received from rx) then
   <something>
end if;
```

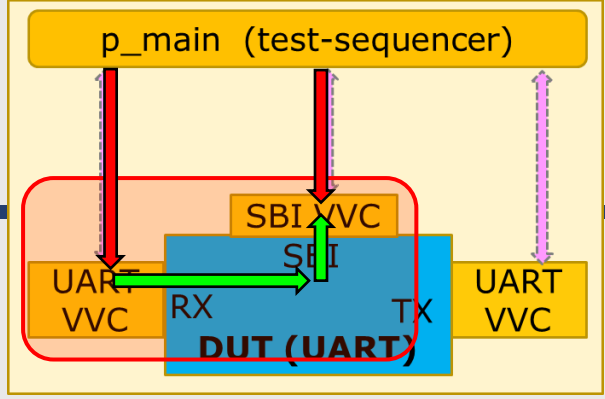
If-then-else
for non-exclusive actions
e.g. inside single process in UART:

For clock @ 100 MHz and bit rate @ 100 kHz (→ byte @ 10 kHz)

- Probability of bug = 1:10.000 per byte
- Probability of detection in sequential simulation:
  - For one single byte: 1:10.000
  - For thousands of bytes: 1:10.000 – in lots of testbenches
    > 1:10.000 – in quite a few testbenches but not necessarily much better...
- Probability of detection in the lab: Depends on setup.
- Probability of bug in final product : **High!**

- Awareness and experience
  - Generally very low
  - Will easily miss such bugs
- Simulation may detect all
  - Experience important
  - Approach matters
  - Need a good TB architecture - to skew interface stimuli
- Review?
  - Important, but complex
- Lab tests?
  - Lots of test cases possible, but...
  - Will seldom test for cycle relations
  - Often restricted by appl. SW

Verification approach



MUST skew one interface access with respect to another

UVVM allows simple targeting of any corner case

Parallel operation - using UVVM VVCs

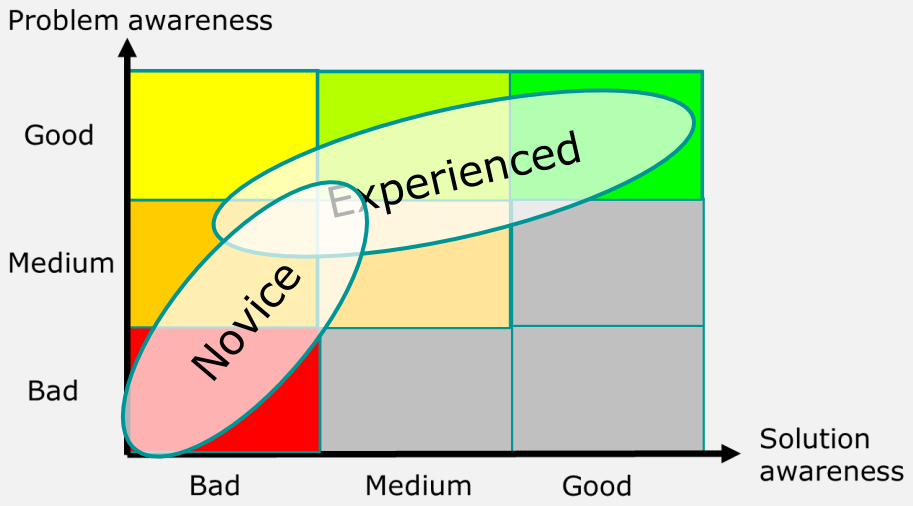
Meaning that sbi_check is reading previously received data

```
for i in -C_CYCLES_BEFORE_CORNER to C_CYCLES_AFTER_CORNER loop
  uart_transmit(UART_VVCT,1,TX, data(i), "New byte on UART RX");
  insert_delay(SBI_VVCT,1, C_FRAME_TIME +  i * C_CLK_PERIOD);
  sbi_check(SBI_VVCT,1, C_ADDR_RX_DATA, data(i-1), message);
  await_completion(UART_VVCT,1,TX, "Finish before next transmit");
end loop;
```

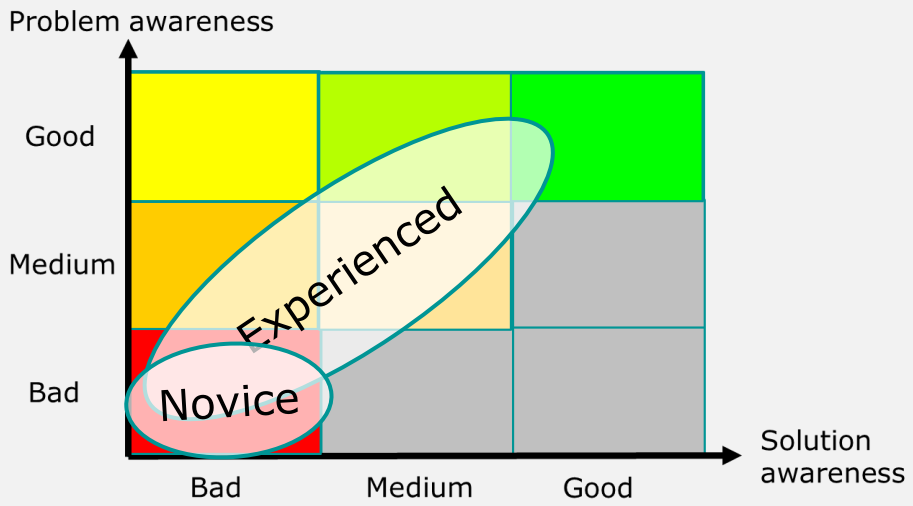
➔ Will hit this corner case within a few iterations

Awareness comparison

For Clock Domain crossing and Value related corner cases



For Cycle related corner cases



Occurrences, Error probability and Severities are comparable

EmLogic

Conclusion

- Most corner cases are given by specification
- Should avoid adding more corner cases by implementation
- Should make sure to verify all corner cases by simulation

- A serious lack of awareness and knowledge on cycle related CC
- **Need a verification system that can hit cycle related CCs**

➔ A corner case is **always** a challenge

➔ Without sufficient awareness and attention
A corner case could be a **major** problem

We need far more awareness on cycle related corner cases

EmLogic

EmLogic

- Independent Design Centre for Embedded Systems and FPGA
- Established 1st of January 2021. **Extreme ramp up**
  - January 2021: 1 person
  - May 2022: → 23 persons (SW:9, HW:3, FPGA:10, DSP:1) - **And still growing fast…**
- Continues the legacy from bitvis
  - All previous Bitvis technical managers are now in EmLogic
- Verification IP and Methodology provider **UVVM**
- Course provider within FPGA Design and Verification
  - Accelerating FPGA Design (Architecture, Clocking, Timing, Coding, Quality, Design for Reuse, …)
  - Advanced VHDL Verification – Made simple (Modern efficient verification using UVVM)
- A partner for ESA projects – (More opportunities due to Norway's low geo return)