Checking Security Path with Formal Verification Tool: New Application Development

Julia Dushina, STMicroelectronics; Saumil Shah, Cadence;
Joerg Mueller, Cadence; Vincent Reynolds, Cadence

Security is a Major Concern

Diverse industries affected

- Credit cards, Set-top-boxes, …

Security failure = Financial failure

- Or even loss of business

Set Top Box division of ST had security concerns

- Needed a solution

Simplified (Security!) Problem Description

Design manipulating keys

- To decrypt data stream provided to consumers' cable and satellite end point

Must check keys are not accidently accessible from internal interfaces

- Security path is safe

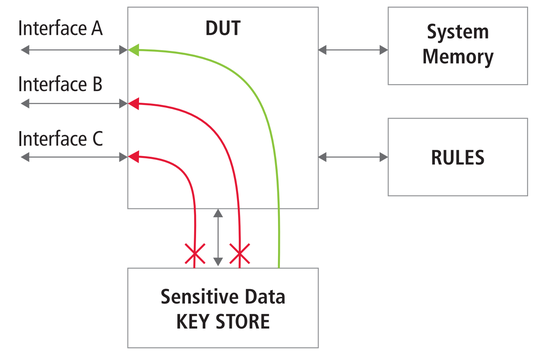
Three major interfaces

- Only one interface can read keys back if rules allow that!

There are also System memory and RULES blocks

- Can be accessed by the design

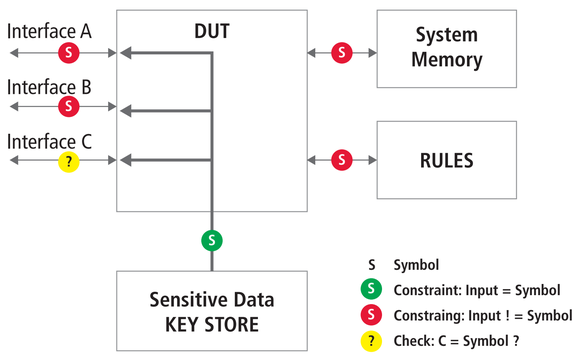
Simplified Design (Key Table) Diagram



Only interface A can read keys back if allowed by the RULES!

- Interfaces B and C can NOT read keys

Approach #1
Symbolic Approach for the Interface C: Idea



Approach #1
Symbolic Approach for the Interface C: PSL

Define a new signal symbol and constrain it to be rigid:

- *constr -add -rigid symbol*

Prohibit all other interface to have the data equal to the symbol:

- *constr -add -inter { interface_a _data != symbol} -name const_data_on_a*
- *constr -add -inter { interface_b _data != symbol} -name const_data_on_b*
- *constr -add -inter { interface_sys_mem _data != symbol} -name const_data_on_sys_mem*

Show with cover the symbol can enter the design at the key store:

- *assert -add -inter { keystore_r_data == symbol } –cover -name cover_symbol_on_key_store*

Prove the absence of the data on the interface C:

Cover must fail!

- *assert -add -inter { (interface_c_return_data == symbol) } -cover -name cover_no_symbol_on_c*
- *assert -add -inter { ! (interface_c_return_data == symbol) } -name check_no_symbol_on_c*

Assertion must pass!

Approach #1
Symbolic Approach for the Interface A

Keys cannot be read by the interface A if prohibited by the rules block

Define a new signal *address_symbol:*

- *constr -add rigid address_symbol*

Prohibit reading a key of the above symbol address:

- *constr -add -inter { (interface_a_req == 1 and interface_a_address == address_symbol) |=> sys_memory_allowed_for_reading_signal == 0} -name constr_no_reading_rules*

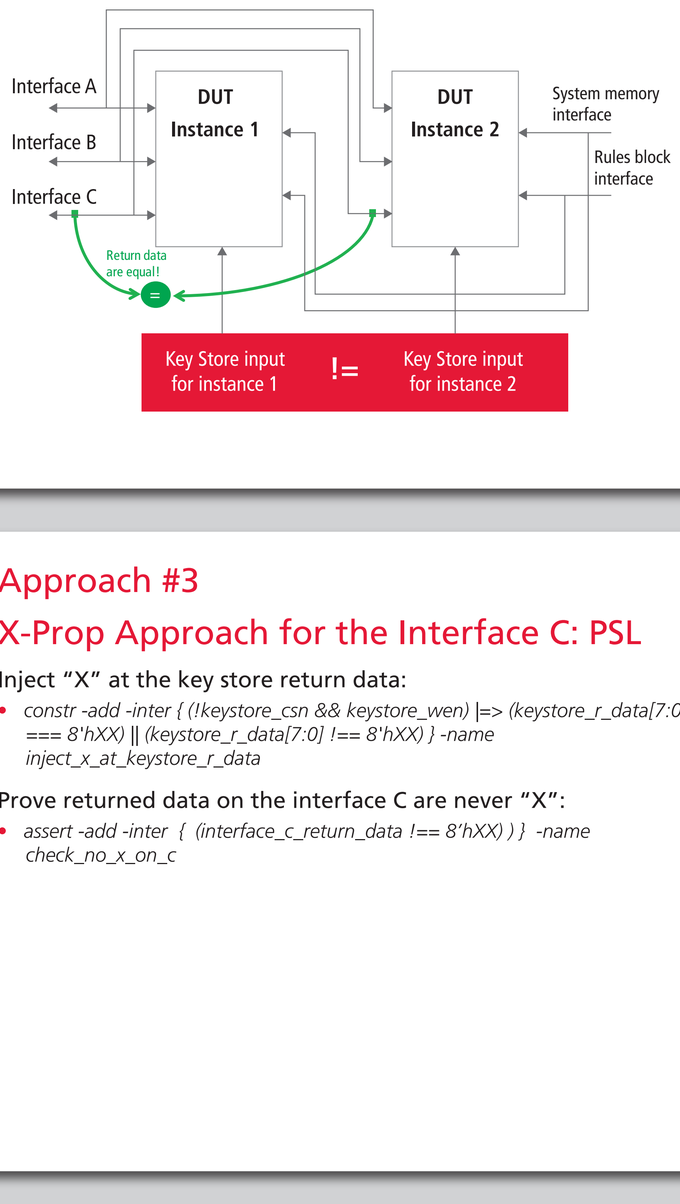
Modify main properties for the interface A:

Cover must fail!

- *assert -add -inter { (interface_a_return_data == symbol and interface_a_address == address_symbol)) } -cover -name cover_no_symbol_on_a*
- *assert -add -inter { ! (interface_a_return_data == symbol and interface_a_address == address_symbol)) } -name check_no_symbol_on_a*

Assertion must pass!

Approach #2
Miter Approach for the Interface C: Idea



Approach #2
Miter Approach for the Interface C: PSL

Tie all inputs but from the Key Store together

Show the Key Store inputs can differ for two instances:

- *assert -add -inter { (key_store_input_inst1 != key_store_input_inst2)) } -cover -name cover_different_key_store_inputs*

Prove returned data on the interface C are the same:

- *assert -add -inter { (interface_c_return_data_inst1 == interface_c_return_data_inst2)) } -name same_data_on_c*

Approach #3
X-Prop Approach for the Interface C: Idea

- Uses IEV capacity to propagate "X"
- Similar to the Miter approach
- Instead of tracing unique different value between two instances, an "X" value is traced!

Approach #3
X-Prop Approach for the Interface C: PSL

Inject "X" at the key store return data:

- *constr -add -inter { (!keystore_csn && keystore_wen) |=> (keystore_r_data[7:0] === 8'hXX) || (keystore_r_data[7:0] !== 8'hXX) } -name inject_x_at_keystore_r_data*

Prove returned data on the interface C are never "X":

- *assert -add -inter { (interface_c_return_data !== 8'hXX)) } -name check_no_x_on_c*

Advantages of Each Approach

Symbolic approach

- interactive and learning process when creating constraints => increase confidence in the design

Miter approach

- observed value is unique by construction; no need for constraints => more automatic

X-propagation

- Uses build-in IEV capacities => even more automatic

Conclusion

Three security paths are verified

- Symbolic approach used as the most "confident" and "investigated"
- Black boxing was used to conclude on assertions
- Proof real life time varying 5 mins – 30 mins

New IEV application is created

- Based on X propagation
- Automated set-up including covers, constraints and checks
- Provides witness waveform in case of security leakage