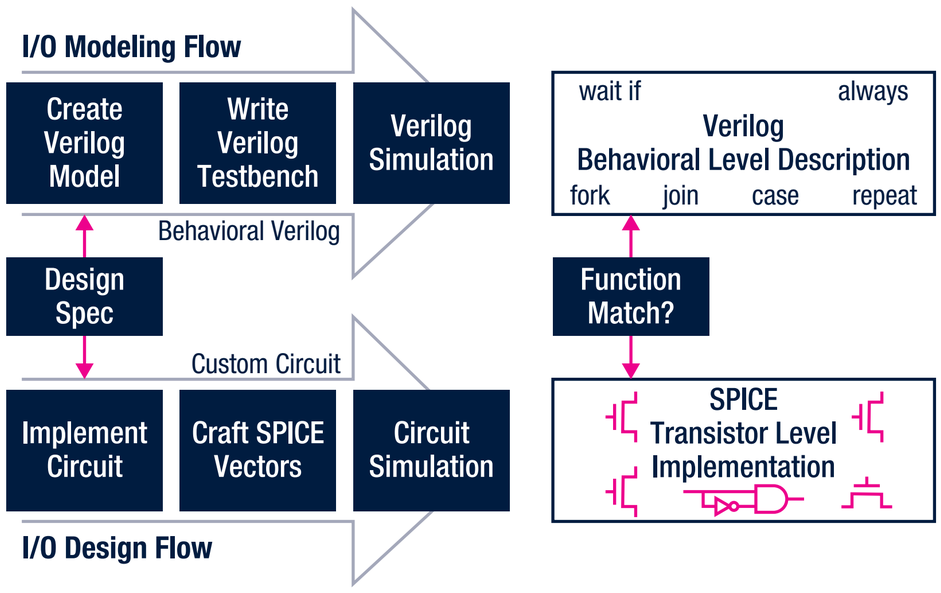
An efficient and SPICE-aligned method for complex IO behavior model generation and verification

life.augmented

By Anil Kumar Dwivedi, Harsh Garg, Avinav Joshi, Natish Singla, Saurabh Srivastava from STMicroelectronics | Matthieu Fillaud from Siemens
Email: anil.dwivedi@st.com, harsh.garg@st.com, avinav.joshi@st.com, natish-fet.singla@st.com, saurabh.srivastava@st.com, matthieu.fillaud@siemens.com

1. INTRODUCTION

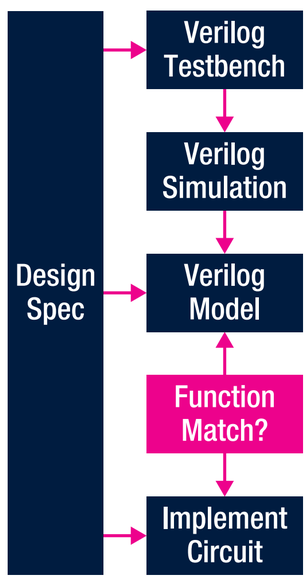


Limitations with conventional generation and verification methodology

- **Design understanding:** deep design understanding is required to create Verilog model and stimuli file for verification
- **Complexity:** increasing IO complexity make it difficult to write a manual Verilog model and create complete binary vectors for exhaustive verification
- **Number of input pins:** with many pins, writing Verilog models and then applying complete binary vector combinations for verification become impractical. (complete binary vectors can go up to 2^N, where N is number of input pins)
- **Equivalence:** even after applying the testbench to verify the Verilog model, there is no method to check the equivalence between the schematic design and the Verilog

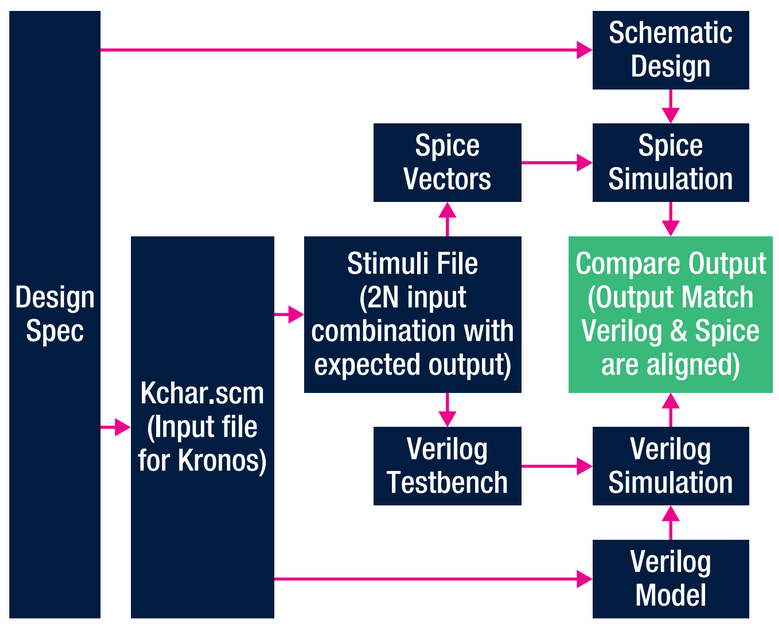
2. IO CONVENTIONAL GENERATION & VERIFICATION METHODOLOGY

- **Manual process:** the conventional methodology for the generation and verification of IO Verilog models is manual.
- Using the design spec as reference the following views are created manually:
  - Schematic design
  - Verilog model
  - Verilog testbench
- **Effort intensive:** same design spec needs to be understood by three different designers to create three different models manually
- **No equivalence check:** no way to check Verilog and the schematic design for equivalence
- Verilog testbench is applied to the Verilog model, to check the functionality of the Verilog, but this approach has many issues:
  - Verilog output corresponding to each stimulus needs to be manually checked which itself is error prone
  - How to ensure completeness of testbench?
  - Testbench generation time increases exponentially as the number of test cases increases



3. PROPOSED GENERATION & VERIFICATION METHODOLOGY FOR IO

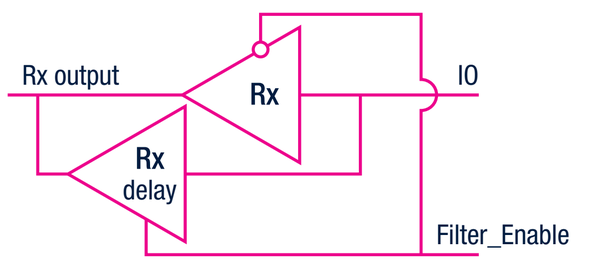
- In collaboration with **Siemens**, a flow has been implemented in **Kronos**, where the user can describe the functionality of the design in **kchar.scm**
- **kchar.scm** is used in Verilog generation as well as in the verification flow
- Verilog is generated using the **kchar.scm**.
- From the same **kchar.scm**, **Kronos** generates the stimuli file, which has 2^N all the input combinations and corresponding expected output. (N is number of inputs)
- Based on the stimuli file, a testbench is generated and applied to the Verilog model



- Using the same stimuli file, spice vectors are generated and applied on the schematic netlist
- Now, the output from the Verilog model and schematic netlist is compared by **Kronos**. If the output from both Verilog and schematic netlist matches, it implies that both are aligned
- If the output differs from the expected output in any of the views (Verilog/schematic netlist) it means that they are not aligned, and **Kronos** flashes the input stimuli in which there is a misalignment

4. VERILOG AND DESIGN MISMATCHES CAUGHT BY PROPOSED METHODOLOGY

Verilog issue as glitch is not modeled



In I2C cell, filter enable signal was found to be modeled wrongly in the Verilog model.

- Glitch is observed at Rx output in SPICE whenever filter is turned ON, even if input at IO is stable
- The glitch is expected but it was not modeled in Verilog

Schematic mismatch with specifications

In one of the IO design, spice behavior was not matched with specifications:

- For the enable signal low, output is expected at high impedance state, which was correctly modeled in Verilog
- But spice output was still following data signal so leading to design update.

5. SUMMARY & RESULTS

Parameter	Conventional methodology	SR_latest
Schematic netlist and Verilog equivalence	Few or none test cases are validated on schematic netlist and Verilog.	Compare the schematic netlist and Verilog for 2N vectors. Testing of more vectors brings consistency in model w.r.t design.
Verilog testbench	Manual and does not contain all the test cases	Automated and contains 2N test cases.
Debugging/output checking	Manual (waveform viewing and validating the output w.r.t. specs)	Automated (outputs are validated corresponding to the expected output)
Coverage	~40-50%	>95%
Cycle time	~1 Man week	~0.4 Man week (Approx. 60% gain in cycle time)

Future scope

- Automatic parsing of netlist and creation of the kchar.scm
- Extend this solution to supply modeling and create power aware Verilog models