CVC: Circuit Validity Checker
An open source netlist reliabilty verification system

D. Mitch Bailey
Shuhari System
Fukuoka, Japan
d.bailey@shuharisystem.com
https://github.com/d-m-bailey/cvc

***Abstract*— This paper presents a device level static verification system for quickly and easily detecting common circuit errors in CDL (Circuit Definition Language) netlists. The system includes the capability to define device parameters and voltages for multiple modes in Microsoft Excel, an interactive option for examining intermediate values, and a GUI for analyzing the results. Errors detected are similar to those found using Mentor's Calibre-PERC [1] or Synopsys' CCK [2]. CVC has been used on dozens of DRAM and SOC designs of up to 3B devices.**

Keywords—CVC, reliability verfication, PERC

I. Introduction

The design of mutli-million gate chips often involves several distinct teams and can include third party modules. Circuit errors can occur for a variety of reasons and may be difficult to detect. For example, interactions between components are expected to follow specifications, but detailed simulation of the full chip to verify the correctness of the specifications is rarely possible because of time constraints. Additionally, spare circuitry connected incorrectly may not be detected because it is assumed to be inactive and therefore not included in some simulation flows. Also, while logic circuit analysis is relatively straight forward, over conservative static analysis of analog circuits can result in too many errors to reasonably verify before tapeout. CVC (Circuit Validity Checker) was developed to quickly pinpoint circuit errors in mixed-signal and low-power designs, including those errors that are difficult to detect with other simulation or verification tools.

II. CVC

A. Philosophy

CVC was designed as a full chip verification system, and accordingly, uses the device level LVS (Layout vs Schematic) netlist as input. This avoids the problems caused when the simulation netlist does not include all the devices in the layout. CVC differs from other tools which detect similar errors in that there is no coding required by the user and all errors for a particular mode are detected in a single run. The user specifies the voltages for the necessary nets, the parameters for each device type, and optionally a few runtime settings. This prevents errors missed due to not running all checks or forgetting to include rules for every device model.

B. Input Parameters

CVC is able to verify netlists with both enhancement and depletion mode n-mosfets and p-mosfets - including LDD (lightly doped drain) devices - bipolar transistors, diodes, and two or three terminal resistors and capacitors. These primary devices may additionally be treated as switches or fuses which may be either on or off. Mosfets require a Vth (threshold voltage) setting and any device may have a maximum allowable voltage specified across any two terminals.

Each voltage definition has a minimum value, a simulation value, and a maximum value. For external power definitions, normally these three values are equal. Internal power definitions may be a minimum and/or maximum limit, a simulation value, or any combination. Voltages are defined from -32V to 32V at 0.001V precision. The voltage levels for nets defined as resistors are automatically calculated using Ohm's Law. Variable input signals are defined with minimum and maximum values only. Voltage levels can be expressed as equations using previously defined nets and mosfet Vth, e.g. `VDD+Vth[PCH]`. Power nets may be defined as 'open', which will result in an error if propagation could result in a current leak. Nets may be defined using wildcards as in Unix shell or regex and may be defined at any level of the hierarchy by using a `*(circuit_name)/net_name` syntax.

Minimum and maximum power are propagated simultaneously throughout the entire circuit, while simulation values are only propagated when known. After each propagation, certain types of errors are detected.

C. Error Types

Min/max errors

Mosfet gate vs source: mosfet gates that will never be fully off. The minimum gate voltage for an n-mosfet is greater than its minimum source voltage or the maximum gate voltage for a p-mosfet is less than its maximum source voltage.

Mosfet source vs bulk: forward biased junction diode. The minimum source voltage for an n-mosfet is less than its maximum bulk voltage or the maximum source voltage for a p-mosfet is greater than its minimum bulk voltage.

Forward bias diodes: diodes that are not always reversed biased. This includes parasitic diodes on 3 terminal resistors or capacitors.

LDD errors: For devices defined as LDD (source and drain are not interchangeable), an error is flagged if the drain is closer to ground than the source for n-mosfets and if the drain is closer to power than the source for p-mosfets.

Simulation errors

Leaks: any static leak between external power or any leak above a defined threshold for other signals.

Simulation and min/max errors

Hi-Z (high impedance) leaks: After simulation propagation, if a gate net has no path to both power and ground, it is considered as a possible Hi-Z input error. If the min/max paths for the mosfet source/drain are to different power nets, then the mosfet is flagged as an error.

Possible leaks: a mosfet connected between 2 different external power nets will be flagged as an error if the simulation level is unknown. This detects errors when gate outputs are tied to power, but input nets are unknown.

Electrical overstress: the maximum voltage across two terminals exceeds the defined limit. Optional conservative checks include the voltage levels before simulation propagation.

Expected values: expected values may be set for the min/sim/max value of any net. After the final propagation, errors are raised if the expected value does not match the calculated value. An expected value of 'open' will flag an error if there is a path to any power net.

D. Program Flow

CVC is written in C++ for speed and parses the netlist with bison/flex. All strings from the netlist are compactly and uniquely stored in an obstack structure [3] and referenced by address. As a typical example, the text for an 800MB netlist is read in less than 2.5 minutes and stored in less than 25MB of memory. The netlist is flattened internally and switches are shorted. Voltage drops across resistors are calculated where possible, followed by the first min/max voltage propagation and relevant error detection. During the subsequent simulation voltage propagation, current leaks are flagged. Using the results of the simulation voltage propagation in the final min/max propagation allows us to detect true Hi-Z input errors while ignoring errors with no leak path. Table I shows representative runtimes for a 700M device DRAM design.

TABLE I. CVC RUN TIMES

Stage	Time (s)	Total Memory (GB)
Input	151	14
Flatten	29	72
Resistor	117	72
First min/max propagation	1174	72
Simulation propagation	198	78
Final min/max propagation	1308	79
Total	**2977**	**79**

III. INTERACTIVE MODE.

CVC has an interactive mode that allows the user to show the propagated voltages at each net and examine the netlist. Starting at the top level of the netlist, the hierarchy may be traversed using the Unix directory command, `cd`, and `pwd` will display the current hierarchy. Command history and file name completion are also available. Wildcards are supported for listing nets and devices. For example, `listnet V*` will list all nets in the current hierarchy that begin with 'V'. `findnet foo*` will list all nets below the current hierarchy that match foo* and show the highest net name for that net. A sample output might be
`/Xtop/Xmid/Xmon/foobar -> /Xtop/foo1`.
`findsubcircuit INV*` will list the full hierarchy for each instance of the subcircuits that begin with 'INV'. In addition to the usual manner of displaying a hierarchical item,
`/XTOP/XMID/XINV/OUT`,
there is an option to include subcircuit names,
`/XTOP(TOP)/XMID(IOBLOCK)/XINV(INV)/OUT`.
`printcdl test` will write the subcircuit 'test' and all its children to a file. The `debug` command will create the files necessary (CDL, power, and environment file) to run CVC on a specified instance of the netlist. This can simplify and speed the analysis of IP.

IV. GUI

CVC and other ERC (electrical rule check) tools can generate many false errors. Some errors are caused by incorrectly propagating voltages in analog circuits while others may be intentional (i.e. gate vs source errors in level shifters). The user must decide whether to add or change the assigned voltages to eliminate the error or flag the error as an exception. While some tools hide all exceptions in the final report, the philosophy of CVC is to use the analysis GUI to tag each error according to its severity. Errors are listed by type and show the lowest level subcircuit and device name along with the number of error instances and total instances. Each subcircuit has an associated checksum, so that changes to the subcircuit are flagged to ensure re-evaluation. Here is a sample line from the GUI.

```
GATE INFO: SUBCKT (MON)/Mp1(PCH) error count 1/6 checksum(380633301 942)
```

Error details – the device terminal connections and full hierarchy – are available in a popup window. There are 4 error levels: 'ERROR' – must be fixed before signoff, "Warning" – problem circuit approved by the circuit designer, "Check" – possible problem that requires circuit designer check, "ignore" – false errors. Each error is assigned a reference along with an error level. This reference may refer to a bookmark of the associated circuit and may contain an error code. For example, 'A001LS' could mean that the circuit can be viewed at bookmark A001 and the error is related to level-shifters. Analysis results are stored in a separate summary file. The GUI shows errors in one of 6 states: checked – ERROR, Warning, Check, or ignore, unchecked – only in error file, unconfirmed – checksum has changed, uncommitted – copied from other mode, unmatched – only in summary file, comment – errors from previous versions that have been fixed. The GUI is capable of displaying the results of multiple verification modes to allow simultaneous analysis. When analysis is complete, the results may be exported to a CSV file for confirmation by the design team.

REFERENCES

[1] https://www.mentor.com/products/fv/multimedia/overview/overview-of-calibre-perc-18d4001a-c1ac-4af0-81c2-1895642f0606

[2] https://www.synopsys.com/verification/ams-verification/reliability-analysis/customsim-circuit-check.html

[3] https://en.wikipedia.org/wiki/Obstack