2018 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Formal Verification of Connections at SoC-level

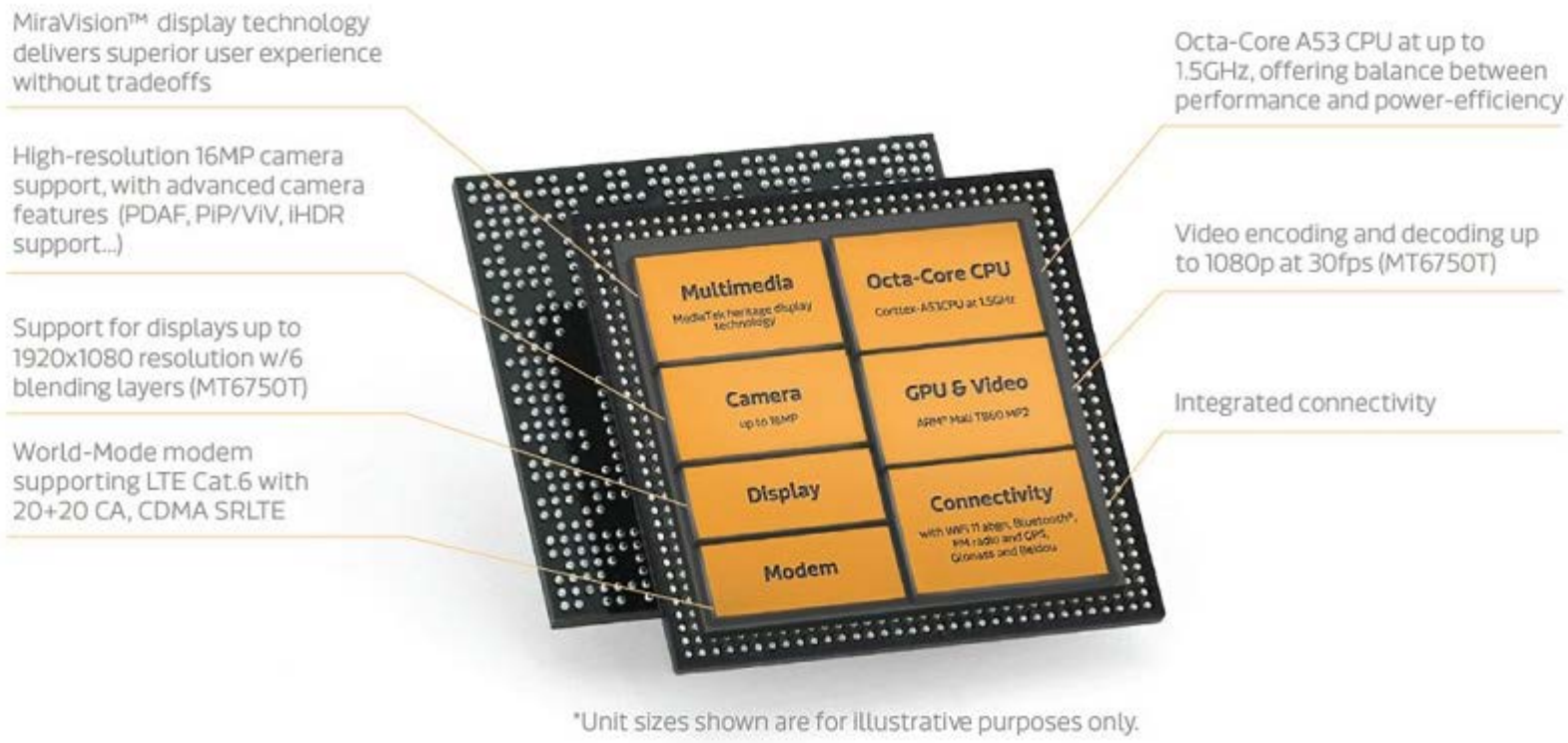
Penny Yang, Prasun Das, Yuya Kao, Mingchu Kuo

MEDIATEK

SYNOPSYS®
Silicon to Software™

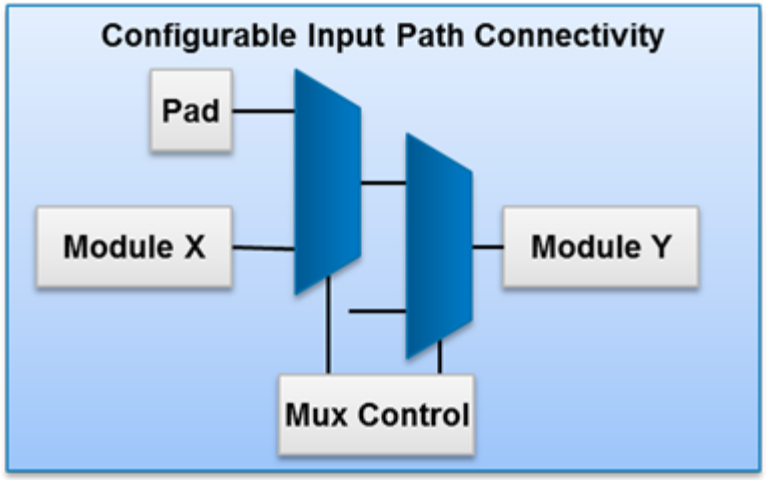
Introduction

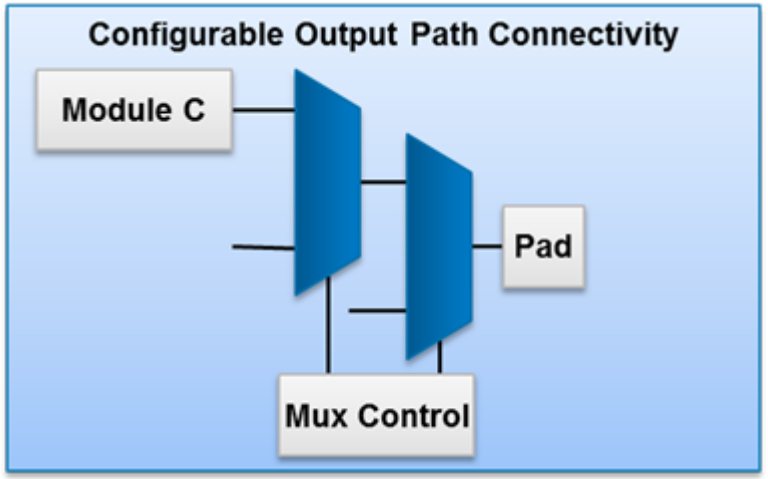
- Complete verification of connections at SoC level is a fundamental requirement to ensure correct operation



SoC Challenges – Padmux

- Larger gate count, but limited I/O pins
- Leads to shared I/O pins, muxing to control access
- Requires significant high level interconnect wiring
- Introduces significant possibility of errors
- Hard to do ECO because combinational logic is often optimized after synthesis



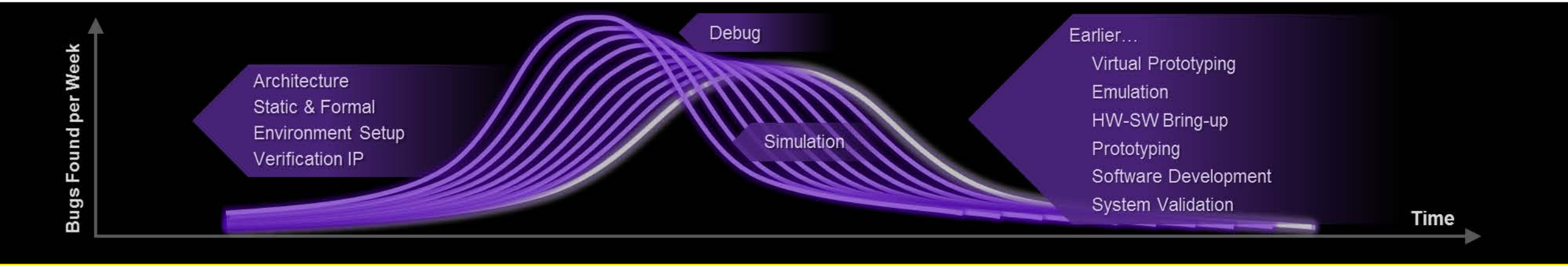


SoC Challenges – Global reset

- SoC designs have multiple sources of global reset, such as power-on reset, hardware reset, software reset and watchdog timer reset.
- These are top-level signals that should be connected to all asynchronous resets in the design, i.e., all asynchronous resets in the design should be asserted when the global reset is asserted.
- Taking watchdog reset verification as an example, a directed test created to verify if the watchdog reset operation works correctly must trigger the watchdog reset condition in the middle of the simulation to check if the watchdog reset has propagated to all the intended flip-flops in the entire SoC.

Background

- Before using formal verification, chip level simulation was used to verify the connections at SoC-level.
  - Fewer simulation patterns in chip level environment
  - Corner case bugs sometimes appeared in uncovered codes
  - Integration of chip level test-bench often comes late in a project cycle
- To be able to shift left, formal verification is adopted due to its exhaustiveness and easy setup.



Motivation

- Formal property verification has been proven to be a reliable method for different kinds of designs' verification signoff. However,
  - Run time could be several days and the iterations was very slow.
  - Manual abstraction is required to achieve full proof
  - Require design knowledge to partition the SoC into several different sub-systems, black-box the modules that were not used, constrain the designs with constants and assumptions.
  - Several months to work on these settings from the beginning till the end of the project
- Trial and error was time consuming and manual abstraction was also prone to false alarms.

Using formal Application on connectivity checking

- Formal Applications are customized for easy setup, use, and debug.
- It is perfect for beginners because there is no need to have formal background or knowledge to write SVA.
- Problems which are ideal for formal connectivity checking
  - SoC I/O Connectivity
  - Block pin muxing/demuxing
  - Connectivity & constant checking of macros
  - Scan mode connectivity & constant checking
  - Reset and global signal connectivity
  - Registers to Debug Bus

Difference between CC and FPV

Connectivity Checks

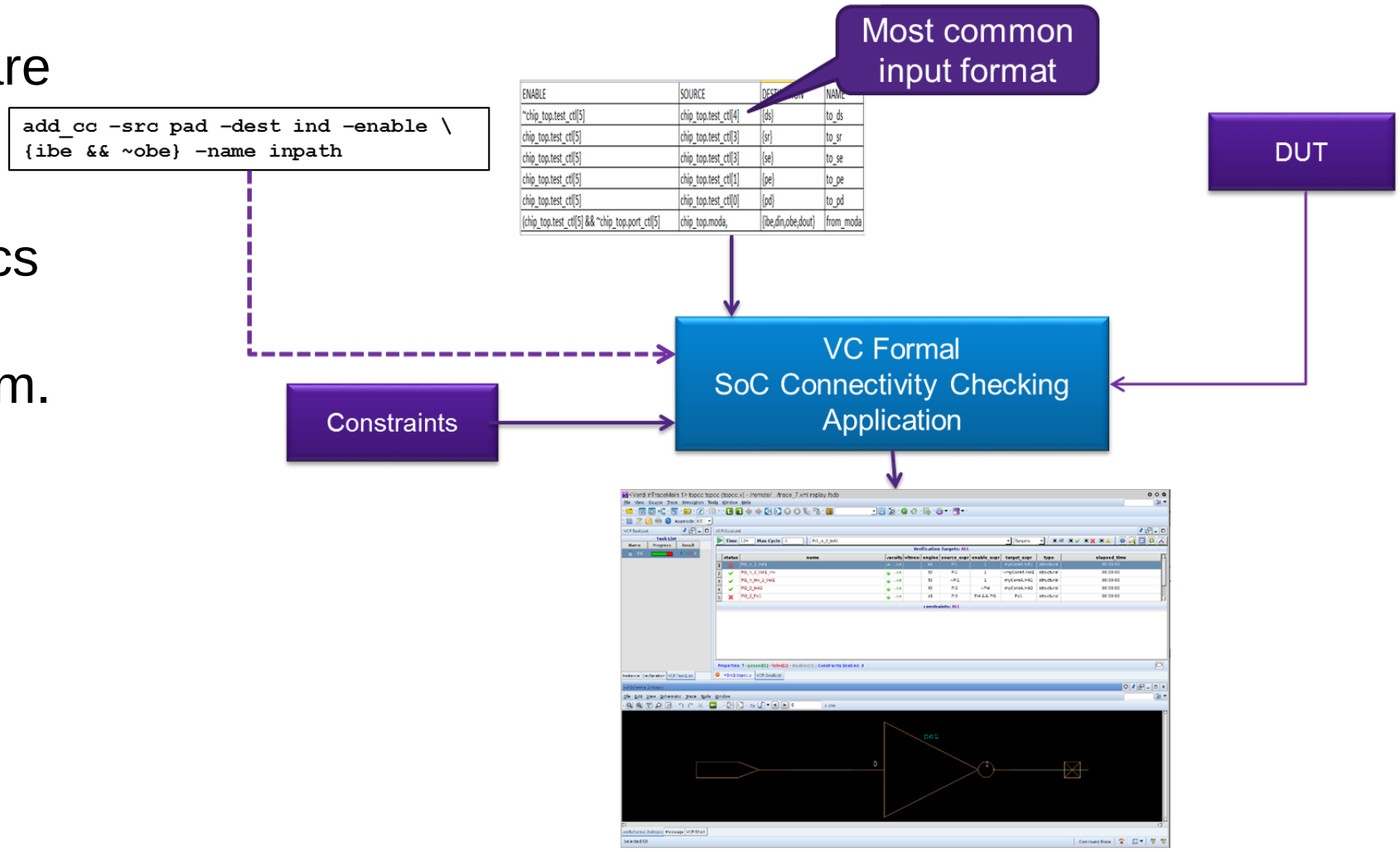
- Value check and directional connection
- With auto-blackboxing mechanism
- Can debug through schematic + waveform + text

Formal Property Verification

- Temporal and combinatorial signal relationships
- No notion of direction in SVA
  - “Out2 == Po2” same as “Po2==Out2”
- Manual partition and abstraction
- Can debug through waveform

Connectivity Checking App

- Tcl commands, CSV formats and Excel files are supported as the input format.
- User can debug the logics in the path using schematics/text/waveform.

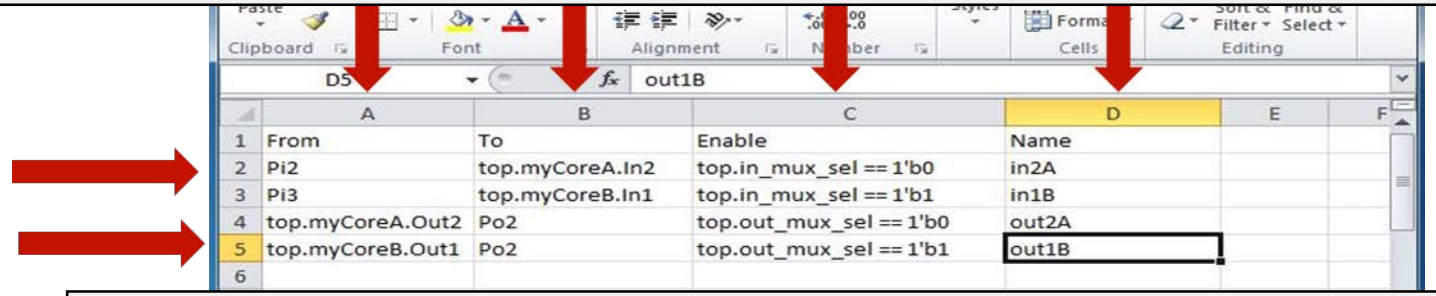


Input Formats

- Source
- Destination
- Enable
- Name
- Start line
- Comment pattern

```
# Generate reset and register list
report_ff_reset -list
# Reset path
set cpu_rst_en u_reset.wdr
source reset_checker.tcl
```

```
#reset_checker.tcl
add_cc -name c1 -src 0 -dest u_cpu.resetn_i -enable $cpu_rst_en
add_cc -name c2 -src 1 -dest u_cpu.g1.reset_i -enable $cpu_rst_en
…
```



	A	B	C	D
1	From	To	Enable	Name
2	Pi2	top.myCoreA.In2	top.in_mux_sel == 1'b0	in2A
3	Pi3	top.myCoreB.In1	top.in_mux_sel == 1'b1	in1B
4	top.myCoreA.Out2	Po2	top.out_mux_sel == 1'b0	out2A
5	top.myCoreB.Out1	Po2	top.out_mux_sel == 1'b1	out1B
6				

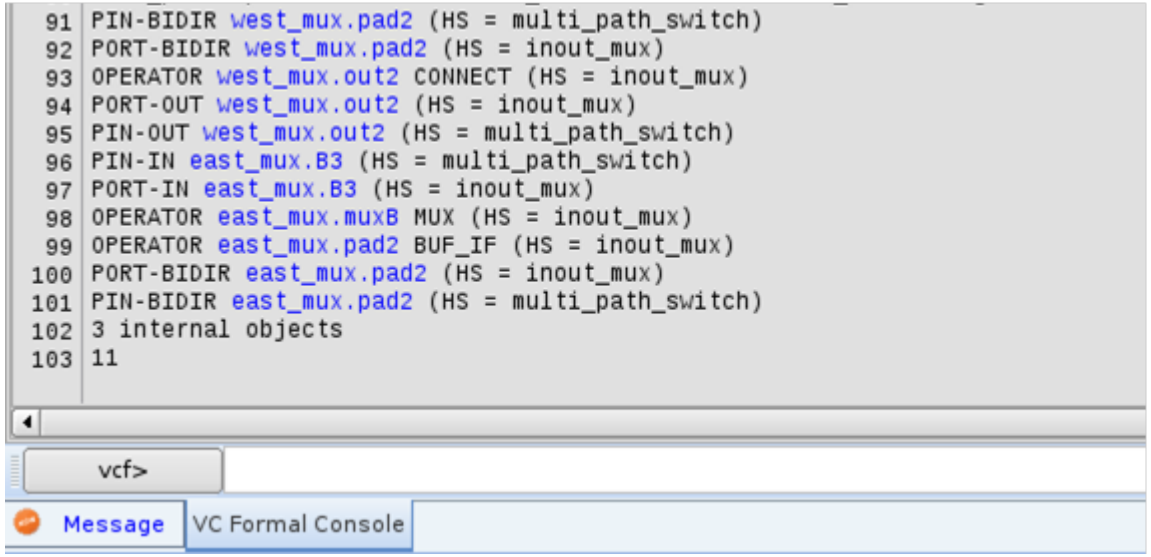
```
#padmux_checker.csv
u_cksys.tck, u_io.PAD_MCK.O,{gpio_mode==1}, a1
u_io.PAD_MDAT.I, u_cksys.dsp_out,{gpio_mode==6}, a2
…
```

Debug Structurally Disconnected Path

- Structurally disconnected check

```
[Error] CC_UID017: Connection 'InA2_2_Pi1' is structurally disconnected.
No path from source to target. Please fix the connection
```

- Debug specific path



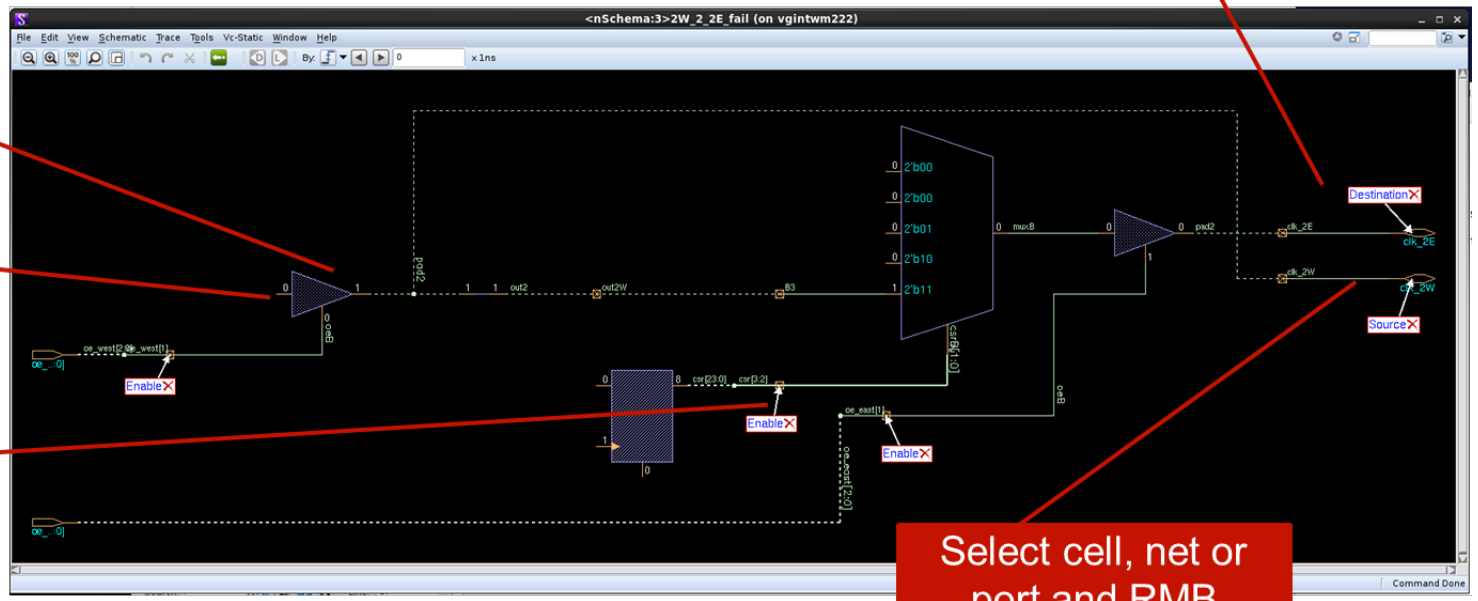
Debugging in Schematic View

Signal values annotated on schematic

Double click on input to show driving logic

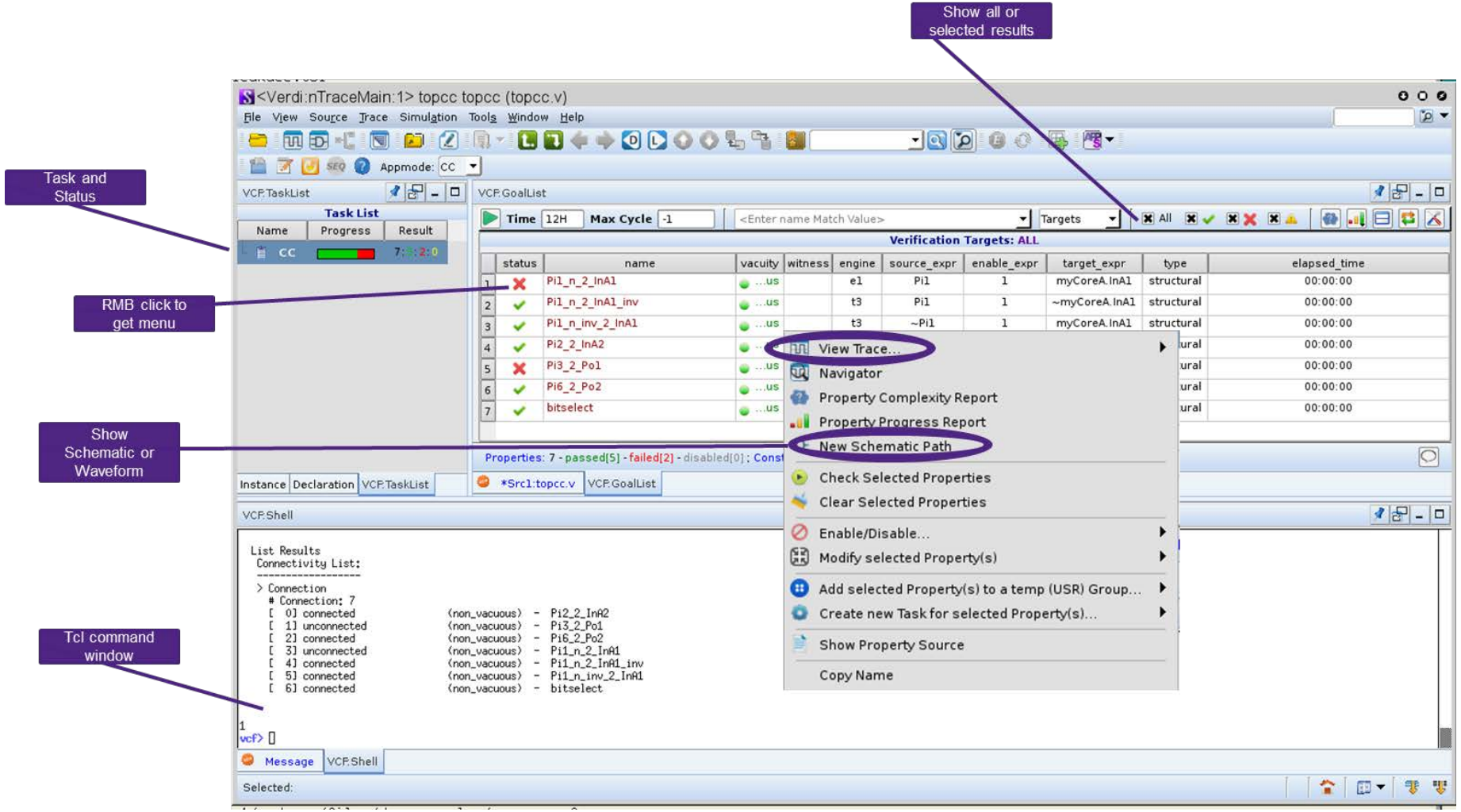
Select cell, right mouse click to show RTL source

Shows source, destination & enables



Select cell, net or port and RMB menu: copy signal path

Debugging in a single GUI platform



Results

Case	# of connections for verification	Verification time w/ traditional FPV based CC flow	Verification time w/ Optimized CC flow	Improvements
Reset 1	173981 (103971 proven, 70010 failed)	35 hrs	40 mins	52X
Reset 2	25000 (23241 proven, 1759 failed)	4 hrs	12 mins	20X
Reset 3	1 (1 proven)	8 hrs	3 mins	160X
Padmux 1	4732 (4731 proven, 1 failed)	4 hrs	3 hrs	1.3X
Padmux 2	447 (441 proven, 6 failed)	5.6 hr	3.4 hr	1.7X

The design is an SoC with 64,056,916 register bits

Results

- The application of CC is optimized for connectivity checking problems.
- The run time for formal verification is improved significantly with the innovative automatic abstraction flow optimized in CC compared to FPV.
- Thus, we can easily integrate the commands into MediaTek’s regular flow using either tcl or csv format

Conclusion

- This presentation presents highly automated methodologies using Formal techniques to verify the correctness of global reset schemes and padmux connection, without the large amount of effort required by manual abstraction in SoC.
- The methodologies described above have been deployed on 10 projects at MediaTek.
- As the results show, we have found that the strength of CC App is more efficient than pure FPV based connectivity verification methodologies.
- The connectivity verification flow can be completed in hours, without any inconclusive properties, on an SoC size design.

Future Work

- Measure Connectivity Checking Coverage
- Spec completeness – are all paths checked?
- Combine with simulation coverage data for connectivity checks
- Uses the same toggle coverage goals as VCS
- Creates a coverage database that can be merged with simulation connectivity coverage data

Connectivity Coverage in GUI