

Sequential Equivalency Checking with Cadence JasperGold

NYCU-EE IC LAB Fall-2024



Lecturer: Yen-Ning Tung

Outline

- ✓ **Section 1. Introduction to SEC**
- ✓ **Section2. JasperGold – SEC Setup**
- ✓ **Section3. JasperGold – Proof & Debug**
- ✓ **Section4. JasperGold – Clock Gating Verification**



Outline

- ✓ **Section 1. Introduction to SEC**
- ✓ Section2. JasperGold – SEC Setup
- ✓ Section3. JasperGold – Proof & Debug
- ✓ Section4. JasperGold – Clock Gating Verification



Equivalence Checking - EC

✓ Problem

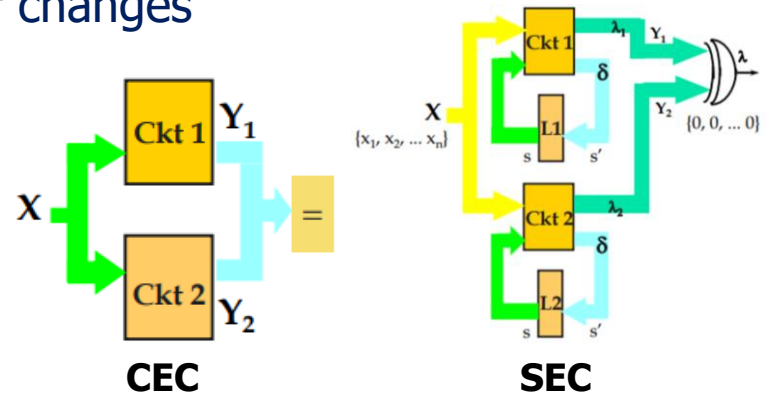
- Given one **RTL design (specification)** and another **optimized RTL/GL design (implementation)**
- Do they implement the same set of behavior?
- **Functional Verification of the change requires a lot of effort!**
 - Full regression required even minor changes

✓ Solution

- EC guarantees equivalence

✓ Equivalence checking

- **Combinational equivalence checking (CEC)**
 - Output depend only on present inputs
- **Sequential equivalence checking (SEC)**
 - Output depend on present inputs as well as past sequence of inputs



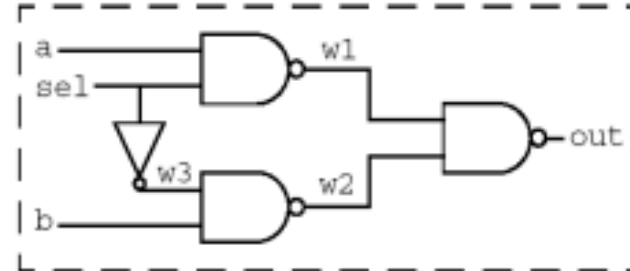
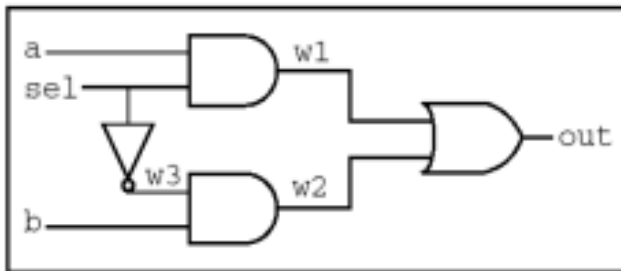
Example: Equivalence Checking

```
out = sel ? a : b ;
```

```
always @ (sel or a or b)  
  if (sel) out = a;  
  else out = b;
```

```
module des1 (out,a,sel,b)  
  output out;  
  input a,sel,b;  
  wire w1,w2,w3,  
    and u1(w1,a,sel)  
    and u2(w2,w3,b);  
  not u3(w3,sel);  
  or u4(out,w1,w2)  
endmodule
```

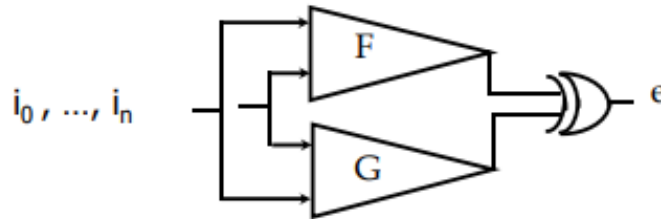
```
module des1 (out,a,sel,b)  
  output out;  
  input a,sel,b;  
  wire w1,w2,w3,  
    nand u1(w1,a,sel)  
    nand u2(w2,w3,b);  
  not u3(w3,sel);  
  nand u4(out,w1,w2)  
endmodule
```



CEC/LEC vs. SEC

✓ **Combinational Equivalence Checking (CEC, LEC)**

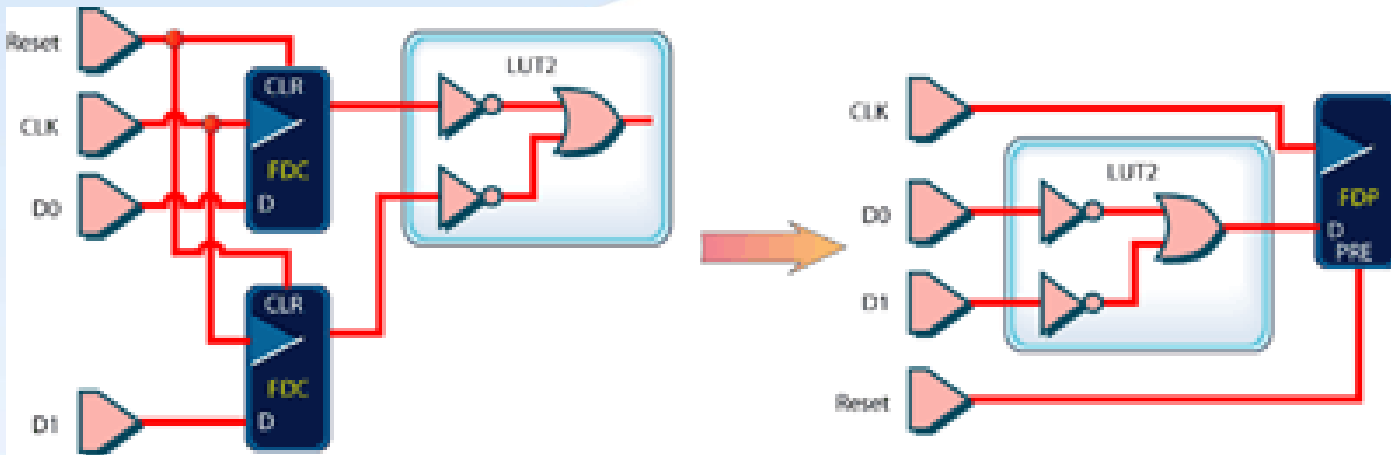
- Same functional behavior at all external ports and internal state element?
- Requires state-matching design
- Performs EC only on the remaining logical cones



✓ **Sequential Equivalence Checking (SEC)**

- Same functional behavior at all external ports?
- Applies to state-matching and non-state-matching designs
- Ignores internal sequential differences

Limitations of traditional LEC



✓ **Example is sequentially equivalent, but...**

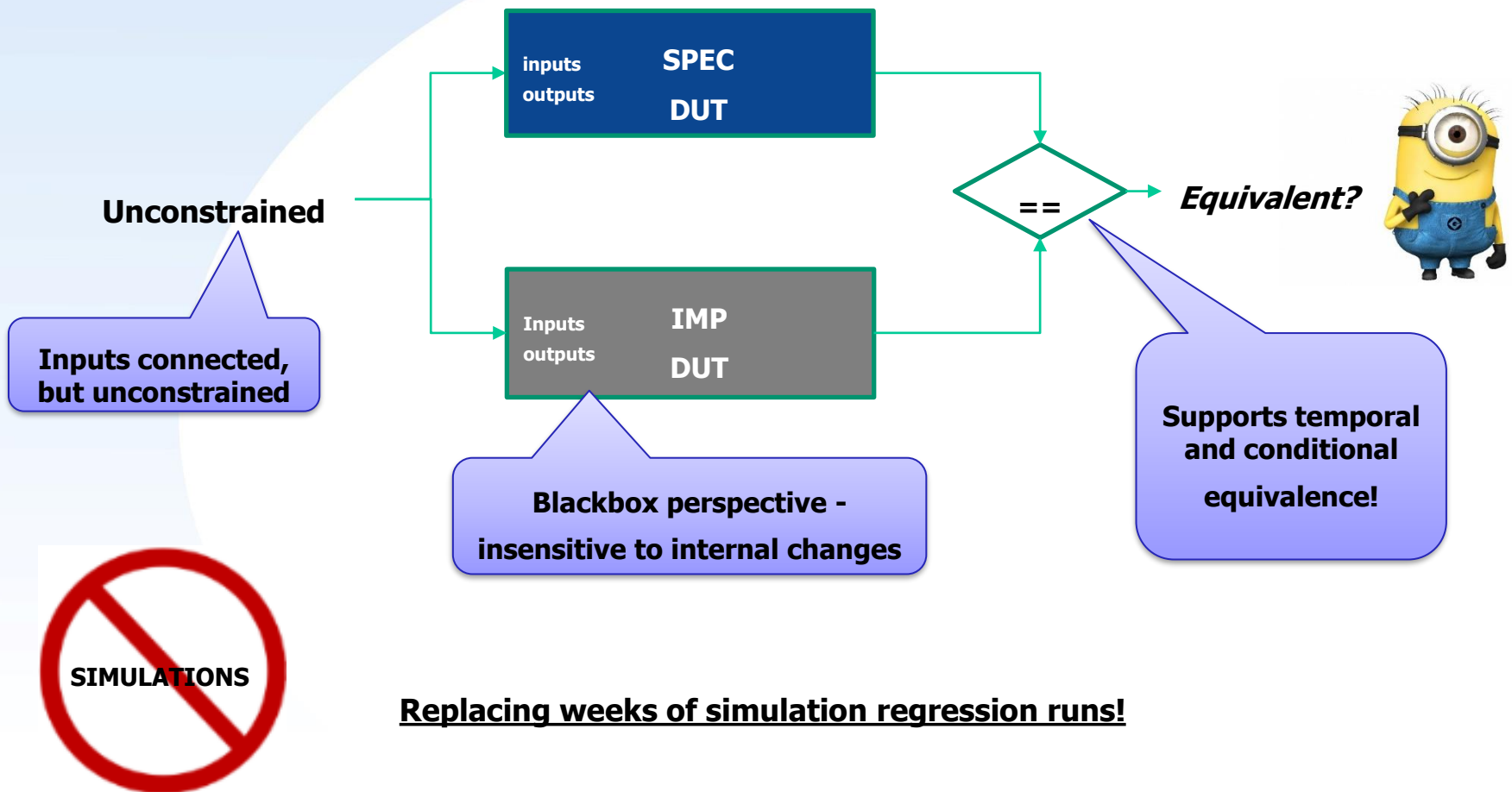
- States cannot be matched
- Internal logic cones are different

✓ **CEC → NOT Equivalent !**

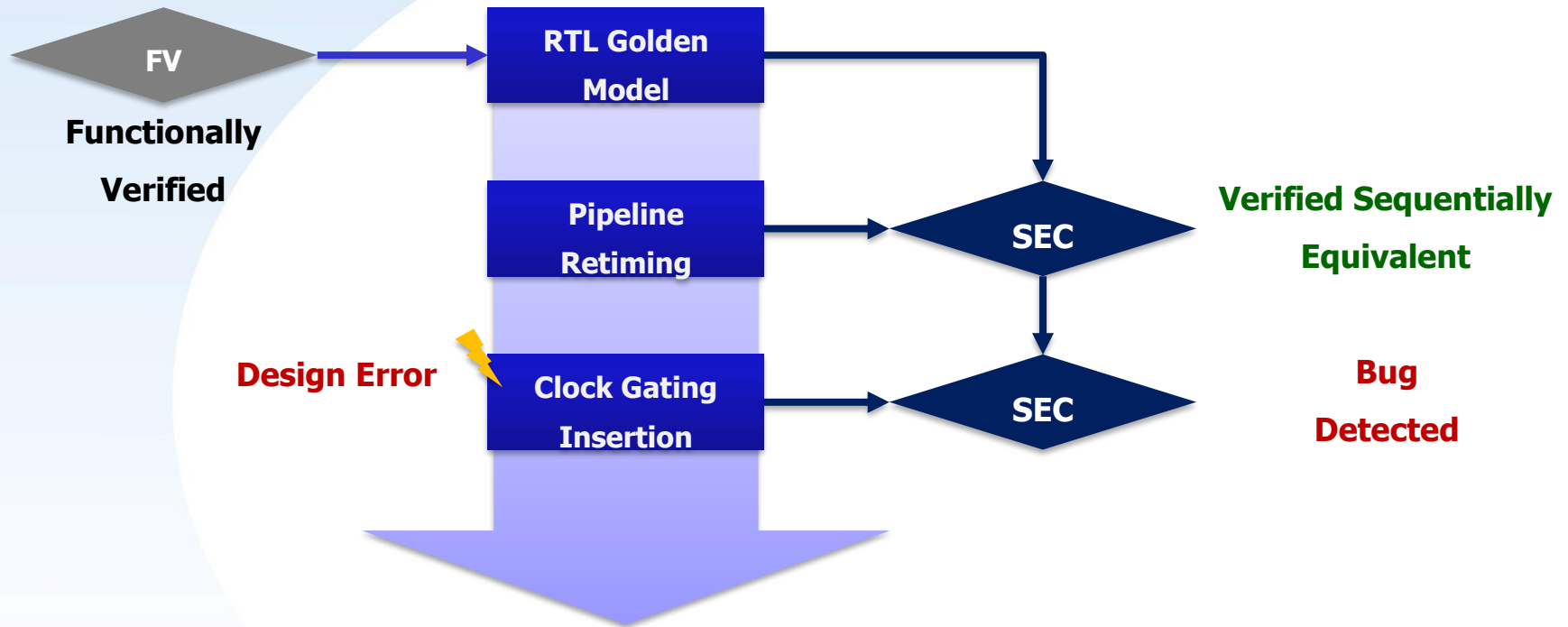
✓ **SEC → Equivalent !**



SEC Technique – Miter Construction



Example RTL Refinement Flow with SEC



SEC Adoption Hints

- ✓ **Recommend incremental flow, and position SEC as the main validation tool**
 - Start from golden refine for power, area, speed. Each step verified with SEC
- ✓ **Understand the similarity, know what is the difference**
 - “Similarity” is important – not design size: Flop count isn’t meaningful in discovery
 - Know the transformation, this helps splitting into sub transformations for SPEC
- ✓ **Clock gating is sweet spot**
 - If mapping constraints may make it involved
 - Almost all customer started with this
- ✓ **Re-partitioning and logic optimization is sweet spot**
 - High similarity. Seek in “implementation” teams for this transformation step

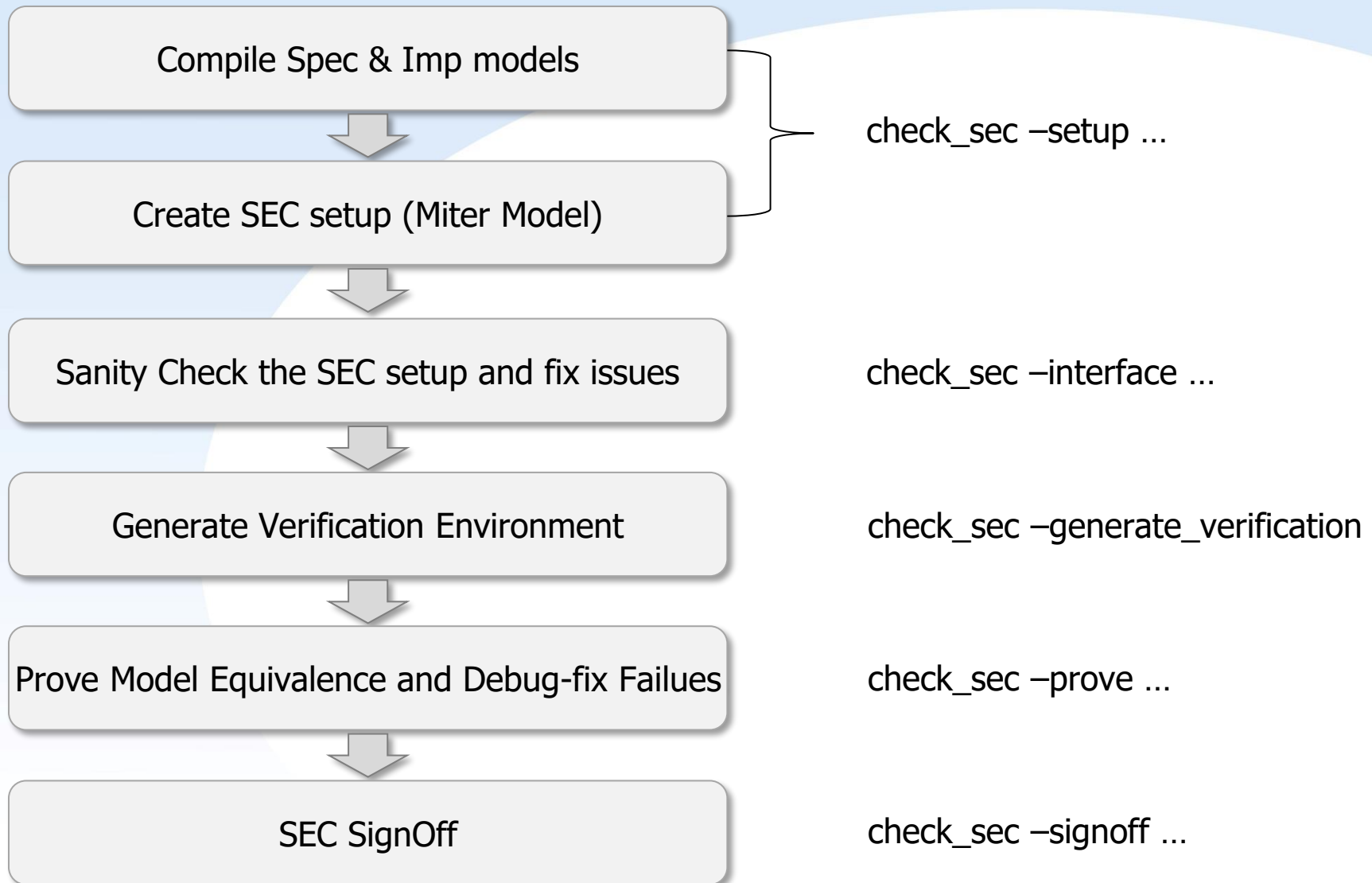


Outline

- ✓ Section 1. Introduction to SEC
- ✓ **Section 2. JasperGold – SEC Setup**
- ✓ Section 3. JasperGold – Proof & Debug
- ✓ Section 4. JasperGold – Clock Gating Verification



SEC flow at a glance



SEC Mapping GUI

The screenshot shows the Cadence SEC Mapping GUI interface. The main window is titled "JasperGold Apps (.../sec_demo/jgproject) - Main". The interface includes a menu bar, a toolbar, and several panels.

Signal type filters: A callout points to the "Filters" section in the "Hierarchy Browser" panel, which allows users to filter signals based on type.

Design browsers for both SPEC and IMP: A callout points to the "Spec Design" and "Imp Design" panels, which show the hierarchy of signals for the specification and implementation designs, respectively.

Mapping pairs: A callout points to the "Signal Mapping" panel, which displays a table of signal mappings between the specification and implementation designs.

Property table: A callout points to the "Property Table" tab in the "Signal Mapping" panel, which displays a table of properties for the mapped signals.

Signal Mapping Table:

Stat	ID	Spec Signal	Imp Signal	Property	Value
●	22	wr_rd2	top_imp.wr_rd2	Primary Input	Auto
●	23	wr_rd3	top_imp.wr_rd3	Primary Input	Auto
●	24	eg.i2c2int_data[7:0]	top_imp.eg.i2c2int_d...	Undriven	Auto
●	25	brdg.int2ig_data[31:0]	top_imp.brdg.int2ig_...	Undriven	Auto
✓	26	rdata0[31:0]	top_imp.rdata0[31:0]	Primary Output	Auto
✓	27	rdata1[31:0]	top_imp.rdata1[31:0]	Primary Output	Auto
✗	28	eg_valid	top_imp.eg_valid	Primary Output	Auto
✓	29	rdata2[31:0]	top_imp.rdata2[31:0]	Primary Output	Auto
✓	30	rdata3[31:0]	top_imp.rdata3[31:0]	Primary Output	Auto
✗	31	ready0	top_imp.ready0	Primary Output	Auto
✗	32	eg_ad_dataout[7:0]	top_imp.eg_ad_dataout[7:0]	Primary Output	Auto
✗	33	ready1	top_imp.ready1	Primary Output	Auto

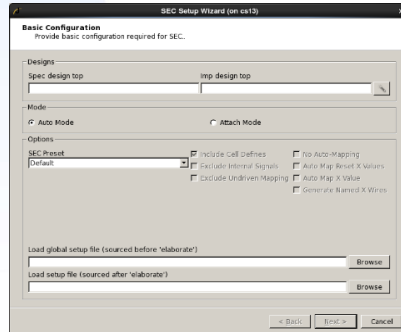
SEC App – GUI Flow

✓ SEC Setup



1. Setup Wizard

- Compile source files
- Elaborate SPEC and IMP
- Load UPF files
- Specify auto mapping options



2. Interface Check

- Performs completeness analysis of your current mapping (Optional)

3. Generate Verification

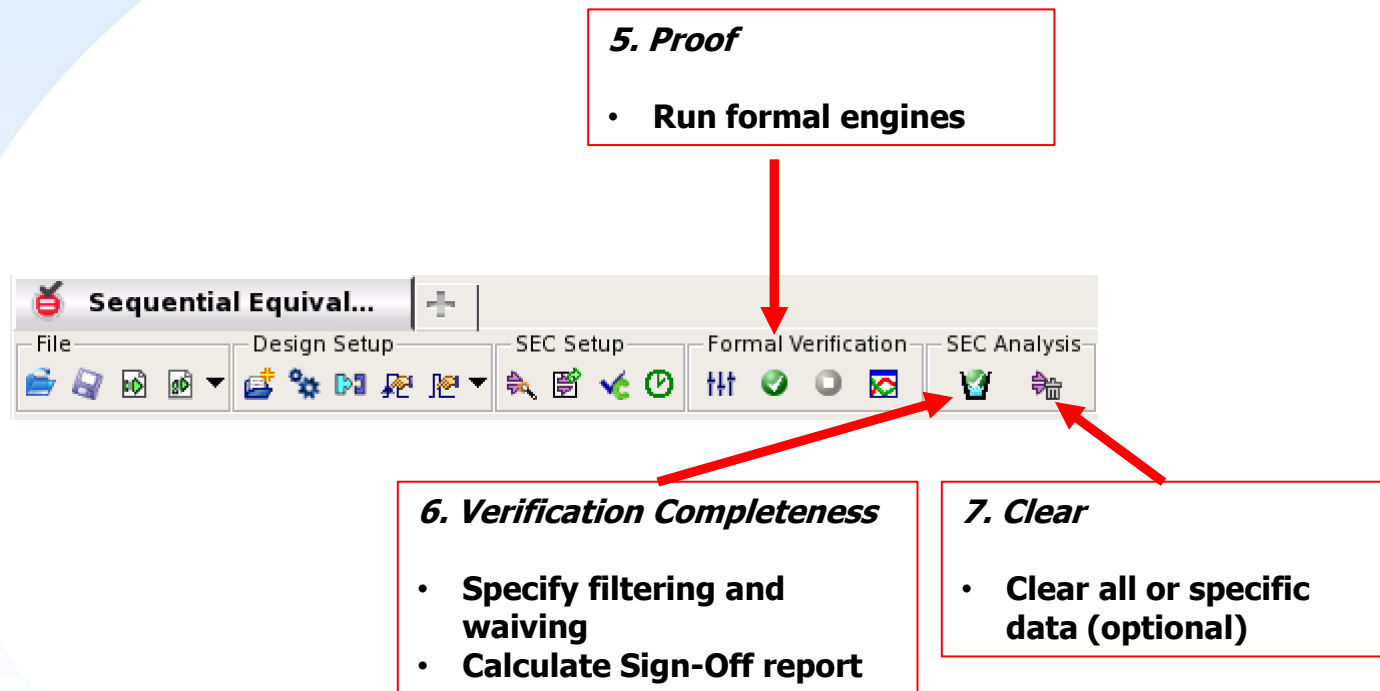
- Create assertions and assumptions for mapping pairs (Required)

4. Clock Tree Analysis

- Run clock tree analysis (Optional)

SEC App – GUI Flow

✓ SEC Proof and Analysis

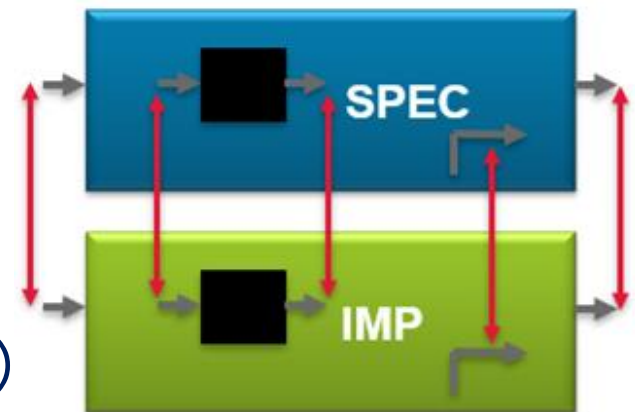


What happens during Setup

- ✓ **Build both designs**
- ✓ **Connect them into a miter model**

- ✓ **Apply auto mapping**

- Primary/DUT inputs (Connection)
- Primary/DUT outputs (Check)
- Black-box outputs (Connection)
- Black-box inputs (Check)
- Undriven or dangling signals (Connection)
- Stopats from both designs (Both)
- Uninitialized state elements initial value (Initial value connection)
- Blackbox: MPRAM ports, dividers / multipliers



SEC Setup Example Tcl Command

```
check_sec -setup \  
    -spec top_spec \  
    -spec_analyze { -sva top_spec.v } \  
    -spec_analyze { -verilog lib1.v } \  
    -spec_analyze { -vhdl lib1.vhd } \  
    -spec_elaborate_opts { -bbox_m module_a } \  
    -imp top_imp \  
    -imp_analyze { -sva top_imp.v } \  
    -imp_analyze { -verilog lib.v } \  
    -imp_analyze { -vhdl lib1.vhd } \  
    -imp_elaborate_opts { -bbox_m module_b }
```

Golden Design (spec)

Revised Design (imp)

SEC Interface Check

✓ SEC allow for automatic mapping of interface signals

check_sec –setup ... (includes implicit auto mapping)

- Input pairs will be driven to be equal → Connection
- Output pairs will be asserted to be equal → Check

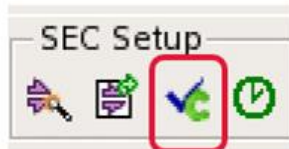
✓ Problem

- Missing input mapping may cause false failures on outputs

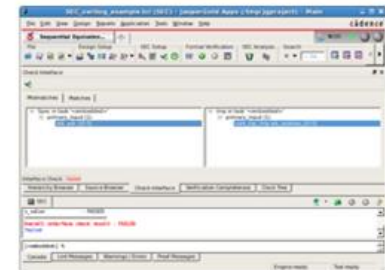
✓ Solution

- Interface check to confirm mapping is complete

Interface check helps to avoid false failures:



⇒ **check_sec – interface ...** ⇒



SEC Interface Check – Mapping Unmapped Signals in GUI

Switch to „Check Interface“ tab and display unmapped interface signals

Select SPEC signals to map

Quick Map

Select IMP signals to map

Right-Button Menu „Quick Map“ - Open Advanced Mapping Dialog

Signal expressions are pre-filled.

Create new mapping pair

Interface Check : **failed**

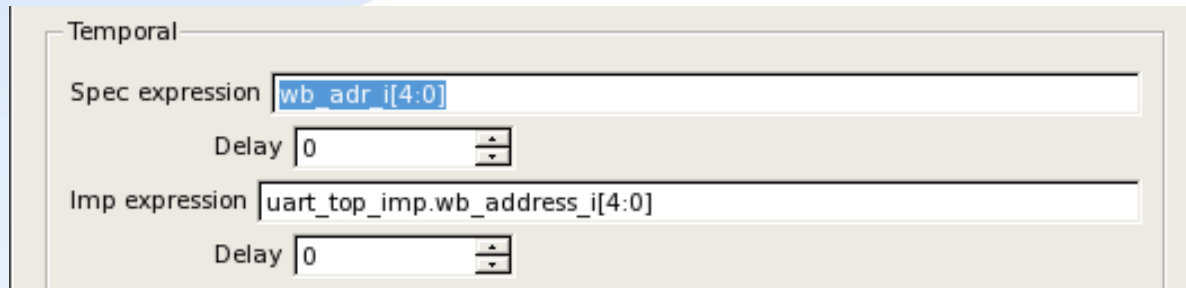
Overall interface check result - **FAILED**

failed

Map Cancel

Manual Mapping with Tcl Commands

✓ Simple mapping



Temporal

Spec expression

Delay

Imp expression

Delay

```
check_sec -map -spec {{wb_adr_i[4:0]}} -imp {{uart_top_imp.wb_address_i[4:0]}}  
-global ...
```

✓ Advanced mapping using regular expressions

Substitution

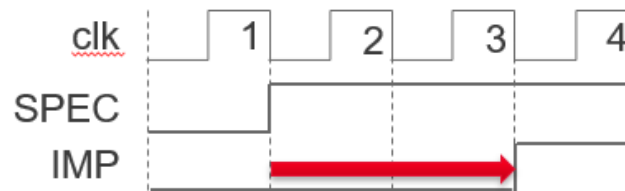
```
check_sec -map -spec {wb_adr(.*)} -imp {top_imp.wb_address$1$} -regexp -verbose
```

See http://www.tcl.tk/man/tcl8.5/TclCmd/re_syntax.htm for details

Manual Mapping with Delay

✓ SPEC and IMP mapping points are equivalent, but with 2 cycles delay

- Used for validating Pipeline Retiming for example



✓ Specify mapping with delay

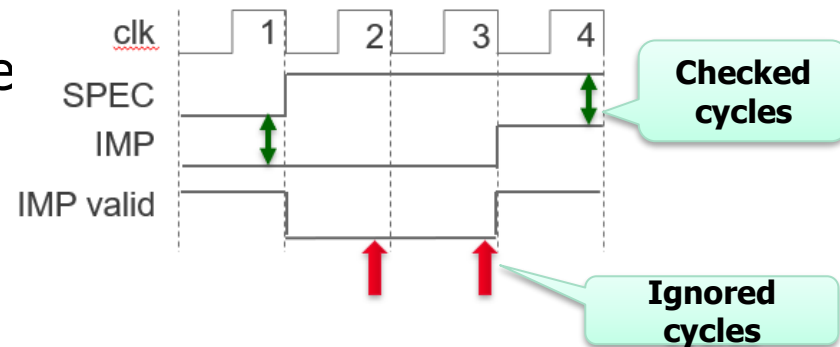
The screenshot shows a 'Temporal' specification window. It contains two sections: 'Spec expression' and 'Imp expression'. The 'Spec expression' is 'wb_adr_i[4:0]' with a 'Delay' of 0. The 'Imp expression' is 'uart_top_imp.wb_address_i[4:0]' with a 'Delay' of 2. The 'Delay' field for the Imp expression is highlighted with a blue selection box.

`check_sec -map -spec ... -imp ... -imp_delay 2`

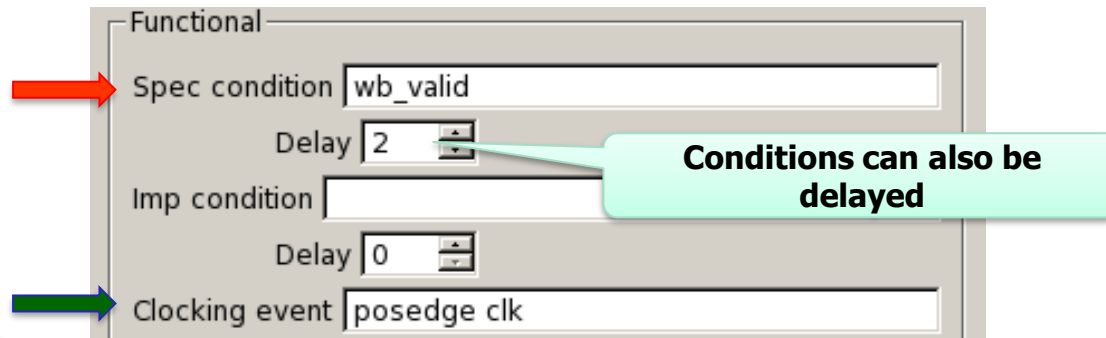
Manual Mapping with Conditions

- ✓ **SPEC and IMP mapping points are equivalent only under a condition `wb_valid`**

- Used for protocols with handshake



- ✓ **Specify mapping with condition and clock**

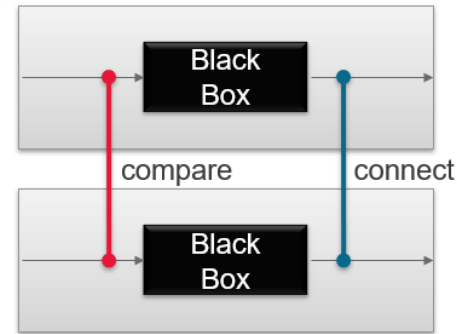


```
check_sec -map -spec ... -imp ... -spec_condition wb_valid \  
-clock {posedge clk}
```

Mapping Black-Boxes

✓ Black-Box hides functionality

- Internal logic of block is removed
- Done via the elaborate stage (-bbox_m or -bbox_i)



✓ Meanings for SEC

- Black-Box outputs are considered inputs, and need to be connected
- Black-Box inputs are considered outputs, and need to be compared (assertions)
- Black-Box ports are automatically mapped by SEC

✓ Notes

- Black-Box modules must be guaranteed to be equivalent
- If Black-Box input assertions fail, the Black-Box output connections are no longer valid
- Output connections are theoretically valid up to the cycle in which the input assertions fail

Outline

- ✓ Section 1. Introduction to SEC
- ✓ Section 2. JasperGold – SEC Setup
- ✓ **Section 3. JasperGold – Proof & Debug**
- ✓ Section 4. JasperGold – Clock Gating Verification



Generate Verification Environment



check_sec -generate

- ✓ **Generates a task with SEC properties**
- ✓ **Example messages**

```
INFO (ISEC031): Exposing connection mappings.  
INFO (ISEC021): Computing SEC environment configuration.  
INFO (ISEC032): Computing clock data.  
INFO (ISEC033): Computing reset data.  
INFO (IRS003): Setting non-resettable flop to value 0  
during reset analysis.  
INFO (ISEC034): Exposing the remaining mapping entries.  
INFO (ISEC042): Running interface check.
```

The Signal Mapping window displays a table of mapped signals. The 'Environment Is Ready' button is highlighted. Below the table, there are tabs for 'Signal Mapping', 'Property Table', and 'Hierarchy Aliasing'.

Status	ID	Spec Signal	Mapping Type	Mapping Source
✓	15	wb_ack_o	Primary Output	Auto
✓	16	int_o	Primary Output	Auto
✓	17	stx_pad_o	Primary Output	Auto
✓	18	rts_pad_o	Primary Output	Auto
✓	19	dtr_pad_o	Primary Output	Auto
✓	20	wb_dat_o[31:0]	Primary Output	Auto
✓	21	regs.transmitter.fifo...	Bbox Input	Auto
✓	22	regs.transmitter.fifo...	Bbox Input	Auto
✓	23	regs.transmitter.fifo...	Bbox Input	Auto
✓	24	regs.transmitter.fifo...	Bbox Input	Auto
✓	25	regs.transmitter.fifo...	Bbox Input	Auto
✓	26	regs.transmitter.fifo...	Bbox Input	Auto

All the mapped signal pair and their status

Assertions and Assumes



Prove

check_spec – prove

✓ **Runs formal analysis on the generate task**

✓ **Mapping pairs results**

- Proven: pair is equal
- CEX: pair is not equal
- Undetermined: prove did not finish
- Unprocessed: prove was not started

✓ **All green → Done**

✓ **Red → Debug**



Double-click to analyze failing mapping pairs in Visualize

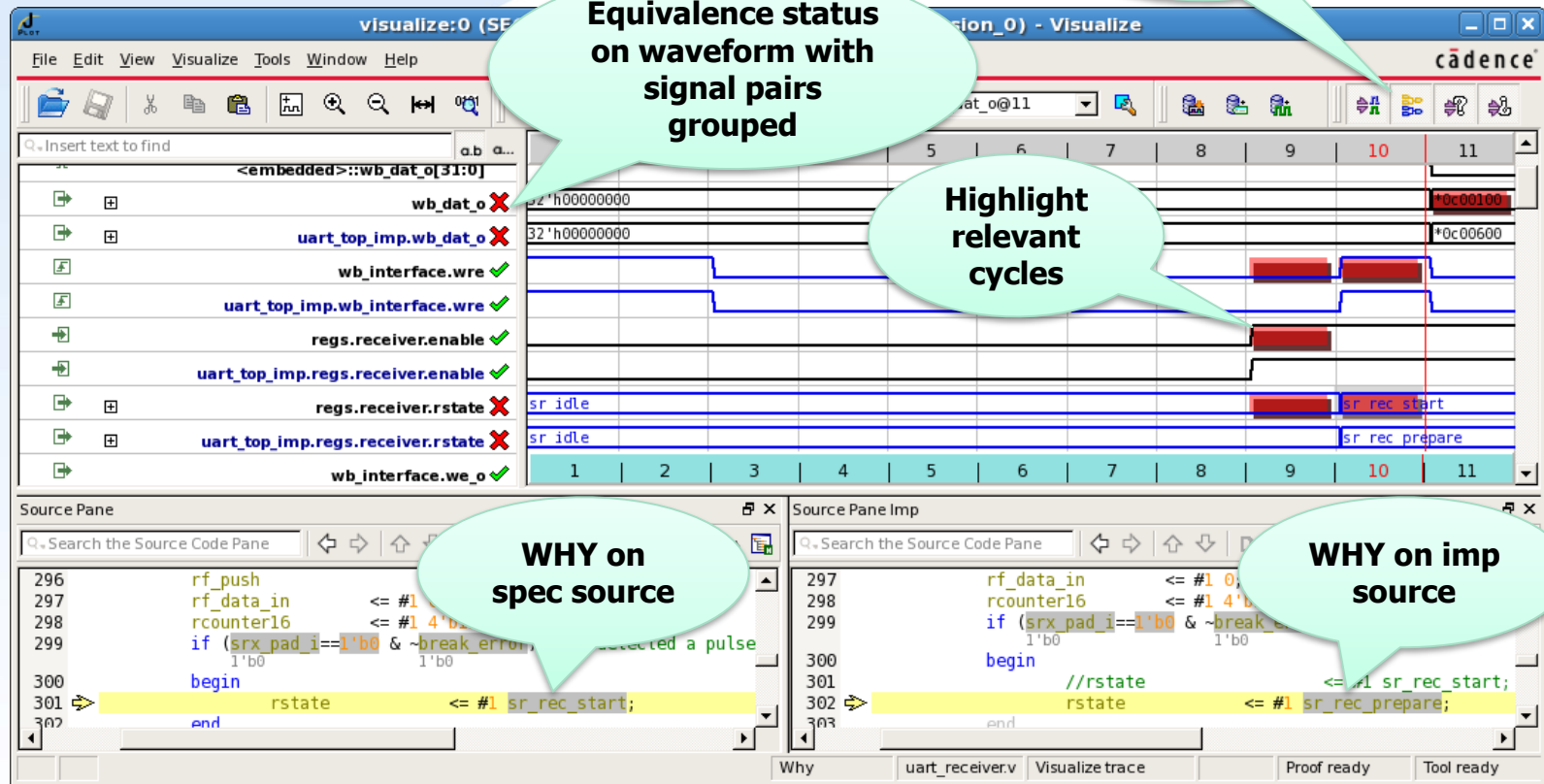
ID	Src Signal	Imp Signal	Mapping Type	Mapping
19	tx_pad_o	uart_top_imp...	Target	Auto
20	wb_ack_o	uart_top_imp...	Target	Auto
21	regs.transmit...	uart_top_imp...	Target	Auto
22	regs.transmit...	uart_top_imp...	Target	Auto
23	regs.transmit...	uart_top_imp...	Target	Auto
24	regs.transmit...	uart_top_imp...	Target	Auto
25	regs.transmit...	uart_top_imp...	Target	Auto
26	regs.transmit...	uart_top_imp...	Target	Auto
27	regs.transmit...	uart_top_imp...	Target	Auto

Pairs: 28 Rules: 28 Selected: 0

Signal Mapping Property/Task Table Hierarchy Aliasing

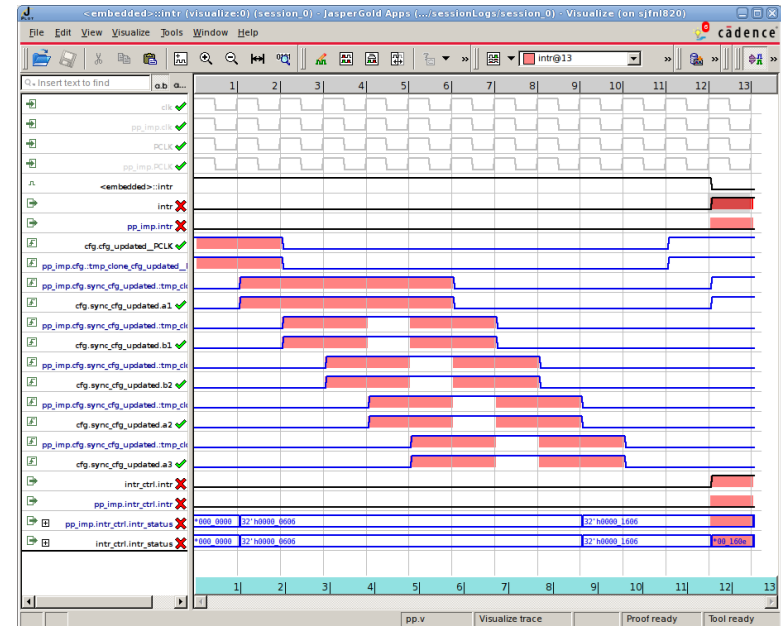
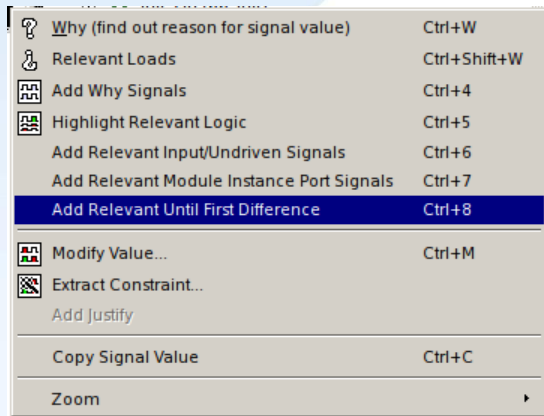
Engine ready Tool ready

Debugging CEX in SEC App



Highlight Relevant Logic Until First Difference

Unique JasperGold Debugging feature for SEC Waveforms



Also useful for traditional backtracing:

- Apply „Highlighting until first difference“
- Remove added signals again
- Perform Why-Plot Iterations

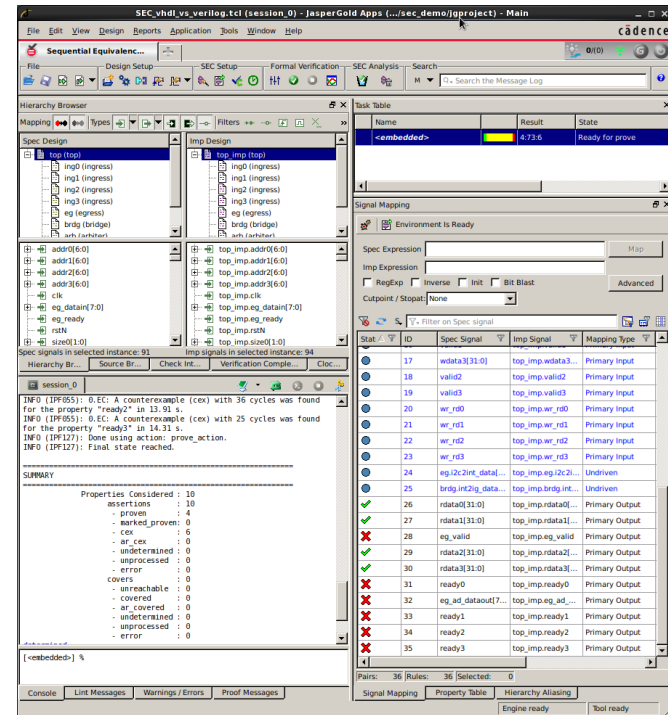
SEC Summary

✓ **Sophisticated Proof and Bug Hunting strategies provide state of the art performance**

✓ **Eliminates the need for weeks of simulation regressions after minor design changes**

✓ **Perform sequential equivalence checking on small and large sub-system blocks**

✓ **Optimized workflow GUI simplifies and speeds verification environment setup and comparative analysis and debug features using "Twin's Debug"**



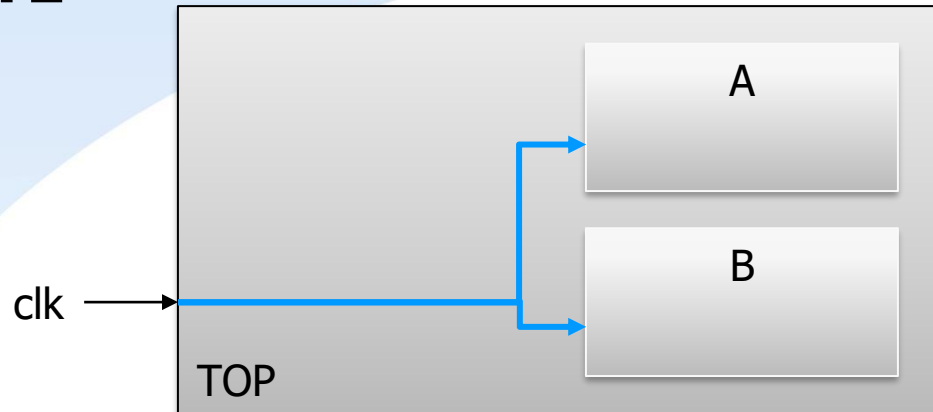
Outline

- ✓ Section 1. Introduction to SEC
- ✓ Section 2. JasperGold – SEC Setup
- ✓ Section 3. JasperGold – Proof & Debug
- ✓ **Section 4. JasperGold – Clock Gating Verification**



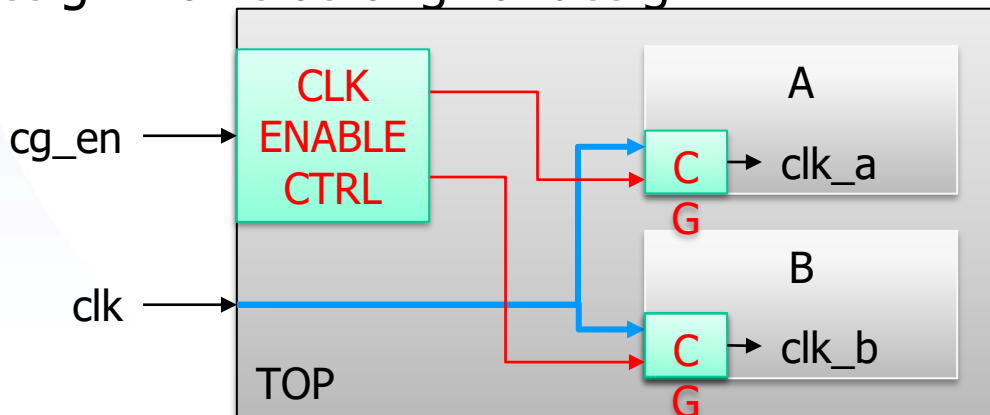
Module Level Clock Gating

✓ Original RTL



✓ Optimized RTL

- `cg_en` can enable or disable clock gating functionality. If `cg_en` is 0, design works as original design.



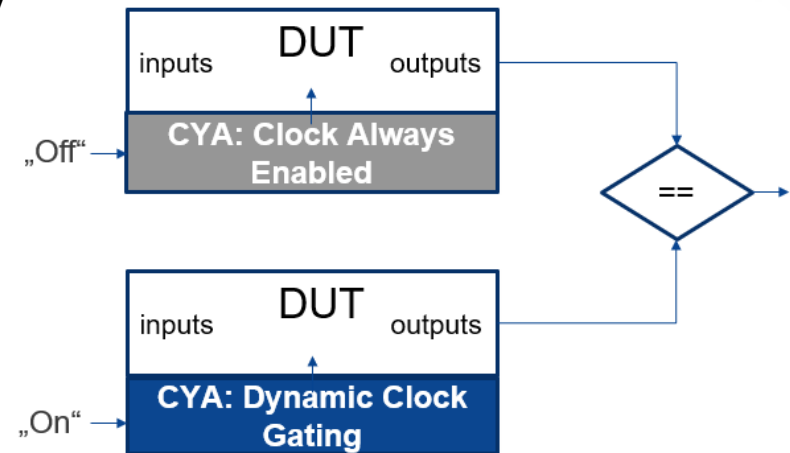
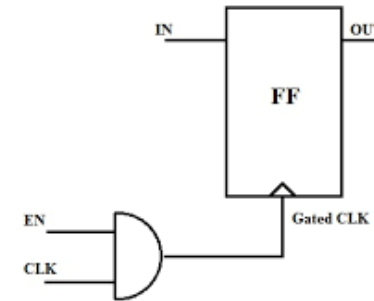
- Enable condition for entire module, group of DFF
- Can be configured globally

Clock Gating (CG) Verification

- ✓ **Phase 1: Original RTL vs. Optimized RTL (CG off)**
 - **Verify that the CG insertion logic in itself is safe**
 - “If CG is disable, is the optimized design equivalent to the original design”?
 - **Increased confidence**
 - Chip can be configured to use RTL equivalent to the original version of the RTL in case optimized RTL with CG logic enabled has undesired behavior
- ✓ **Phase 2: Original RTL vs. Optimized RTL (CG on)**
 - Verify that RTL with CG logic enabled is equivalent in functionality to the RTL with CG logic off
 - Combined with Phase 1 Pass result, this is the same as checking Original RTL vs Optimized RTL (CG On)

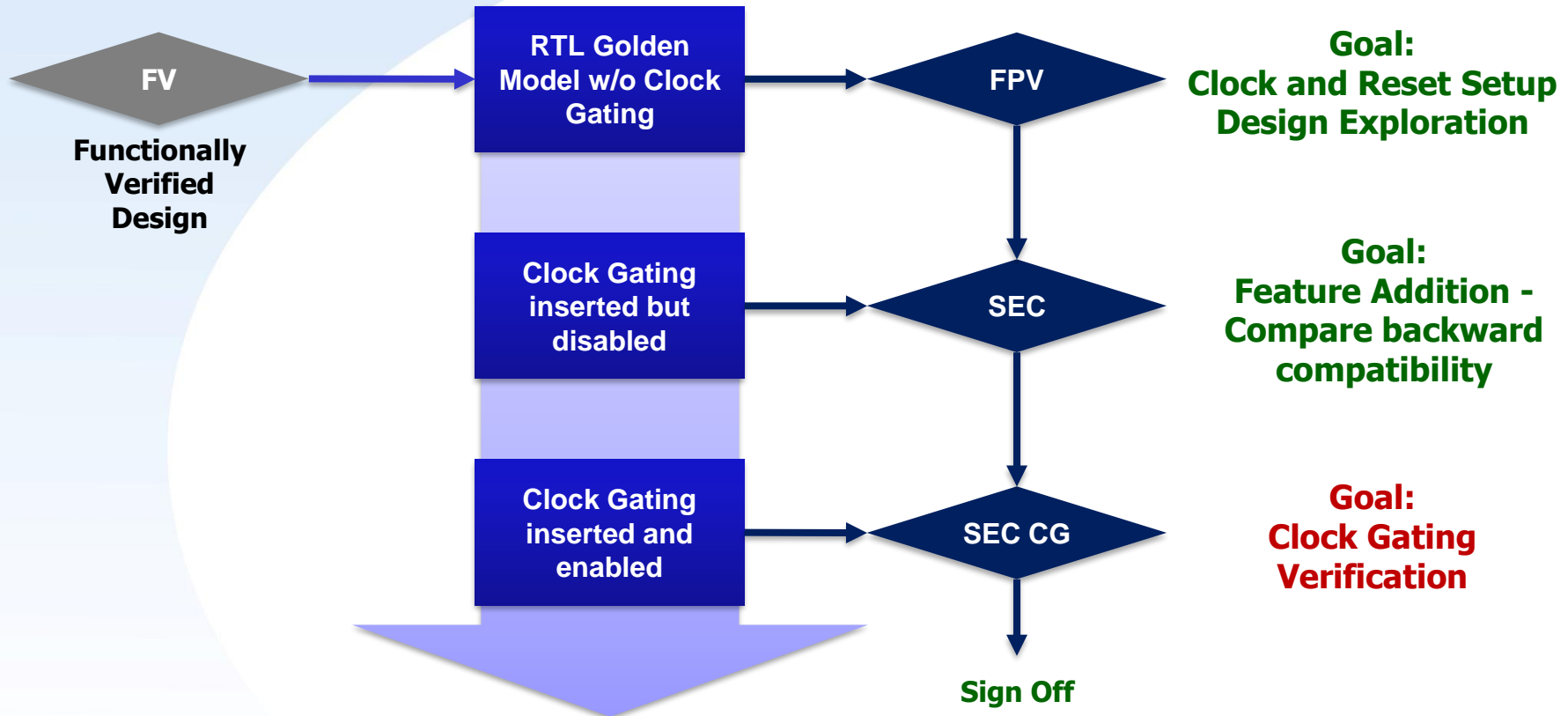
Clock Gating (CG) Verification Setup

- ✓ **IMP and SPEC are identical**
- ✓ **Design has clock gating logic**
 - Purpose is to save dynamic power consumption
 - Clock is off when not needed, enabled when needed
 - No functional difference on outputs
 - Configuration logic (CYA) introduced to disable CG, in case bug is found in Silicon
- ✓ **Goal: CG does not impact functionality**
- ✓ **Value: Extremely high**
 - Clock Gating bugs are critical
 - Usually means „lights out“ for whole chip area
 - Disabling CG means to miss advertised power specification
- ✓ **Effort: Low**
 - Easy to setup, difficult to debug
 - Constraints/conditional mappings may be needed



```
Disable clock gating in SPEC  
% assume {cg_en == OFF} -env
```

Clock Gating Verification Sign-Off Flow



Clock Gating Verification Setup

- ✓ Clock Gating Enable signal must be manipulated in SEC CG setup

- ✓ If clock gating enable signal is internal signal: Add stopats

```
stopat {clk_cfg_en imp_top.clk_cfg_en} -env
```

- disconnects RTL driver
- creates free undriven signals

- ✓ Waive clock gating enable signal pair

```
check_sec -waive -waive_signals  
{clk_cfg_en imp_top.clk_cfg_en}
```

- unmaps existing mappings
- prevents initial value mapping
- discards interface check failure

- ✓ Control clock gating in SPEC

```
assume {clk_cfg_en == 0} -env  
# assume {imp_top.clk_cfg_en == 1} -env
```

- disable clock gating in SPEC
- free clock gating in IMP



Clock Gating Verification Proof

- ✓ Proof strategy “Design Style – Clock Gating” is optimized for this problem

`set_sec_autoprove_strategy design_style`

`set_sec_autoprove_design_style_type clock_gating`

