Breaking combinational cycles

Danil Sokolov, Victor Khomenko, Alex Yakovlev

Newcastle University, UK

Introduction

- Combinational cycles
  - Common in asynchronous circuits (implement “memory”)
  - ... but they upset conventional EDA tools
- Static timing analysis (e.g. SYNOPSYS PRIMETIME)
  - Eliminates cycles by disabling some timing arcs
  - Unpredictable and suboptimal choice of timing arcs to disable
  - May remove important timing paths, e.g. critical paths
- ATPG and offline testing (e.g. SYNOPSYS TETRAMAX)
  - Limited *controlability* of signals in the combinational cycles
    i.e. inability to set signal to a specific state via primary inputs
- Need for design automation

Cycle breaking in WORKCRAFT

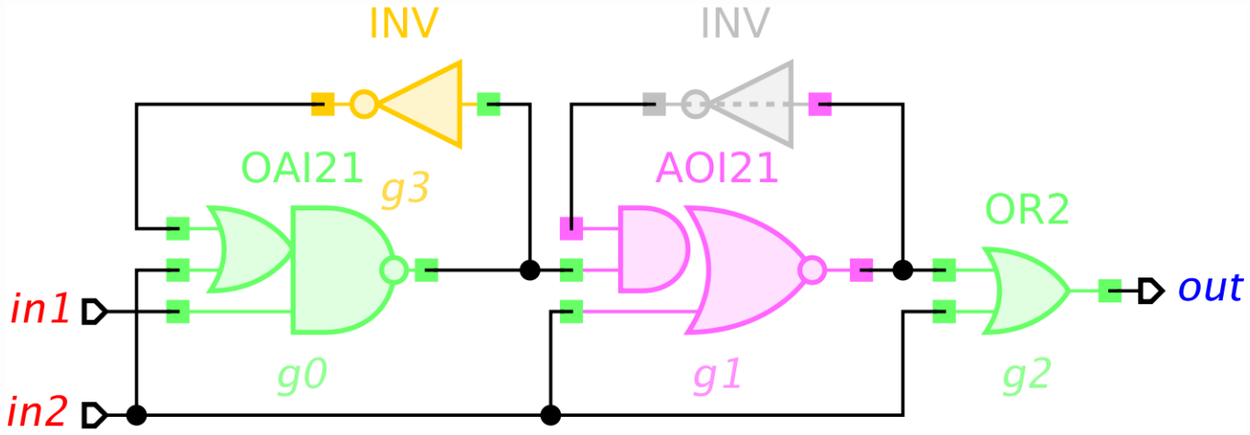
- **Path breaker** property
  - Boolean flag, unset (False) by default
  - Associated with input and output pins of circuit components
  - Used to break combinational cycles
- Input pin whose **Path breaker** property is set (True)
  - Generates `set_disable_timing` constraint to disable the timing arcs from the input pin
  - These constraints can be dumped to an SDC file for use with conventional EDA tools
  - Not always possible to break all cycles while preserving important timing paths
- Output pin whose **Path breaker** property is set (True)
  - Inserts specially implemented *testable buffer* and *testable inverters* after the output pin
  - Testable elements are designed to break cycles without removing timing paths
  - SCAN-enabled testable elements for compatibility with conventional DfT [1]

---

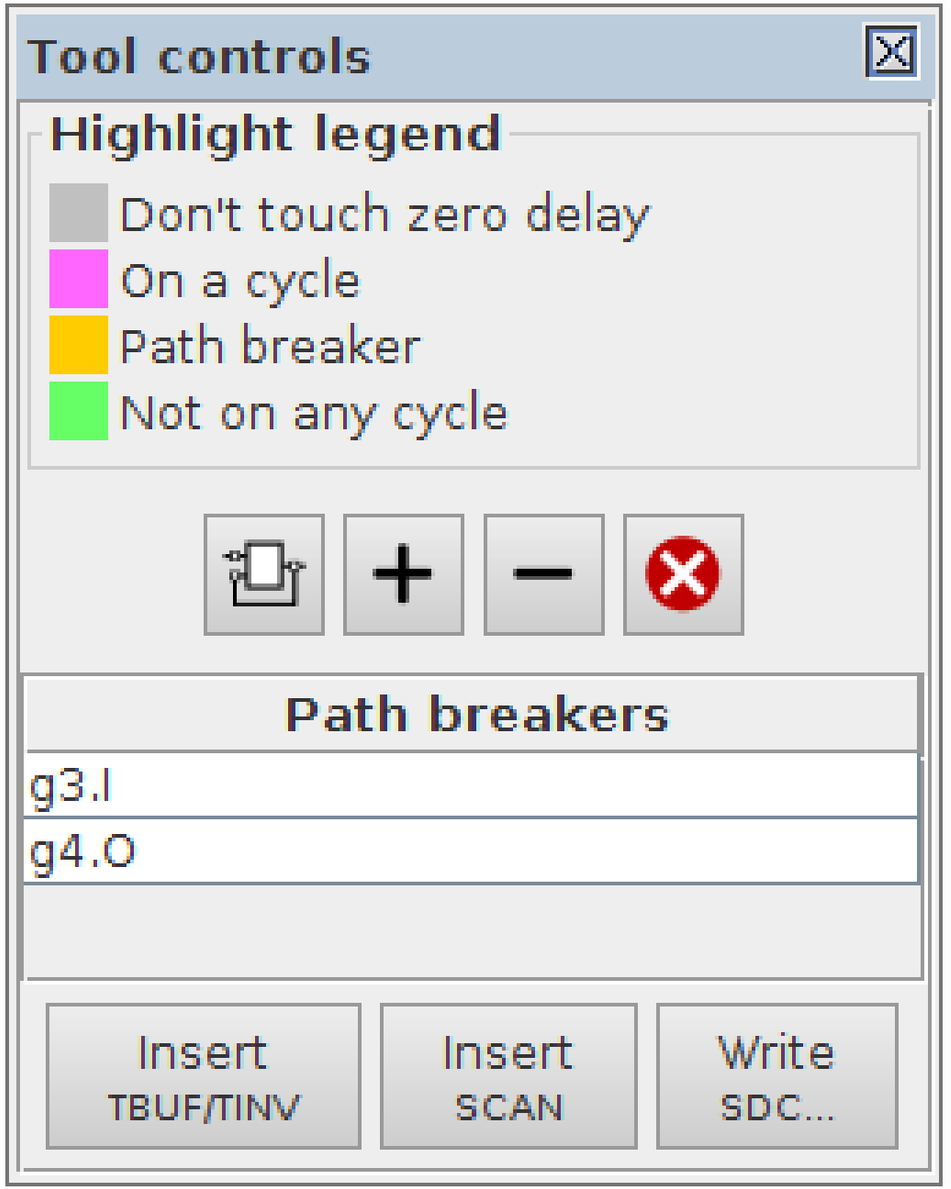
[1] D. Lloyd, R. Illman: “Scan insertion and ATPG for C-gate based asynchronous designs”, SNUG, 2014.

Cycle analyser tool

- Highlighting of gates and pins

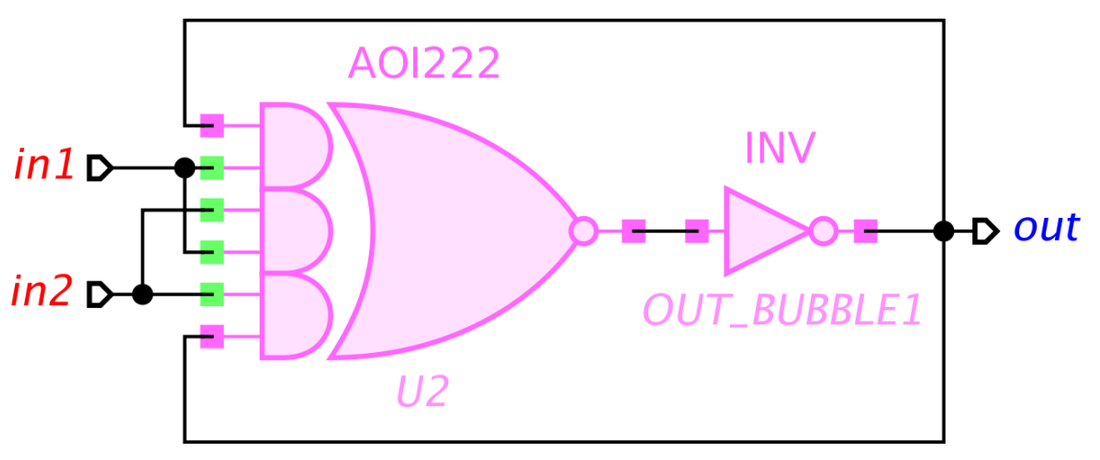


- Toggle **Path breaker (PB)** property
  - Pin click – toggle PB of the pin
  - Gate click – toggle PB of gate output pin
- Automatic loop breaking
- Insertion of testable buffers/inverters
- Insertion of SCAN chain
- Writing `set_disable_timing` constraints



Disabling timing arcs

- Demo: *celement-aoi222.circuit.work*

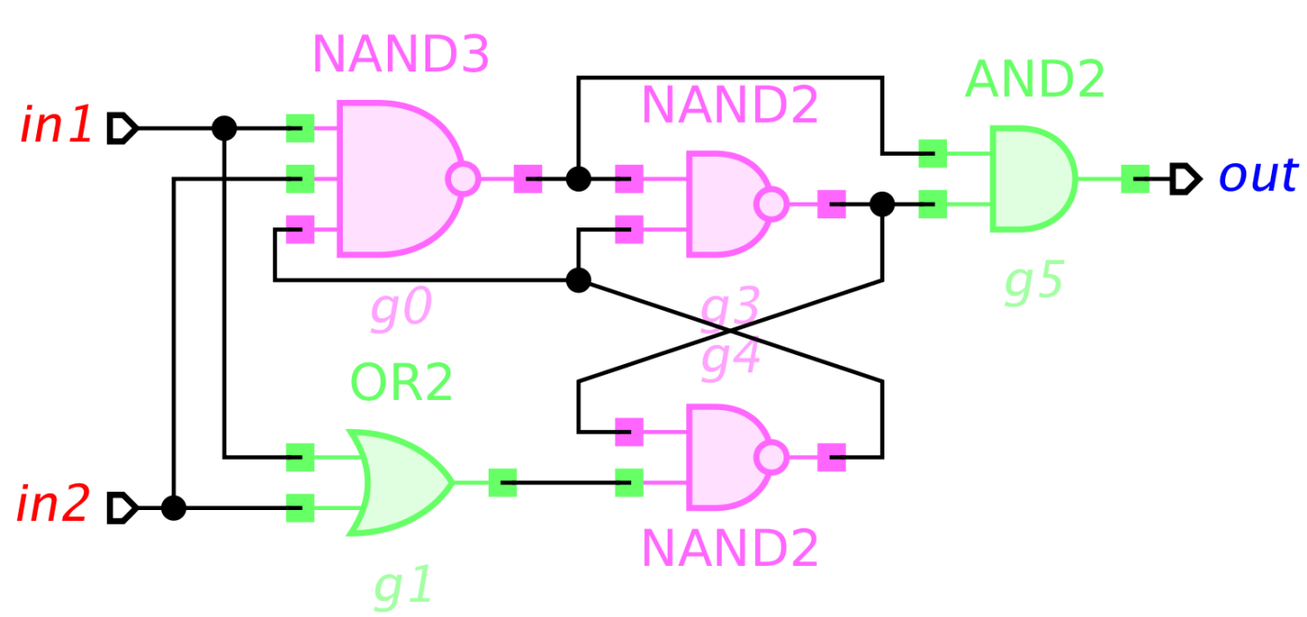


- **OUT_BUBBLE1.I** – bad path breaker as removes input-output paths (PRIMETIME choice)
- **U2.A** and **U2.F** – good pair of path breakers (under assumption that feedbacks are fast)
- **Write SDC...** button for dumping `set_disable_timing` constraints:

```
set_disable_timing U2 -from A2 -to ON
set_disable_timing U2 -from C2 -to ON
```

Disabling timing arcs: Not always possible

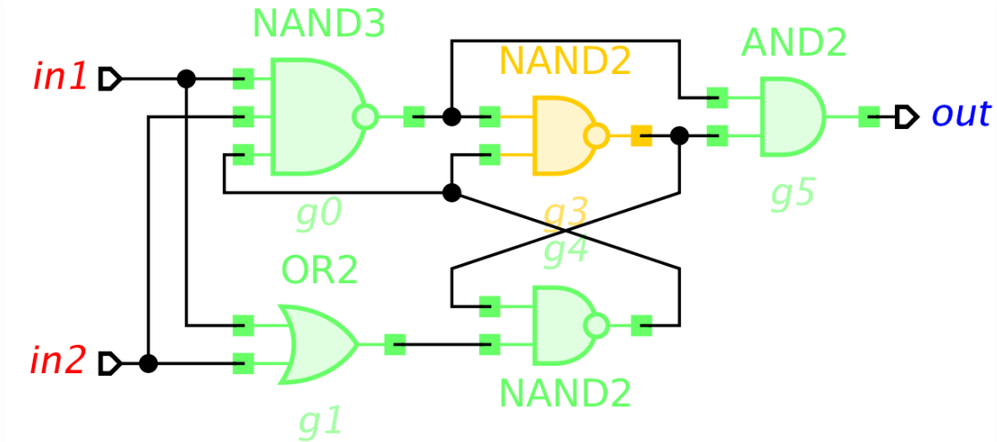
- Demo: *celement-decomposed.circuit.work*



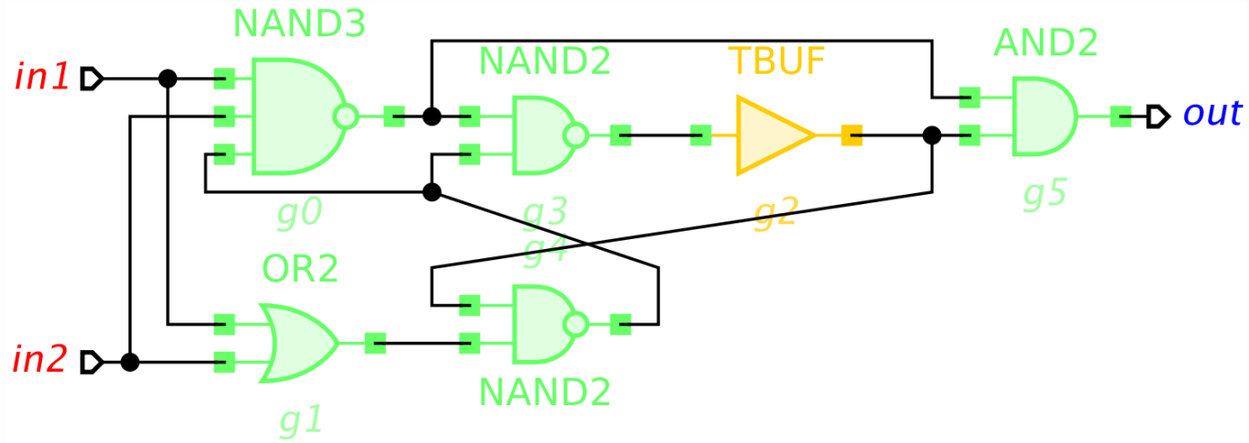
- Needs **g3.B** or **g4.A** as path breaker, but both are on critical paths:
- Rise phase: in1+, g1+, in2+, g0-, g3+, **[via g4.A]** g4-, g0+, out+
- Fall phase: in1-, in2-, g1-, g4+, **[via g3.B]** g3-, out-

Insertion of testable buffers

- Demo: *celement-decomposed-tbuf.circuit.work*
  -  – bulk **Path breaker** operations and automatic cycle breaking



  - **Insert TBUF/TINV** operation



SCAN chain

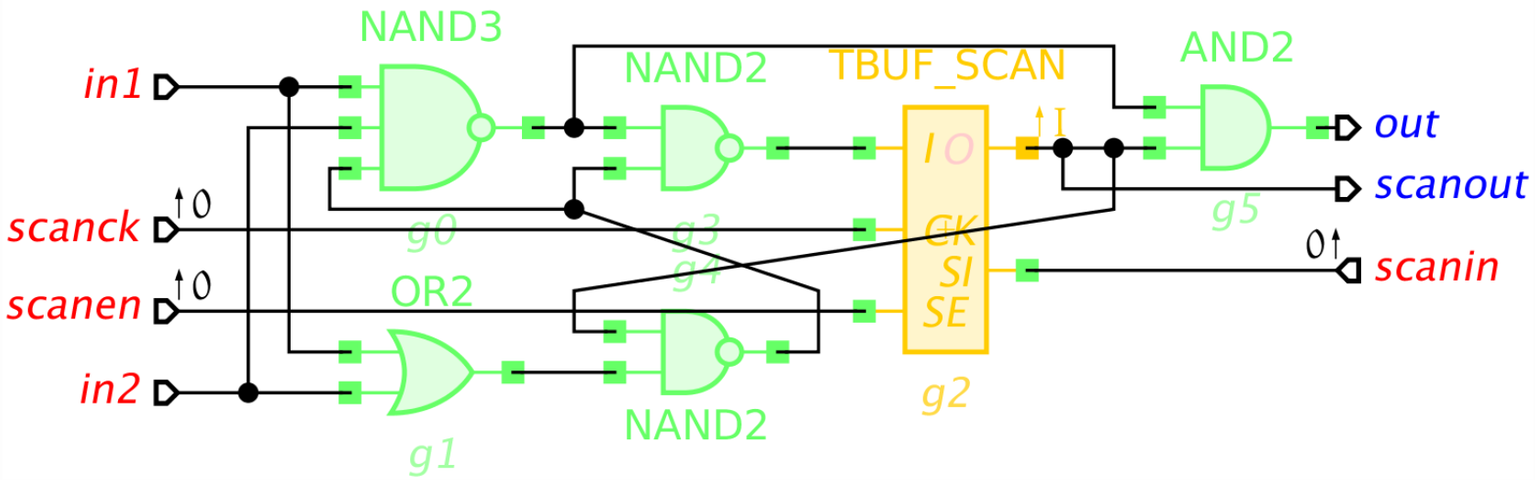
- Automatic insertion of basic SCAN chain in WORKCRAFT:

  1. SCAN ports scanck, scanen, scanin, and scanout are added
  2. Testable elements are replaced by their SCAN-enabled alternatives with additional CK, SE, and SI pins
  3. Ports scanck and scanen are connected to CK and SE pins of testable elements
  4. Testable elements are arranged in a daisy-chain between scanin and scanout ports (the order is quite random with no optimisation)

- Alternatively, after insertion of TBUF/TINV gates, a custom script can be employed for a sophisticated SCAN chain insertion

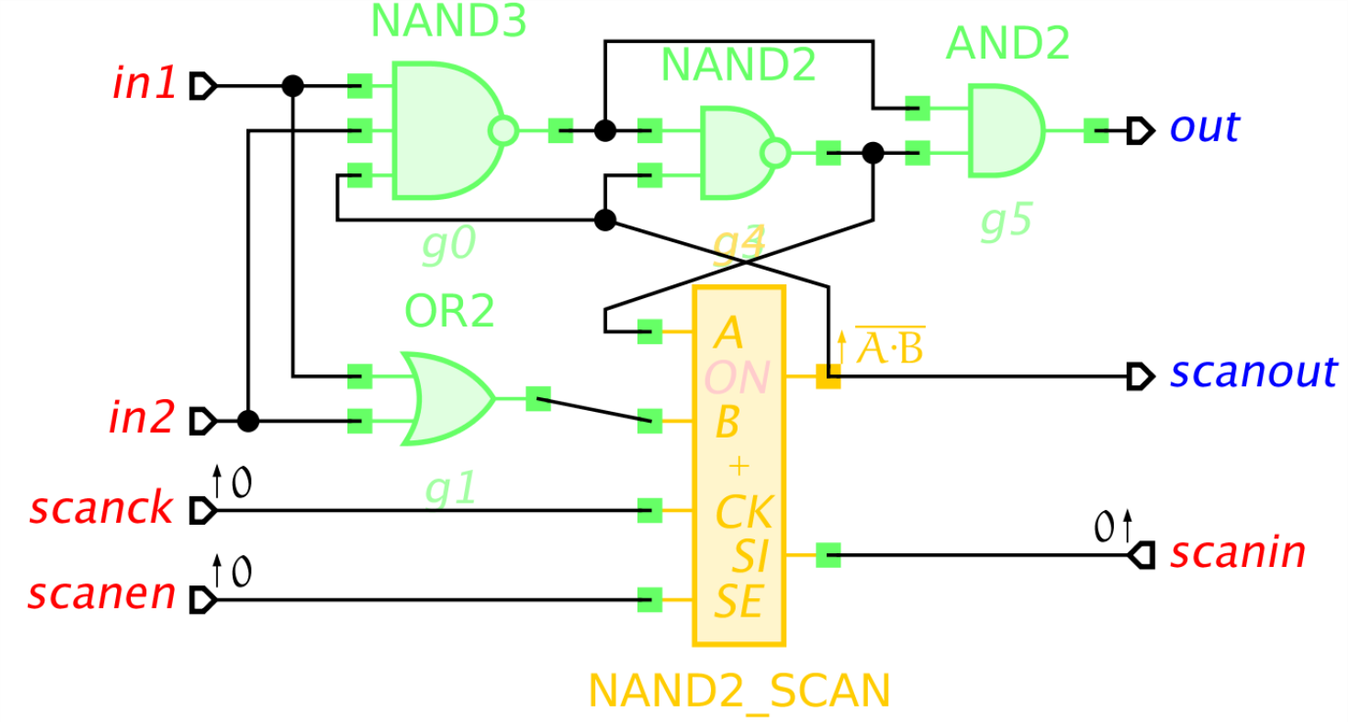
SCAN chain: “Insert SCAN” after “Insert TBUF/TINV”

- Only need SCAN-enabled implementations for TBUF (and possibly TINV)
- May impose some area and latency overheads
- Demo: *celement-decomposed-tbuf-scan.circuit.work*



SCAN chain: “Insert SCAN” without “Insert TBUF/TINV”

- Smaller area and latency
- Requires SCAN-enabled alternatives for (a subset of) library gates
- Demo: *celement-decomposed-alt-scan.circuit.work*

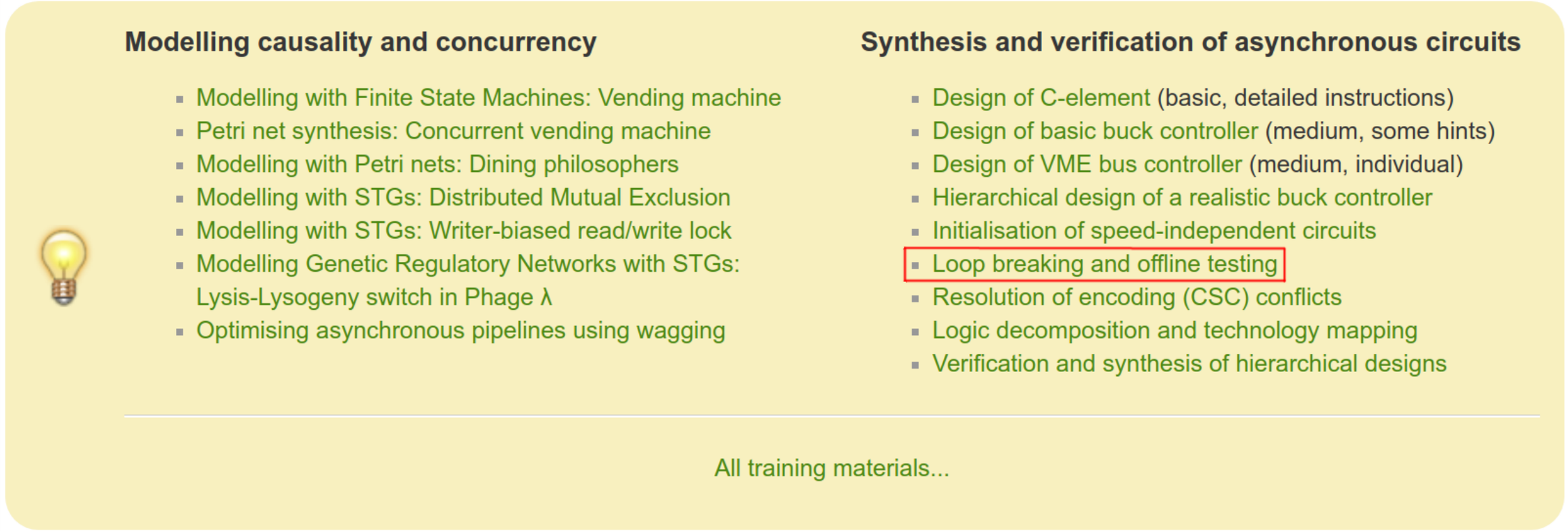


Verification

- Insertion of testable buffers and SCAN *should not break* the circuit
- Still, always verify the circuit after modification
- Use the original STG as the environment for the modified circuit
- Two warnings are expected and safe to ignore
  - Unused scanout signal (not present in the original STG)
  - Dead places associated with scanck, scanen, scanin signals (they are forced to 0 in the mission mode)

Practical: Loop breaking and offline testing

- Tutorials section at *workcraft.org*



- Direct link: *https://workcraft.org/tutorial/synthesis/loop_breaking/start*