Circuit Pattern Matching

Workflow Overview

Insight Analyzer v5.2
December 29 2021

Overview

Related topics
Adaptive circuit recognition:

- *Circuits are recognized by function, with variable patterns.*
- *Level shifters and Isolation.*
- *Memory bit cells.*
- *Latches, keepers, logic feedback loops.*
- *Current mirrors, amplifiers, bias networks, and other analog functions.*

Adaptive circuit recognition is outside the scope of this document. It is covered in other documentation for Insight Analyzer.

This document presents a workflow for literal circuit pattern matching, using Insight Analyzer.

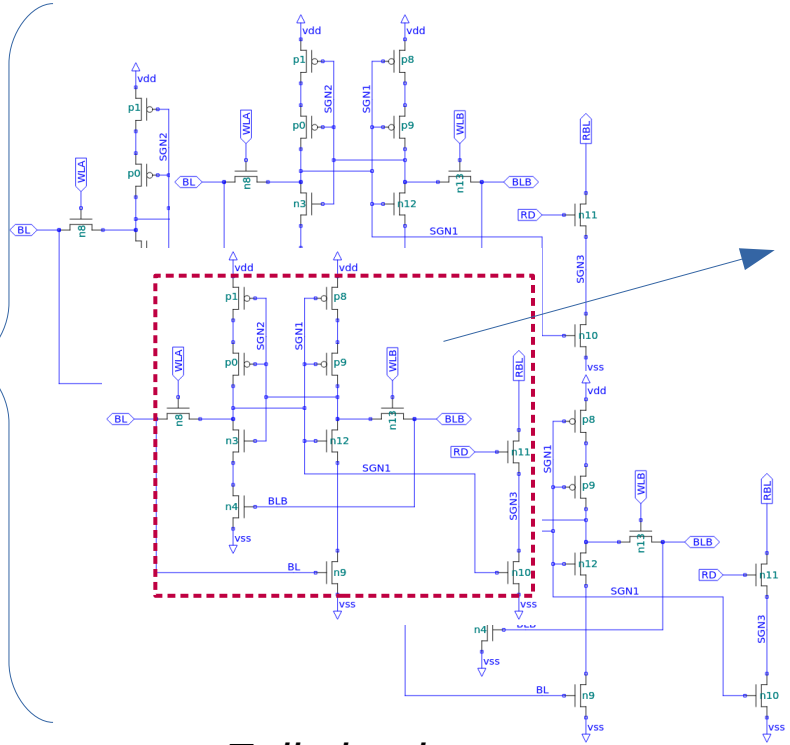
- Circuit patterns (subgraphs) are defined by user, using a netlist format.
- Found circuits have one-to-one correspondence with a defined pattern.
- Subgraph recognition helps direct circuit checking to sensitive areas.
- Subgraph recognition helps eliminate false positives from circuit checking.

Key Concepts

```
Subgraph "name" # from /XI1
AddDevice 1 Nfet # MI27
AddDevice 2 Nfet # MI28
AddDevice 3 Nfet # MI29
AddDevice 4 Nfet # MI30
AddDevice 5 Nfet # MI33
AddDevice 6 Nfet # MI34
AddDevice 7 Nfet # Mn3
AddDevice 8 Nfet # Mn4
AddDevice 9 Pfet # MI31
AddDevice 10 Pfet # MI32
AddDevice 11 Pfet # Mp0
AddDevice 12 Pfet # Mp1
# nets:
AddNet     1 I "RD"
ConnectNet 1 5-g
AddNet     2 I "WLA"
ConnectNet 2 4-g
AddNet     3 I "WLB"
ConnectNet 3 1-g
AddNet     4 B "BL"
ConnectNet 4 2-g 4-sd
AddNet     5 B "BLB"
ConnectNet 5 1-sd 8-g
AddNet     6 B "RBL"
ConnectNet 6 5-sd
AddNet     rtn
ConnectNet rtn 2-sd 6-sd 8-sd
AddNet     8 L "N_14" # (private local net)
ConnectNet 8 1-sd 10-sd 11-g 12-g 3-sd 7-g
AddNet     9 L "N_6" # (private local net)
ConnectNet 9 2-sd 3-sd
AddNet     10 L "N_12" # (private local net)
ConnectNet 10 10-g 11-sd 3-g 4-sd 6-g 7-sd 9-g
```

Applied to circuit

Text based template



Full circuit

Pattern is found during netlist loading.

Access later, during checks:

- “Transistor A belongs to pattern X”.
- Get all members, visit all locations, etc.
- Use to remove false positives.
- Use to target specific checks.

Subgraph Definition

Device declarations specify MOSFET gender, possibly with additional refinements.

Net declarations include role (in, out, bi, local), and optional reference name.

Power and ground may be defined, to help “anchor” the found subgraphs in circuit.

```
Subgraph "name" # from /XI1
AddDevice 1 Nfet # MI27
AddDevice 2 Nfet # MI28
AddDevice 3 Nfet # MI29
AddDevice 4 Nfet # MI30
AddDevice 5 Nfet # MI33
AddDevice 6 Nfet # MI34
AddDevice 7 Nfet # Mn3
AddDevice 8 Nfet # Mn4
AddDevice 9 Pfet # MI31
AddDevice 10 Pfet # MI32
AddDevice 11 Pfet # Mp0
AddDevice 12 Pfet # Mp1
# nets:
AddNet     1 I "RD"
ConnectNet 1 5-g
AddNet     2 I "WLA"
ConnectNet 2 4-g
AddNet     3 I "WLB"
ConnectNet 3 1-g
AddNet     4 B "BL"
ConnectNet 4 2-g 4-sd
AddNet     5 B "BLB"
ConnectNet 5 1-sd 8-g
AddNet     6 B "RBL"
ConnectNet 6 5-sd
AddNet     rtn
ConnectNet rtn 2-sd 6-sd 8-sd
AddNet     8 L "N_14" # (private local net)
ConnectNet 8 1-sd 10-sd 11-g 12-g 3-sd 7-g
AddNet     9 L "N_6" # (private local net)
ConnectNet 9 2-sd 3-sd
```

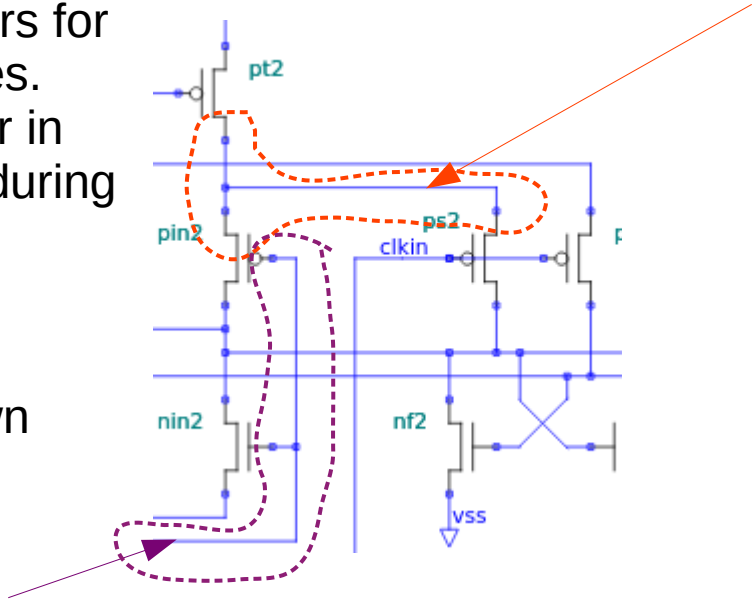
Text based template

Pin connections are allow flexibility. MOSFET source and drain may be swapped in the actual circuit.
Example: transmission gate is bi-directional.

Roles of Circuit Nodes

To optimize circuit pattern matching, templates should have appropriate indicators for the role of particular nodes. This is a significant helper in reducing compute times during recognition.

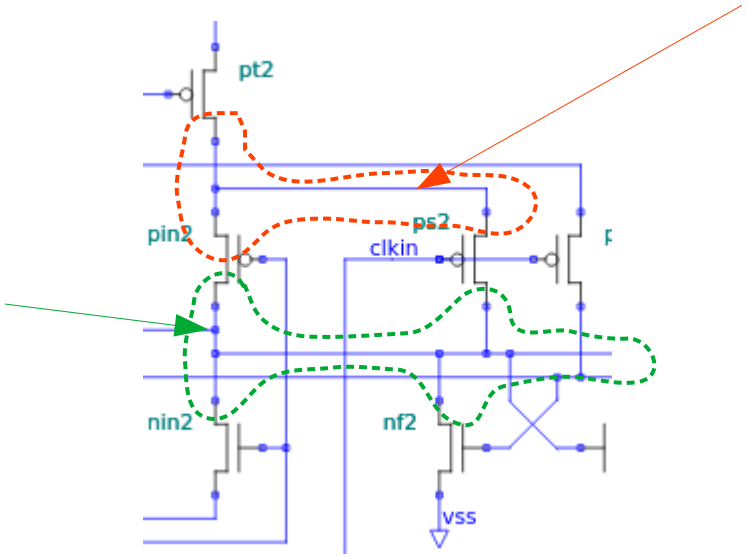
- An external node is known to connect outside of the subgraph. The template must allow this (mark node as “I”, “O”, or “B”).



- An internal node is constrained inside the subgraph. For effective matching, the template should indicate this where applicable (mark node as “L”).

Roles of Circuit Nodes

- A net with only gates is certainly an “input”. (Must not be local, and should not be output).
- A net with NMOS and PMOS source or drain should be considered output, to allow connection to other circuits.



- Usually, an internal node will have only NMOS or only PMOS source or drain. (If there is a mix of N+P, or if there are gates, I/O is implied and the net is not “local”.)

Template Generator

- To aid in the creation of subgraph templates, Insight Analyzer provides an automated utility.
- Based on an actual, example netlist, the subgraph template generator creates text based descriptions from the original circuit.
- The generator reads both single-level and multi-level cells, creating single templates per each.
- The templates are ready for use when created, but can be modified as needed, using a text editor.

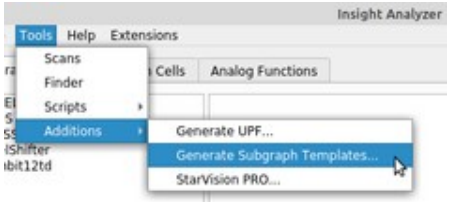
Workflow: Generate Templates

1) Load netlist in Insight Analyzer. The netlist should contain cells with dedicated functions (large flat cells will not be used to generate templates).

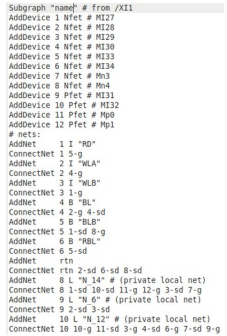
2) Define power and ground rails as normally done. This is important for generating templates.

3) Use menu Tools > Additions > Generate Subgraph Templates.

4) Edit the resultant file: Remove unwanted templates, and refine good templates with net roles (“I”, “O”, etc). Apply other options or edits as appropriate.



Menu selection



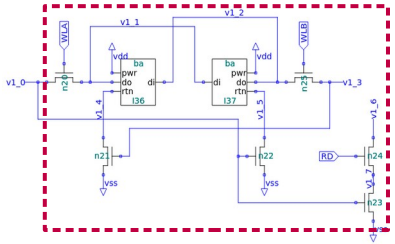
Text file in editor

Multi-Level Matching

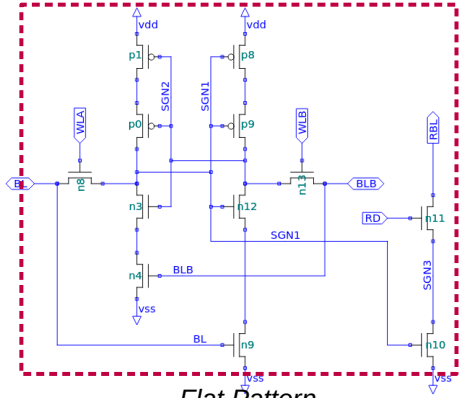
There are limitations imposed on multi-level matching. The limitations help to maintain reasonable run times while finding realistic circuit patterns. Circuit structures that exceed these limitations are probably better addressed with procedural coding.

Example: Two MOSFETs connected across distant corners of a large SoC would fall outside the realm of a “static pattern”. Such case is better found with a traditional application script.

- Insight Analyzer v5.2 supports subgraph matching through multiple levels.
- The template is always flat (all transistors listed together), similar to a netlist ‘subckt’.
- The matched subgraph may span multiple levels of hierarchy.



Multi Level Hierarchy



Flat Pattern

Workflow: Confirm Matching

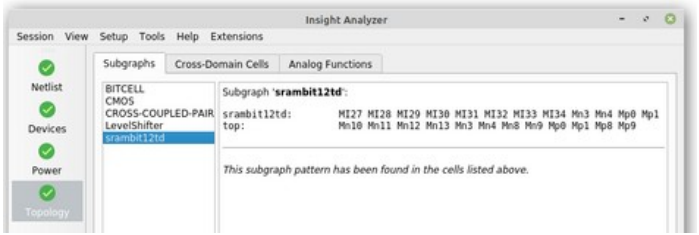
5) Save the subgraph templates file, and close the prior session of Insight Analyzer.

6) Add a command, in your run script, to invoke the new subgraph templates file:

```
project addSubgraphTemplatesFile \
"SubgraphTemplates.config"
```

7) Use your run script to re-start Insight Analyzer, and load a circuit netlist.

8) Find the subgraphs review workspace under “Topology”. Review the matched subgraphs and confirm the expected results.

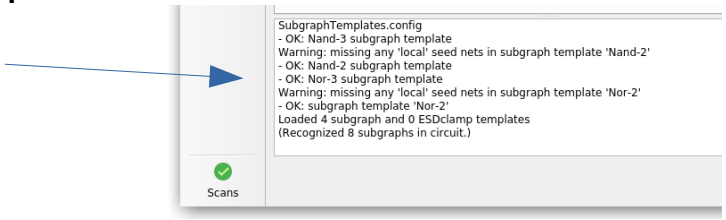


Subgraphs review workspace

Workflow: Debugging

If subgraphs are not matched as expected, there are some additional steps to help with debugging.

9) Open the main log under “View > Log”, and confirm that your subgraph templates file was found and loaded. There may be additional info about the matching process in this log.



Viewing the main log

Workflow: Debugging

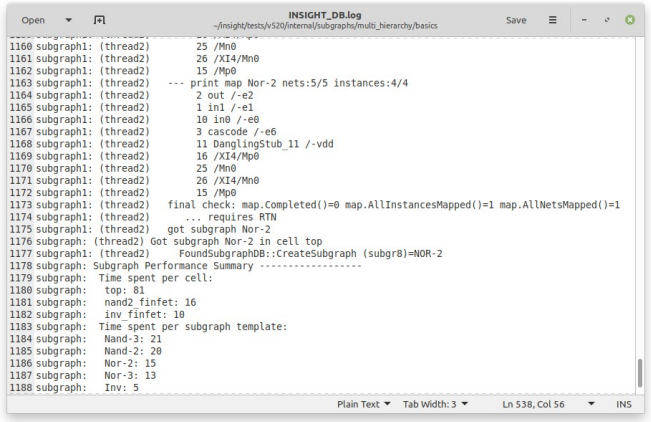
If additional debugging is needed, close the Insight Analyzer and follow the next steps.

10) In Linux, set this environment variable:
`setenv INSIGHT_DB subgraph,subgraph1`

11) Re-start Insight Analyzer as before, with the same netlist.

12) Find the debugging log file, INSIGHT_DB.log. In the file you may find: Info about the template definitions, the process of matching in each cell, and summary notes at the end.

13) If needed, you may increase log file detail by adding to the environment variable:
`setenv INSIGHT_DB …,subgraph2,subgraph3`



Detailed log file INSIGHT_DB.log

Applications

After recognition has been completed, it's time for applications (checks) that leverage the found subgraphs...

- Single-level subgraphs are easy to find in an application. Most applications are implemented at the transistor level. Transistors may be queried for their subgraphs. Example:
  `cdb inst _ getSubgraphs`
- Multi-level subgraphs are more complex than single level. These complexities should be considered at the applications stage.
- Multi-level subgraphs must be queried from the appropriate "point of reference" (POR): The upper-most cell that contains two or more children that form the subgraph is the POR.
- To invoke the POR, you must use the full hierarchy path when querying a primitive instance.
  Good: `cdb inst /x1/x2/x3/M0 …`
  Bad: `cdb inst INV:M0 …`
- Some API commands are not supported for multi-level subgraphs (next page).

Applications

Application commands for single-level and multi-level subgraphs:

Application Commands	… for Single-Level	… for Multi-Level
`cdb inst _ getSubgraphs`	supported	*
`cdb inst _ getSubgraph <name>`	supported	supported
`cdb inst _ isLogic, isTristate, isNAND, isNOR, isInverter`	supported	*
`cdb inst _ isMemberOfSubgraph <name>`	supported	supported
`cdb cell _ getChildSubgraphs`	supported	supported when in POR cell
`cdb net _ getSubgraph`	supported	*
`csubgr _ getInstances`	supported	supported
`csubgr _ getNets`	supported	*

(Not optimal for implementation)*

Example Application

This example reports the locations of each found subgraph of a specific name.

```
devparams configure / apply
pwrdefs define / apply
project addSubgraphTemplatesFile "templates.config"
csrc load “netlist.cdl”
source “application.tcl”
```

The run script

```
Subgraph "MySG"
MultiLevel yes
AddDevice 7  Pfet
AddDevice 8  Pfet
...
AddNet      2  o "out"
ConnectNet  2 7-sd  8-sd 9-sd
...
AddNet      rtn
ConnectNet  rtn 10-sd
```

Subgraph file, “templates.config”

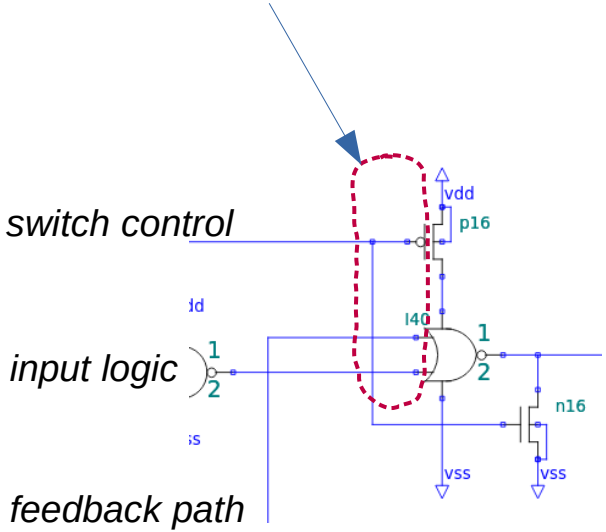
```
set outFile [ open “myReport.out” w ]
cscan cells {; # visit each cell
   foreach SG [ cdb cell %c getChildSubgraphs ] {
      if {[ csubgr $SG getName ] eq “MySG" } {
         set instanceList [ csubgr $SG getInstances -namesOnly ]
         puts $outFile “Cell %c has MySG: $instanceList”
      }
   }
}
```

The application script, “application.tcl”

Next Limitation

Logic equivalence while literal pattern is changed.

Some pattern changes may be disallowed.



- The previous pages present a workflow for literal subgraph matching.
- Literal subgraph matching has a key limitation: The matching is done on circuit connections, but can not account for logical or functional identity in a non-conforming circuit pattern.
- To handle cases such as level shifters and latches, Insight Analyzer uses adaptive matching.
- However, the user defined subgraph templates are taken literally. There is no method for a user to define allowed variations such as the ordering of inputs on a CMOS stack. Such would complicate the user's input in terms of definitions, and would be beyond the scope of the current software.

