Fall 2025-ECGR-4090/5090-Hardware Security and Trust
Dr. Fareena Saqib
Lab - Logic Locking for IP Protection
Due 11/30/2025

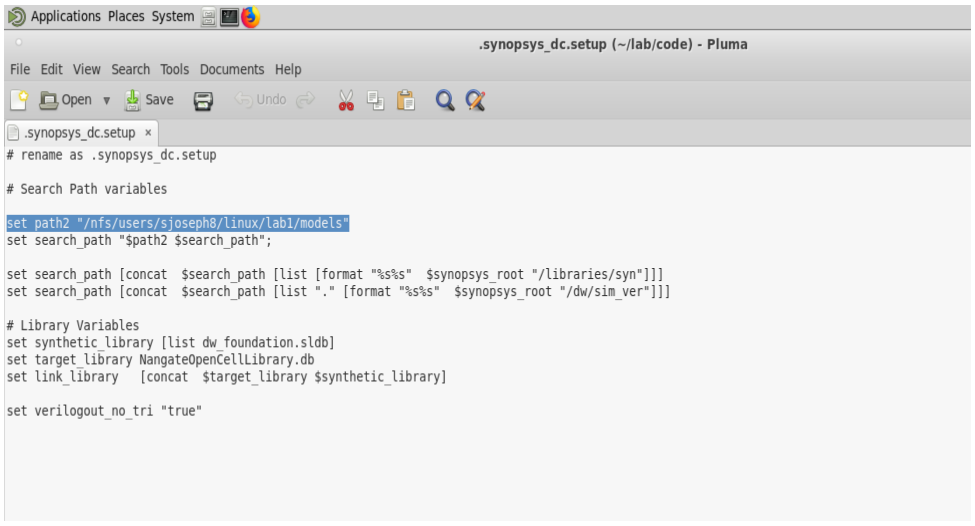
The Step 1 to step 3 are done and the results are posted on Canvas as the lab machines are not UpToDate.

Step 1 (Students are not required to complete this step as the output files are shared in the lab folder):

Login to your mosaic machine, preferably use **engr.lcs-8.uncc.edu**. Download and extract the zip from canvas home page. There should be a /code and /models directory. The /code directory should contain synthesis.tcl, .synopsys_dc.setup, c432.v and cadence.genlib file. The models directory should contain NangateOpenCellLibrary.

Step 2 (Students are not required to complete this step as the output files are shared in the lab folder):

Ensure that .synopsys_dc.setup is present in the code directory.
Open the ".synopsys_dc.setup" using a text editor. Edit the path2 variable to point to models folder in parent directory.

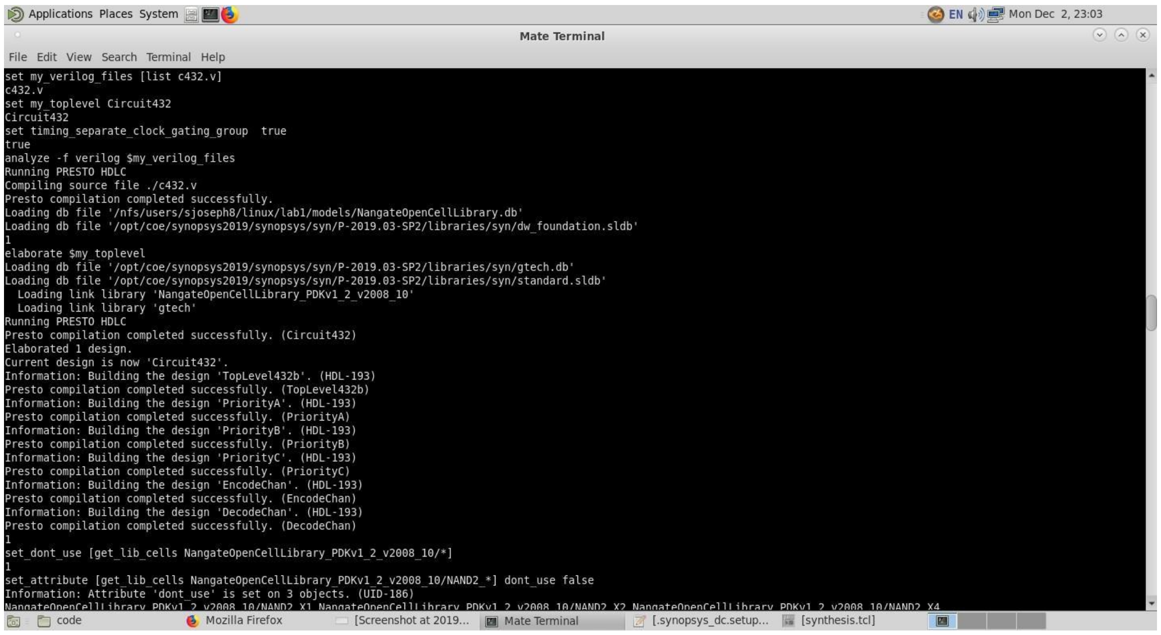

Applications Places System
.synopsys_dc.setup (~/lab/code) - Pluma
File Edit View Search Tools Documents Help
Open Save Undo
.synopsys_dc.setup

```
# rename as .synopsys_dc.setup

# Search Path variables

set path2 "/nfs/users/sjoseph8/linux/lab1/models"
set search_path "$path2 $search_path";

set search_path [concat  $search_path [list [format "%s%s"  $synopsys_root "/libraries/syn"]]]
set search_path [concat  $search_path [list "." [format "%s%s"  $synopsys_root "/dw/sim_ver"]]]

# Library Variables
set synthetic_library [list dw_foundation.sldb]
set target_library NangateOpenCellLibrary.db
set link_library   [concat  $target_library $synthetic_library]

set verilogout_no_tri "true"
```

Figure 1: Synopsys design compiler setup

Step 3 (Students are not required to complete this step as the output files are shared in the lab folder):

Open terminal in the code directory. Type “dc_shell -f synthesis.tcl”. Comments have been attached in the tcl file for better understanding. If everything is loaded properly, output should look like below.

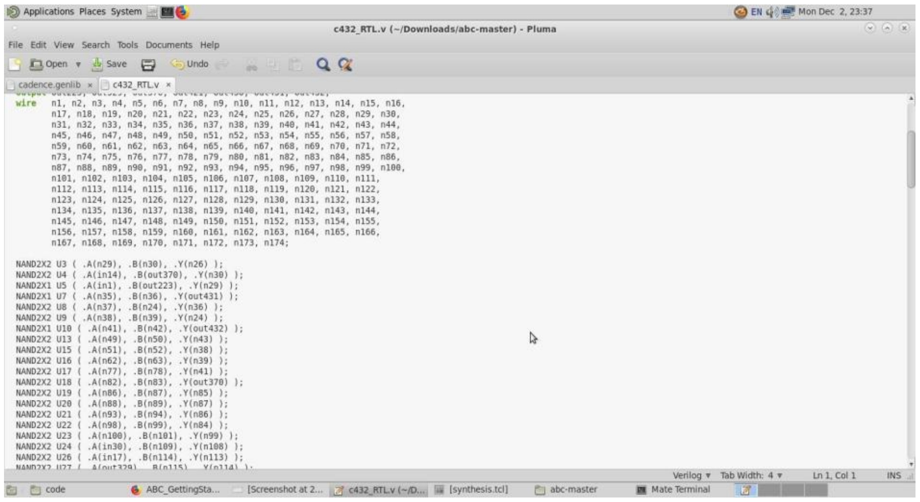


```
set my_verilog_files [list c432.v]
c432.v
set my_toplevel Circuit432
Circuit432
set timing_separate_clock_gating_group  true
true
analyze -f verilog $my_verilog_files
Running PRESTO HDLC
Compiling source file ./c432.v
Presto compilation completed successfully.
Loading db file '/nfs/users/sjoseph8/linux/lab1/models/NangateOpenCellLibrary.db'
Loading db file '/opt/coe/synopsys2019/synopsys/syn/P-2019.03-SP2/libraries/syn/dw_foundation.sldb'
1
elaborate $my_toplevel
Loading db file '/opt/coe/synopsys2019/synopsys/syn/P-2019.03-SP2/libraries/syn/gtech.db'
Loading db file '/opt/coe/synopsys2019/synopsys/syn/P-2019.03-SP2/libraries/syn/standard.sldb'
  Loading link library 'NangateOpenCellLibrary_PDKv1_2_v2008_10'
  Loading link library 'gtech'
Running PRESTO HDLC
Presto compilation completed successfully. (Circuit432)
Elaborated 1 design.
Current design is now 'Circuit432'.
Information: Building the design 'TopLevel432b'. (HDL-193)
Presto compilation completed successfully. (TopLevel432b)
Information: Building the design 'PriorityA'. (HDL-193)
Presto compilation completed successfully. (PriorityA)
Information: Building the design 'PriorityB'. (HDL-193)
Presto compilation completed successfully. (PriorityB)
Information: Building the design 'PriorityC'. (HDL-193)
Presto compilation completed successfully. (PriorityC)
Information: Building the design 'EncodeChan'. (HDL-193)
Presto compilation completed successfully. (EncodeChan)
Information: Building the design 'DecodeChan'. (HDL-193)
Presto compilation completed successfully. (DecodeChan)
1
set_dont_use [get_lib_cells NangateOpenCellLibrary_PDKv1_2_v2008_10/*]
1
set_attribute [get_lib_cells NangateOpenCellLibrary_PDKv1_2_v2008_10/NAND2_*] dont_use false
Information: Attribute 'dont_use' is set on 3 objects. (UID-186)
NangateOpenCellLibrary_PDKv1_2_v2008_10/NAND2_X1 NangateOpenCellLibrary_PDKv1_2_v2008_10/NAND2_X2 NangateOpenCellLibrary_PDKv1_2_v2008_10/NAND2_X4
```

Figure 2: Synthesize netlist

You’ll find a lot of files generated in the directory. We’ll only look at c432_RTL.v

NOTE: The output files c432_RTL.v and C17_RTL.v files are posted on canvas in “Files > Assignments > Assignment 4> Synthesized-RTL-Files”. You may start the lab from step 4.



```
wire   n1, n2, n3, n4, n5, n6, n7, n8, n9, n10, n11, n12, n13, n14, n15, n16,
       n17, n18, n19, n20, n21, n22, n23, n24, n25, n26, n27, n28, n29, n30,
       n31, n32, n33, n34, n35, n36, n37, n38, n39, n40, n41, n42, n43, n44,
       n45, n46, n47, n48, n49, n50, n51, n52, n53, n54, n55, n56, n57, n58,
       n59, n60, n61, n62, n63, n64, n65, n66, n67, n68, n69, n70, n71, n72,
       n73, n74, n75, n76, n77, n78, n79, n80, n81, n82, n83, n84, n85, n86,
       n87, n88, n89, n90, n91, n92, n93, n94, n95, n96, n97, n98, n99, n100,
       n101, n102, n103, n104, n105, n106, n107, n108, n109, n110, n111,
       n112, n113, n114, n115, n116, n117, n118, n119, n120, n121, n122,
       n123, n124, n125, n126, n127, n128, n129, n130, n131, n132, n133,
       n134, n135, n136, n137, n138, n139, n140, n141, n142, n143, n144,
       n145, n146, n147, n148, n149, n150, n151, n152, n153, n154, n155,
       n156, n157, n158, n159, n160, n161, n162, n163, n164, n165, n166,
       n167, n168, n169, n170, n171, n172, n173, n174;

NAND2X2 U3 ( .A(n29), .B(n30), .Y(n26) );
NAND2X2 U4 ( .A(in14), .B(out370), .Y(n30) );
NAND2X1 U5 ( .A(in1), .B(out223), .Y(n29) );
NAND2X1 U7 ( .A(n35), .B(n36), .Y(out431) );
NAND2X2 U8 ( .A(n37), .B(n24), .Y(n36) );
NAND2X2 U9 ( .A(n38), .B(n39), .Y(n24) );
NAND2X1 U10 ( .A(n41), .B(n42), .Y(out432) );
NAND2X2 U13 ( .A(n49), .B(n50), .Y(n43) );
NAND2X2 U15 ( .A(n51), .B(n52), .Y(n38) );
NAND2X2 U16 ( .A(n62), .B(n63), .Y(n39) );
NAND2X2 U17 ( .A(n77), .B(n78), .Y(n41) );
NAND2X2 U18 ( .A(n82), .B(n83), .Y(out370) );
NAND2X2 U19 ( .A(n86), .B(n87), .Y(n85) );
NAND2X2 U20 ( .A(n88), .B(n89), .Y(n87) );
NAND2X2 U21 ( .A(n93), .B(n94), .Y(n86) );
NAND2X2 U22 ( .A(n98), .B(n99), .Y(n84) );
NAND2X2 U23 ( .A(n100), .B(n101), .Y(n99) );
NAND2X2 U24 ( .A(in30), .B(n109), .Y(n108) );
NAND2X2 U26 ( .A(in17), .B(n114), .Y(n113) );
NAND2X2 U27 ( .A(out329), .B(n115), .Y(n114) );
```

Figure 3: RTL verilog file

Step 4:

Install the **abc** synthesis tool from Berkley. Download the entire folder “The ABC Tool” from “**Files > Assignments > Assignment 4 > The ABC Tool”.** We’ll use the **abc** tool to convert the generated Verilog file to Bench format. Bench is a format of HDL.

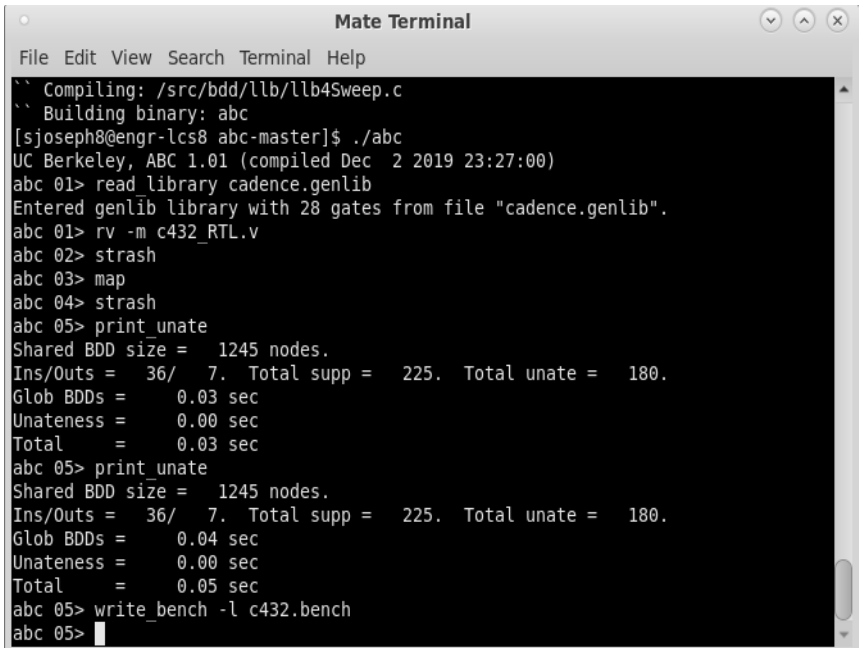
Refer to the attached **ABC_GettingStarted.pdf** for extra information pertaining to the ABC tool. **abc10216.exe will launch the abc application. {Steps 4 and Step 5 are used for the .v to .bench conversion}**

Step 5:

Enter the following commands.

read_library cadence.genlib
rv -m c432_RTL.v
strash
map
strash
write_bench -l c432.bench

Check the directory, c432.bench should be generated. This is the bench file in which you’ll be doing insertion using python script.



```
`` Compiling: /src/bdd/llb/llb4Sweep.c
`` Building binary: abc
[sjoseph8@engr-lcs8 abc-master]$ ./abc
UC Berkeley, ABC 1.01 (compiled Dec  2 2019 23:27:00)
abc 01> read_library cadence.genlib
Entered genlib library with 28 gates from file "cadence.genlib".
abc 01> rv -m c432_RTL.v
abc 02> strash
abc 03> map
abc 04> strash
abc 05> print_unate
Shared BDD size =   1245 nodes.
Ins/Outs =   36/   7.  Total supp =   225.  Total unate =   180.
Glob BDDs =     0.03 sec
Unateness =     0.00 sec
Total     =     0.03 sec
abc 05> print_unate
Shared BDD size =   1245 nodes.
Ins/Outs =   36/   7.  Total supp =   225.  Total unate =   180.
Glob BDDs =     0.04 sec
Unateness =     0.00 sec
Total     =     0.05 sec
abc 05> write_bench -l c432.bench
abc 05>
```

Figure 4:. Bench format conversion using ABC tool

Step 6:

The HOPE tool is used for the fault analysis. Review **"Fault Analysis-Based Logic Encryption"** paper in the main folder and go through the log files to understand the gates that show stuck at 0 and stuck at 1 fault.

Hope tool uses the c432.bench to generate stuck at 0 and stuck at 1 faults.

Install the **hope** tool. Download the entire folder "The HOPE Tool" from "**Files > Assignments > Assignment 4 > The HOPE Tool"** and copy the generated c432.bench file into this folder.

Step 7:

Open a command prompt and navigate to the folder of HOPE. Enter the following command:

```
hope.exe -l c432_log -D -s 9999 -r 10000 c432.bench > c432_log.out
```

Step 8:

Two files should be generated, **"c432_log.out"** and **"c432_log.txt"**. Go through the log files and understand the outputs.

Step 9:

Write your python script that prompts the user for the number of gates to be added. Your code may suggest an optimal number of key gates for insertion.

Read the paper to suggest a key insertion scheme of your design. Discuss your key insertion algorithm in your report.

Step 10:

Evaluate the output of your design with correct and incorrect key configuration.

Write the pre key insertion testbench and post key insertion testbench to evaluate the correctness of the logic locked design on FPGA

Project Directions:

1. Write a python script to perform the key gate insertion.
2. Read the paper for the fault based key insertion scheme we discussed in class. In your report discuss your key insertion scheme. Verify your design with correct key to the key gates and discuss the number of bitflips with the wrong key combinations.
3. Evaluate the security of the design, and discuss vulnerabilities associated to the selected key insertion scheme.

Project deliverables:

1. Report with steps and snapshots of the above-mentioned steps.
2. Details of the scheme for key insertion.
3. Code developed.
4. Testbench to validate the design with correct key insertions.
5. Impact of key insertions on the output with wrong key combinations.
6. Pre and post key insertion FPGA testing using testbench or optional Board testing