IBEX based SoC - MILESTONE 2

Ali Ghazal	900171722
Samah Ayman	900171848
Israa Fahmy	900171831
Sarah Ali	900172348

1. **PLAN:**

   STEP 1: Collecting the needed tools. More on this in the tools sections **[DONE]**
   STEP 2: Exploring those tools **[undergoing]**
   STEP 3: Understanding the SoC architecture **[undergoing]**
   STEP 4: Searching for the needed missing modules **[undergoing]**
   STEP 5: Connecting everything together **[undergoing]**

   STEP 6: OPENLANE FLOW **[undergoing]**
   STEP 7: OPENLANE EXPLORATION
   STEP 8: Interfacing IBEX core with AHB bus **[undergoing]**
   STEP 9: Interfacing IBEX core with the reset of the SocGen RTL-generated modules
   STEP 10: Final Simulation of the SoC on Caravel Chip

   Between all steps, we are doing verification whenever possible

2. **Tools:**

SoC GEN	Generating components for the SoC.
OPENLANE	Automated RTL to GDS-II flow based on OpenROAD, Yosys, Magic, Netgen, Fault and custom methodology scripts for design exploration and optimization.
IBEX	The main CPU component.
SV2V	Transforming from IBEX' SystemVerilog to verilog to be compatible with Yosys.

3. **Progress:**

   SoC GEN:

   1. We read the slides and watched the tutorials posted on github
   2. We experimented with the basic flow of the program
      a. By modifying the JSON format of IPs, Masters ...
      b. By trying to understand the hierarchy of the output and functionality of each module
   3. We produced the N5 demo project and tried hardening it using open lane

   IBEX:
   1. We read its documentation available on GitHub.
   2. We explored the RTL models of IBEX core, the integration module, and

tried to formulate a good understanding of the integration module parameters and interfaces.

3. We used the default parameters for the integration module during our synthesis (default parameter for the register file: RegFileFF, for example). In order to do so:
    - Firstly we converted the SystemVerilog RTLs to Verilog in order to be compiled by Yosys.
    - Secondly, we ran OpenLANE on the Verilog RTLs of IBEX core for the synthesis step.
    - We got the timing analysis from OpenSTA
    - The point at which the synthesis fails is the OpenROAD step.

4. Problems (solved)

- Yosys does not completely support SystemVerilog so we needed to convert from SystemVerilog to Verilog using SV2V.
- We did this using sv2v .. to automate the tool and account for the dependencies we wrote a bash script based on ibex_yosys_syn:

```
for file in ../rtl/*.sv; do
  module=`basename -s .sv $file`
  sv2v \
    --define=SYNTHESIS \
    ../rtl/*_pkg.sv \
    -I../vendor/lowrisc_ip/prim/rtl \
    $file \
    $LR_SYNTH_OUT_DIR/generated/${module}.v
done
```

  This generated the needed Verilog files.

  And we tried synthesizing those files with yosys.
- Script for the config.tcl for IBEX:

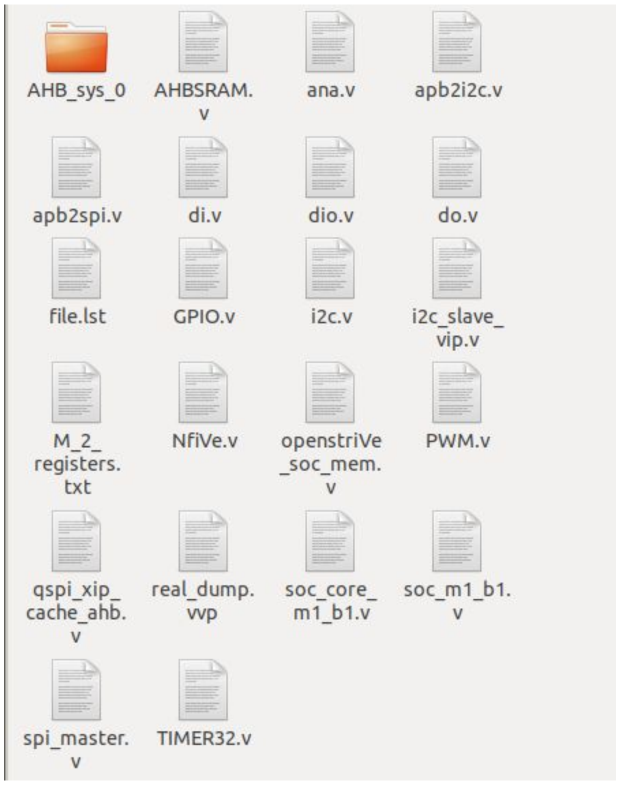
```
# Design
set ::env(DESIGN_NAME) soc_m1_b1

set ::env(VERILOG_FILES) "[glob $::env(DESIGN_DIR)/src/*.v]
[glob $::env(DESIGN_DIR)/src/*/*.v] [glob
$::env(DESIGN_DIR)/src/*/*/*.v]"

set ::env(CLOCK_PERIOD) "14"
set ::env(CLOCK_PORT) "HCLK"
set ::env(CLOCK_NET) $::env(CLOCK_PORT)


set filename
$::env(DESIGN_NAME)/$::env(PDK)_$::env(STD_CELL_LIBRARY)_config
.tcl
if { [file exists $filename] == 1} {
      source $filename
}
```

5. **Problems (undergoing)**
    - During running OpenLANE, synthesis is failing as a result of combinational network error; the produced RTL models are combinational and they need to be logical instead to be synthesized.
    - Information available regarding the field contents of the AHB bus is not sufficient. This is needed for designing the interface between the core and the bus generated by SoCGen.

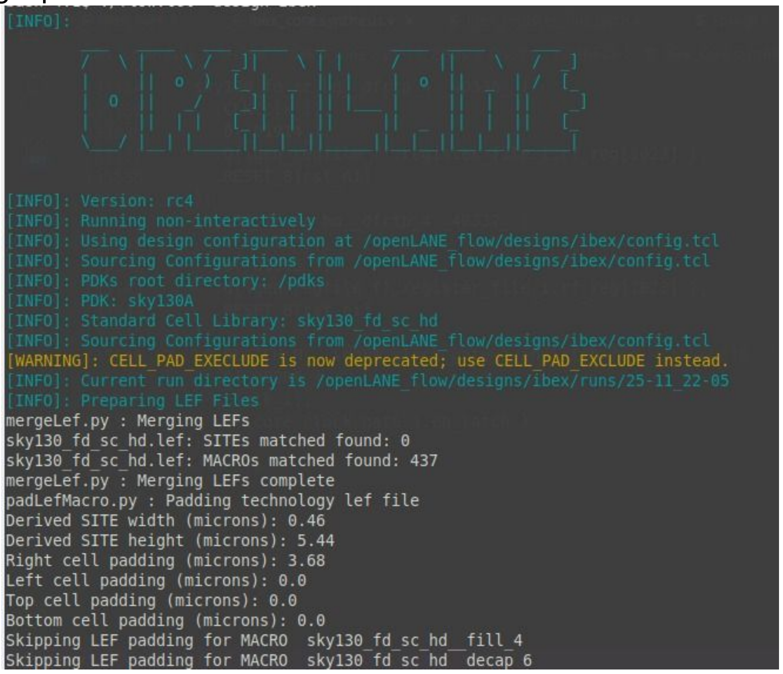
6. **Next step**
    - Understanding the SoC modules generated by SoCGen
    - Designing the required interfaces between IBEX Core and the SoCGen modules
    - Fixing the bugs occurring during hardening the IBEX core using OpenLANE (errors with OpenROAD step)

7. Outputs

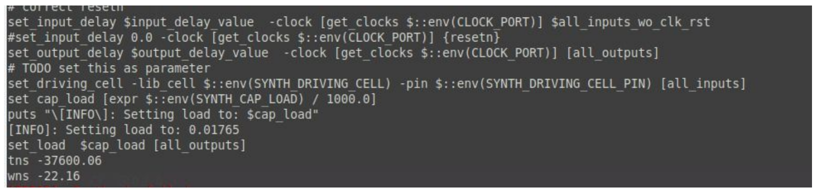
SoCGen output files



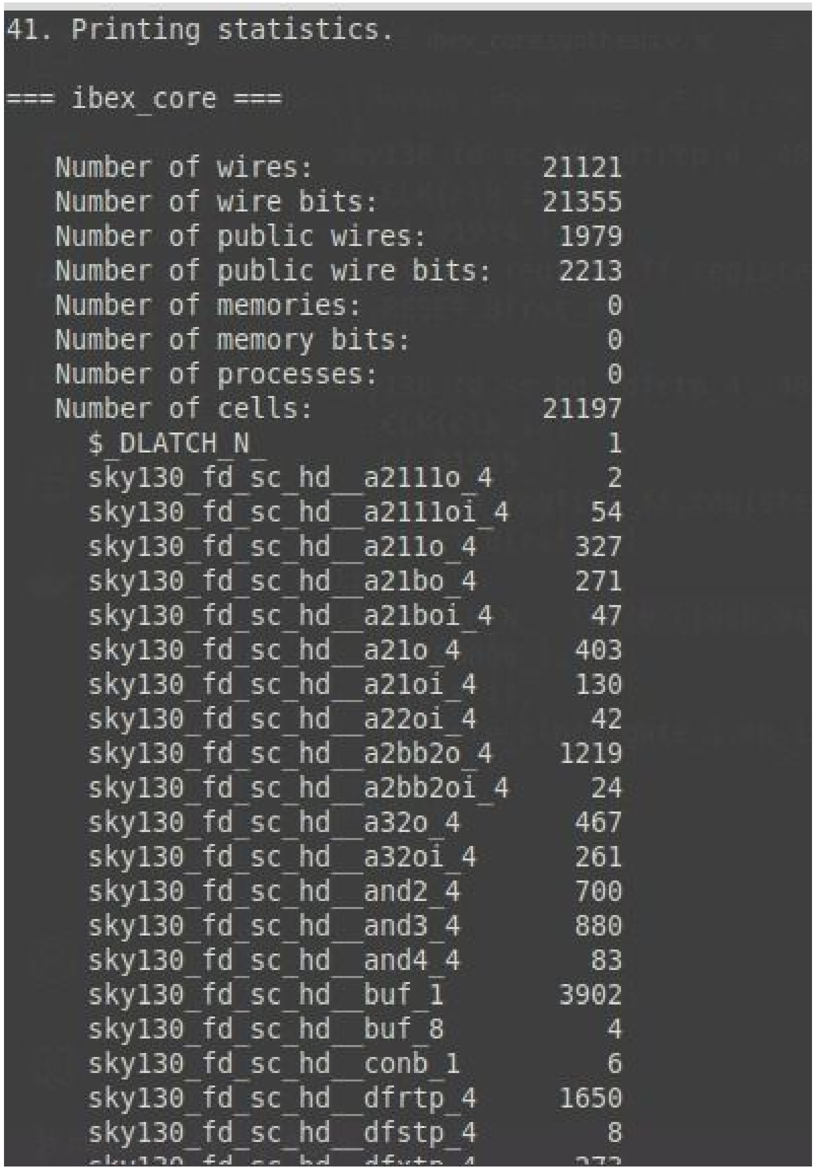
Running OpenLANE



```
[INFO]:

[INFO]: Version: rc4
[INFO]: Running non-interactively
[INFO]: Using design configuration at /openLANE_flow/designs/ibex/config.tcl
[INFO]: Sourcing Configurations from /openLANE_flow/designs/ibex/config.tcl
[INFO]: PDKs root directory: /pdks
[INFO]: PDK: sky130A
[INFO]: Standard Cell Library: sky130_fd_sc_hd
[INFO]: Sourcing Configurations from /openLANE_flow/designs/ibex/config.tcl
[WARNING]: CELL_PAD_EXECLUDE is now deprecated; use CELL_PAD_EXCLUDE instead.
[INFO]: Current run directory is /openLANE_flow/designs/ibex/runs/25-11_22-05
[INFO]: Preparing LEF Files
mergeLef.py : Merging LEFs
sky130_fd_sc_hd.lef: SITEs matched found: 0
sky130_fd_sc_hd.lef: MACROs matched found: 437
mergeLef.py : Merging LEFs complete
padLefMacro.py : Padding technology lef file
Derived SITE width (microns): 0.46
Derived SITE height (microns): 5.44
Right cell padding (microns): 3.68
Left cell padding (microns): 0.0
Top cell padding (microns): 0.0
Bottom cell padding (microns): 0.0
Skipping LEF padding for MACRO  sky130_fd_sc_hd__fill_4
Skipping LEF padding for MACRO  sky130_fd_sc_hd__decap_6
```



```
# correct resetn
set_input_delay $input_delay_value  -clock [get_clocks $::env(CLOCK_PORT)] $all_inputs_wo_clk_rst
#set_input_delay 0.0 -clock [get_clocks $::env(CLOCK_PORT)] {resetn}
set_output_delay $output_delay_value  -clock [get_clocks $::env(CLOCK_PORT)] [all_outputs]
# TODO set this as parameter
set_driving_cell -lib_cell $::env(SYNTH_DRIVING_CELL) -pin $::env(SYNTH_DRIVING_CELL_PIN) [all_inputs]
set cap_load [expr $::env(SYNTH_CAP_LOAD) / 1000.0]
puts "\[INFO\]: Setting load to: $cap_load"
[INFO]: Setting load to: 0.01765
set_load  $cap_load [all_outputs]
tns -37600.06
wns -22.16
```



```
41. Printing statistics.

=== ibex_core ===

   Number of wires:              21121
   Number of wire bits:          21355
   Number of public wires:        1979
   Number of public wire bits:    2213
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:              21197
     $_DLATCH_N_                     1
     sky130_fd_sc_hd__a2111o_4       2
     sky130_fd_sc_hd__a2111oi_4     54
     sky130_fd_sc_hd__a211o_4      327
     sky130_fd_sc_hd__a21bo_4      271
     sky130_fd_sc_hd__a21boi_4      47
     sky130_fd_sc_hd__a21o_4       403
     sky130_fd_sc_hd__a21oi_4      130
     sky130_fd_sc_hd__a22oi_4       42
     sky130_fd_sc_hd__a2bb2o_4    1219
     sky130_fd_sc_hd__a2bb2oi_4     24
     sky130_fd_sc_hd__a32o_4       467
     sky130_fd_sc_hd__a32oi_4      261
     sky130_fd_sc_hd__and2_4       700
     sky130_fd_sc_hd__and3_4       880
     sky130_fd_sc_hd__and4_4        83
     sky130_fd_sc_hd__buf_1       3902
     sky130_fd_sc_hd__buf_8          4
     sky130_fd_sc_hd__conb_1         6
     sky130_fd_sc_hd__dfrtp_4     1650
     sky130_fd_sc_hd__dfstp_4        8
```

Comment: We tried to synthesize IBEX RTLs to generate a net list but turned out that there was a corrupted network inside the generated netlist that we are trying to allocate it originally inside the IBEX verilog files (Error: Combinational network).