Google Summer of Code 2024 Final Report

Student: V Vignesh Karthik
GitHub: https://github.com/VigneshKarthikV
Project: ASIC Design of TCAM IPs using Sky130 and OpenLane
Organisation: Microelectronics Research Lab
Mentors: Sajjad Ahmed
Project GitHub:
https://github.com/merledu/OpenTCAM/tree/dev_layout

1. Introduction

1.1. Objective

This project involves making the layouts using open source ASIC Design tools such as OpenLane and open source PDKs such as Sky130 to run the ASIC Design Flow on TCAM IPs for various configurations.

1.2. What is a TCAM?

Ternary Content Addressable Memory is a type of memory structure where data in the memory is searched by matching every bit with the input data in different addresses and after checking all the matched lines, the data and address can be output. Ternary here refers to three states of data, namely, 0, 1 and x. In this context, x refers to a don't care which means it can either be 0 or 1. This is typically difficult to store in a physical form and requires additional transistors and thereby increasing area and power consumption. In this project, we use SRAM macros to emulate TCAM IPs and perform a TCAM to SRAM mapping to store the data.

The below figure shows a simplified view of a TCAM. The input data 110 is matched with different data in memory and two potential match addresses are 10 and 11. We resolve this using a priority encoder as shown.

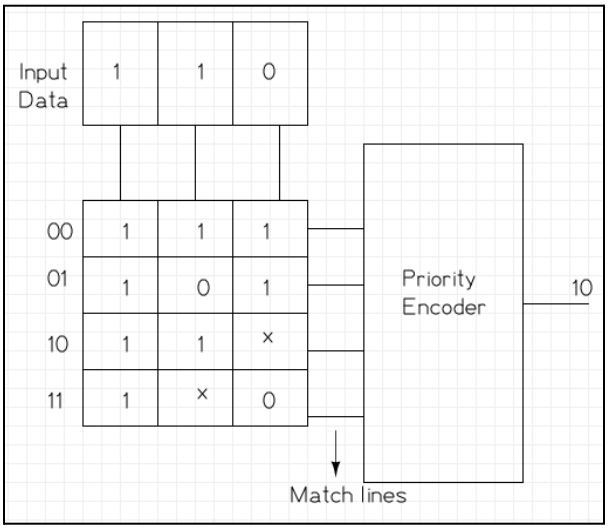

Figure 1: A simple TCAM

1.3. TCAM to SRAM Mapping

- SRAM size (2n X W) ⇔ TCAM size (W X n)
- Please refer to this link to see an example of TCAM to SRAM mapping.

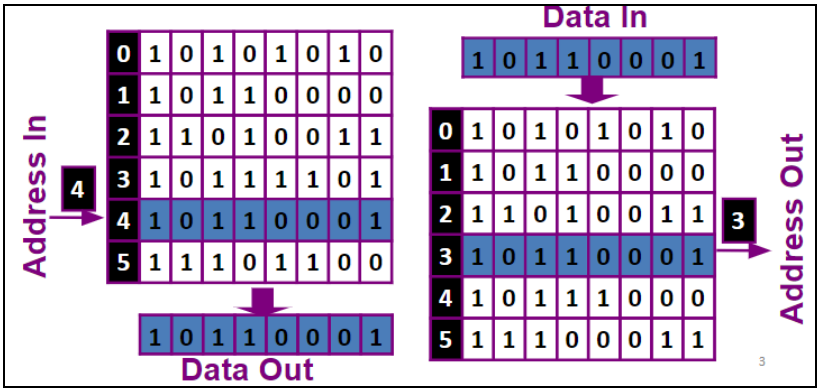

Figure 2: TCAM to SRAM mapping

1.4. TCAM Block Diagram

- Let us consider the TCAM 28x32. The inputs to the TCAM are connected to the two SRAM instances each containing two ports, 1 read/write port and 1 read port.
- The address decoder splits the addresses to the SRAM. For read operation, bit 1 and 0 are inserted accordingly to get the right address range and for write operation it uses block select signal to select the SRAM.
- The outputs from the two ports(shown on the right side of SRAMs in green colour) are ANDed together. These signals from the two AND gates from the two SRAMs are ANDed and sent to a priority encoder to get the output.

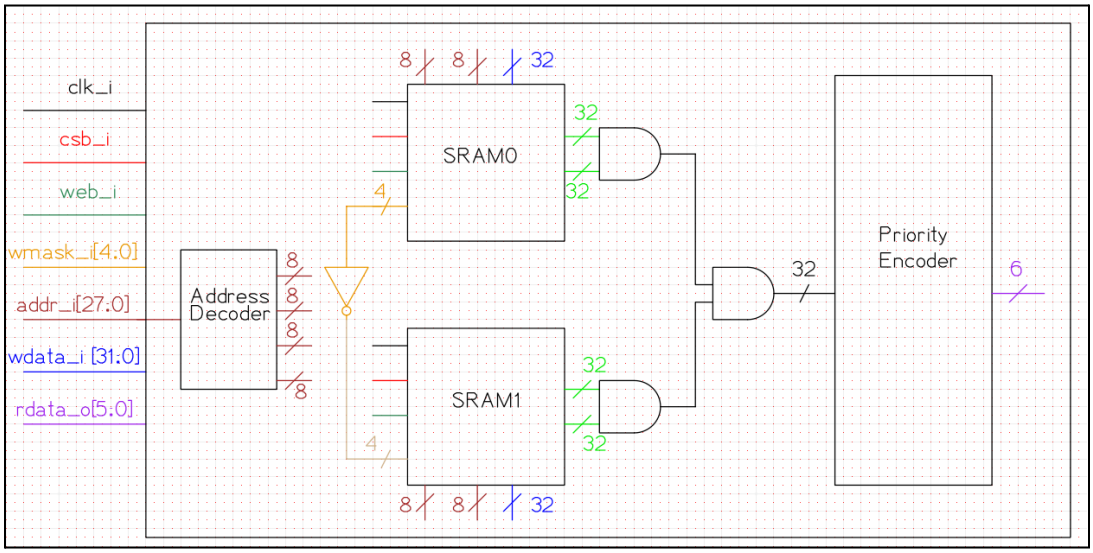

Figure 3: TCAM 28x32 Block Diagram

2. User Guide for ASIC Design of TCAM IPs using Sky130

2.1. Environment setup

- Add your custom design, lets say for TCAM 28x32, in OpenLane using the following command on a terminal emulator:

```
make mount
./flow.tcl -design ./designs/ci/tcam_32x28 -init_design_config
-add_to_designs
```

This creates a directory `tcam_32x28` with a `config.json` file and `/src` directory.

- Add the RTL code in the `/src` directory. For example, `top_tcam_32x28.sv` can be added.
- Add the required configurations in the `config.json` file. Some of the variables used will be discussed in the later sections.

2.2. Running the OpenLane flow

- OpenLane flow can be run in two ways, it can be run by executing the flow.tcl file or in interactive mode. Here we will be discussing how to run it in **interactive mode**.
- The following can be run to put OpenLane into interactive mode in the `/OpenLane` directory:

```
./flow.tcl –interactive
```

- Then run the following:

```
package require openlane
prep -design <design> [-tag TAG] [-config CONFIG]
[-init_design_config] [-overwrite]
run_synthesis
run_floorplan
run_placement
run_cts
run_routing
run_magic
run_magic_spice_export
run_magic_drc
run_lvs
run_antenna_check
```

- Additional steps can be run that the user can find on the OpenLane user guide.
- Once the above steps are run, the run will be saved in the `/runs` directory and you can view the layout in various stages such as f**loorplanning, placement, routing and signoff**. For example, to view the floorplan in .odb format using the openroad viewer, you can run the following:

```
python3 gui.py --viewer openroad --stage floorplan
./designs/ci/runs/<TAG>
```

Note: If you get the following error:

```
[ERROR GUI-0070] Error: gui.tcl, 14 can't read
"::env(SCRIPTS_DIR)": no such variable
```

Set the environment variable from the `/OpenLane` as shown:

```
export SCRIPTS_DIR=./scripts
```

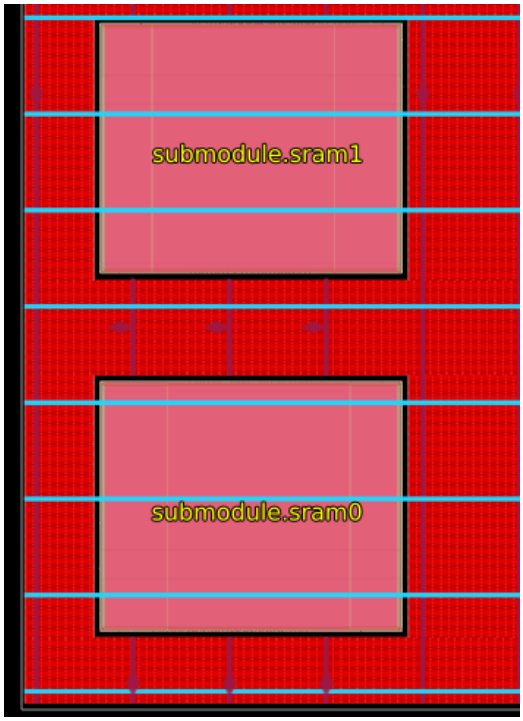

Figure 4: Floorplan of the TCAM 28x32

2.3. Flow variables

OpenLane flow configuration variables can be set based on the following:

- `PL_TARGET_DENSITY`: It reflects how spread the cells would be on the core area.
- `FP_SIZING`: Whether to use relative sizing by making use of FP_CORE_UTIL or absolute one using DIE_AREA
- `FP_PDN_MACRO_HOOKS`: Specifies explicit power connections of internal macros to the top level power grid
- `ROUTING_CORES`: Specifies number of threads to be used by TritonRoute
- `MACRO_PLACEMENT_CFG`: Can be used to specify a custom macro placement file
- `DRT_MAX_LAYER`: Specifies the max layer for detailed routing
- `GLB_RESIZER_SETUP_MAX_BUFFER_PERCENT`: Specifies a max number of buffers to insert to fix hold violations in terms of percentage of the number of instances in the design.`GLB_RESIZER_SETUP_MAX_BUFFER_PERCENT` can also be used.
- `GLB_RESIZER_HOLD_MAX_BUFFER_PERCENT`: Specifies a max number of buffers to insert to fix hold violations in terms of percentage of the number of instances in the design. `PL_RESIZER_HOLD_MAX_BUFFER_PERCENT` can also be used.

2.4. Flow automation script

2.4.1. Features

- The function `find_sram_module_line(file_path, search_module)` searches for lines containing the string `'sky130_sram_1kbyte_1rw1r_32x256_8'` in `top_tcam_32x28.sv` and returns it, which is further given to the function `sram_instance_name(sram_lines)` which returns the instance name, for example, `sram0` which is instantiated in the `top_tcam_32x28.sv`
- Once the SRAM instances are identified, the script places these macros starting with minimum spacing between the macros and minimum area. It gradually increases the area and spacing in case the flow doesn't complete.
- The function `area(sram_coordinates, macro_margin, margin_index)` calculates the area of the places macros, `generate_pdn(sram_coordinates)` connects the PDN hooks which is required by the flow configuration variable `FP_PDN_HOOKS` in the `config.json` which will be discussed in the later sections.
- The function `capture_flow_output(command)` and `capture_error(arr)` capture if there is any error in the flow. If there is any error, it increases the area and runs the flow again. It does this until a specified number of interactions or if the error is fixed.

2.4.2. Using the automation script

- Place the macro_placement_coordinates.py file in the same directory as the config.json and use the script1 file to run OpenLane in interactive mode.
- The paths in macro_placement_coordinates.py should be changed accordingly.
- Run the following python script in the docker container bash terminal:

```
python3 macro_placement_coordinates.py
```

3. TCAM Configurations

3.1. TCAM 28x32

3.1.1. SRAM Mapping:

- Two SRAM 256x32 modules are instantiated called **sram0 and sram1**. The mapping is as follows:
  - **sram0:**
    - Address range [127:0] corresponds to `{1'b0, addr_i[13:7]}`
    - Address range [255:128] corresponds to `{c_hi,addr_i[6:0]}`
    - `c_hi = 1'b1`

- **sram1:**
  - Address range [127:0] corresponds to `{1'b0, addr_i[27:21]}`
  - Address range [255:128] corresponds to `{c_hi,addr_i[20:14]}`
  - `c_hi = 1'b1`

- For read operation, the address must be given to `addr_i[27:0]` and the address split is done internally to the SRAM instances.
- For write operation, the lower 8 bits of `addr_i` must be used while making the 8th bit `1'b1` to enable write using `aw_select`.
- The signals are ANDed and an output is obtained by giving these signals to a priority encoder.

3.1.2. Layout

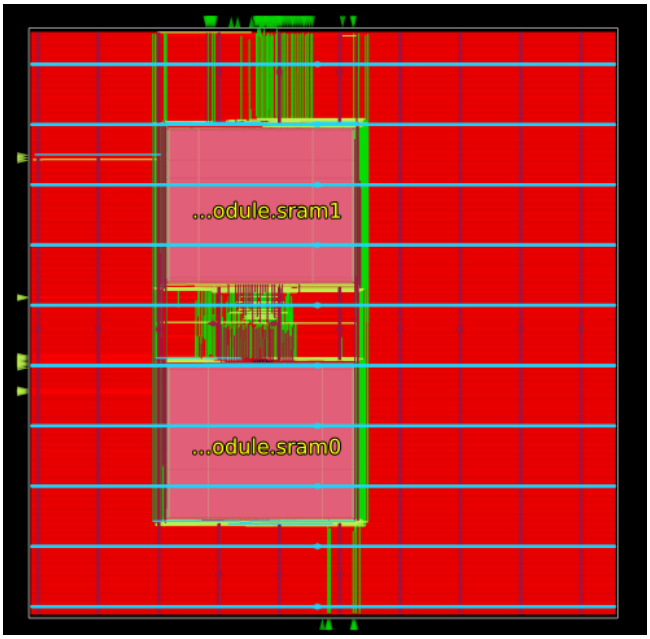

Figure 5: Layout of TCAM 28x32

3.1.3. Report

3.1.3.1. Area, Clock Frequency and Wiring Length

Core Area	(5.52, 10.88), (1194.08, 1187.84) = 1.39 mm^2
Die Area	(0, 0), (1200, 1200) = 1.44 mm^2
Clock Frequency	125 MHz
Wiring Length	186975 um

3.1.3.2. Power

Group	Internal Power	Switching Power	Leakage Power	Total Power
Sequential	0.00e+00	0.00e+00	0.00e+00	0.00e+00 0.0%
Combinational	2.49e-05	9.73e-06	1.83e-09	3.47e-05 1.4%
Clock	3.14e-05	7.62e-06	5.60e-09	3.90e-05 1.6%
Macro	2.34e-03	0.00e+00	3.81e-05	2.38e-03 97%
Pad	0.00e+00	0.00e+00	0.00e+00	0.00e+00 0.0%
Total	2.40e-03	1.73e-05	3.81e-05	2.45e-03 100%

3.2. TCAM 28x64

3.2.1. SRAM Mapping:

- Two TCAM 28x32 are instantiated and in turn four SRAM 256x32 modules are instantiated called **tcam0.sram0, tcam0.sram1, tcam1.sram0** and **tcam1.sram1**. The mapping is as follows:
    - **tcam0:**
        - Address is `addr_i[27:0]`
        - `wdata_i` is split and lower half is given
        - **sram0:**
            - Address range [127:0] corresponds to `{1'b0, addr_i[13:7]}`
            - Address range [255:128] corresponds to `{c_hi,addr_i[6:0]}`
            - `c_hi = 1'b1`
        - **sram1:**
            - Address range [127:0] corresponds to `{1'b0, addr_i[27:21]}`
            - Address range [255:128] corresponds to `{c_hi,addr_i[20:14]}`
            - `c_hi = 1'b1`
    - **tcam1:**
        - Address is `addr_i[27:0]`
        - `wdata_i` is split and upper half is given
        - **sram0:**

        - Address range [127:0] corresponds to `{1'b0, addr_i[13:7]}`
        - Address range [255:128] corresponds to `{c_hi,addr_i[6:0]}`
        - `c_hi = 1'b1`
    - **sram1:**
        - Address range [127:0] corresponds to `{1'b0, addr_i[27:21]}`
        - Address range [255:128] corresponds to `{c_hi,addr_i[20:14]}`
        - `c_hi = 1'b1`
- For read operation, the address must be given to `addr_i[27:0]` and the address split is done internally to the SRAM instances.
- For write operation, the lower 8 bits of `addr_i` must be used while making the 8th bit `1'b1` to enable write using `aw_select`.
- The signals are ANDed and an output is obtained by giving these signals to a priority encoder.
- The outputs from the SRAMs are concatenated to form a 64 bit output.

3.2.2. Layout

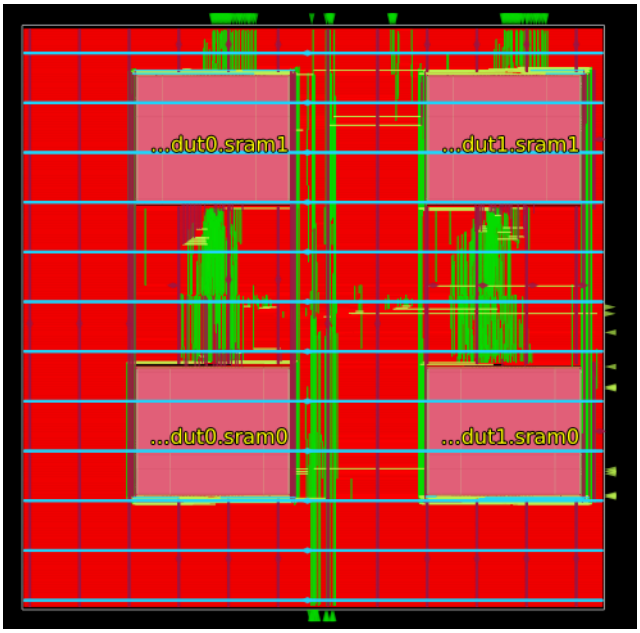

Figure 6: Layout of TCAM 28x64

3.2.3. Report

3.2.3.1. Area, Clock Frequency and Wiring Length

Core Area	(5.52, 10.88), (1794.46 1787.04) = 3.17 mm^2

Die Area	(0, 0), (1800, 1800) = 3.24 mm^2
Clock Frequency	125 MHz
Wiring Length	442212 um

3.2.3.2. Power

Group	Internal Power	Switching Power	Leakage Power	Total Power
Sequential	0.00e+00	0.00e+00	0.00e+00	0.00e+00 0.0%
Combinational	1.36e-04	9.45e-05	4.42e-09	2.31e-04 1.5%
Clock	1.01e-04	6.77e-05	8.31e-09	1.69e-04 1.1%
Macro	1.46e-02	0.00e+00	7.61e-05	1.47e-02 97.4%
Pad	0.00e+00	0.00e+00	0.00e+00	0.00e+00 0.0%
Total	1.49e-02	1.62e-04	7.61e-05	1.51e-02 100.0%

3.3. TCAM 56x32

3.3.1. SRAM Mapping:

- Two TCAM 28x32 modules are instantiated in turn called **tcam0.sram0, tcam0.sram1, tcam1.sram0** and **tcam1.sram1**. The mapping is as follows:
  - **tcam0:**
    - Input address is `addr_i[27:0]`
    - `p_out1` is taken from TCAM0 and will be given ANDed `p_out2` to give `p_out`
    - **sram0:**
      - Address range [127:0] corresponds to `{1'b0, addr_i[13:7]}`
      - Address range [255:128] corresponds to `{c_hi,addr_i[6:0]}`
      - `c_hi = 1'b1`
    - **sram1:**
      - Address range [127:0] corresponds to `{1'b0, addr_i[27:21]}`

        - Address range [255:128] corresponds to `{c_hi,addr_i[20:14]}`
        - `c_hi = 1'b1`
    - **tcam1:**
        - Input address is `addr_i[55:28]`
        - `p_out2` is taken from TCAM1 and will be given ANDed `p_out1` to give `p_out`
        - **sram0:**
            - Address range [127:0] corresponds to `{1'b0, addr_i[13:7]}`
            - Address range [255:128] corresponds to `{c_hi,addr_i[6:0]}`
            - `c_hi = 1'b1`
        - **sram1:**
            - Address range [127:0] corresponds to `{1'b0, addr_i[27:21]}`
            - Address range [255:128] corresponds to `{c_hi,addr_i[20:14]}`
            - `c_hi = 1'b1`
- For read operation, the address must be given to `addr_i[55:0]` and the address split is done internally to the SRAM instances.
- For write operation, the lower 8 bits of `addr_i` must be used while making the 8th bit `1'b1` to enable write using `aw_select`.
- `p_out1` and `p_out2` are ANDed to give `p_out` and further it is given to a priority encoder to obtain `rdata_o[5:0]`

3.3.2. Layout

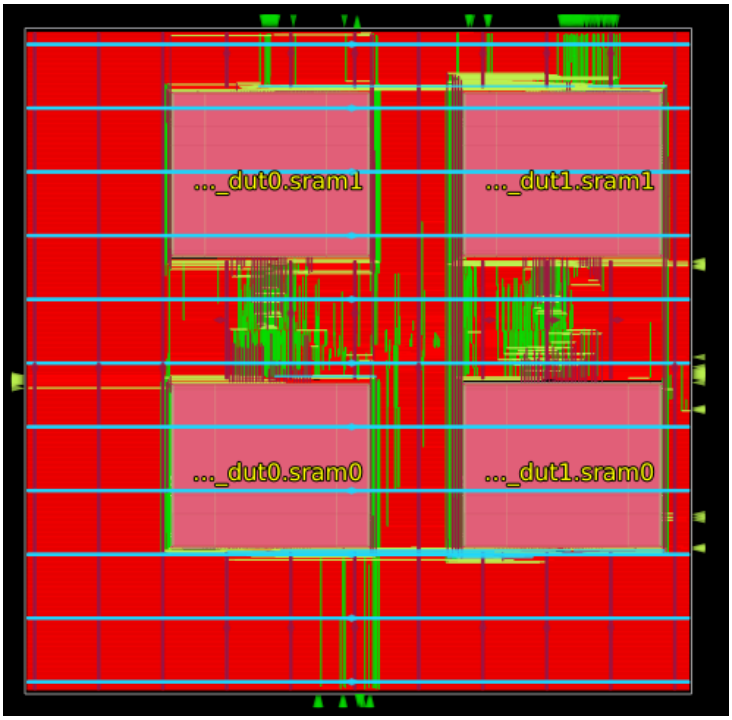

Figure 7: Layout of TCAM 56x32

3.3.3. Report

3.3.3.1. Area, Clock Frequency and Wiring Length

Core Area	(5.52, 10.88), (1594.36, 1588.48) = 2.5 mm^2
Die Area	(0, 0), (1600, 1600) = 2.56 mm^2
Clock Frequency	125 MHz
Wiring Length	391437 um

3.3.3.2. Power

Group	**Internal Power**	**Switching Power**	**Leakage Power**	**Total Power**
Sequential	0.00e+00	0.00e+00	0.00e+00	0.00e+00 0.0%
Combinational	1.54e-04	1.34e-04	4.16e-09	2.88e-04 1.9%
Clock	1.01e-04	6.18e-05	7.83e-09	1.63e-04 1.1%
Macro	1.46e-02	0.00e+00	7.61e-05	1.47e-02 97.0%

Pad	0.00e+00	0.00e+00	0.00e+00	0.00e+00 0.0%
Total	1.49e-02	1.96e-04	7.61e-05	1.52e-02 100.0%

4. SRAM Configurations

4.1. SRAM 32x4_8

4.1.1 Layout

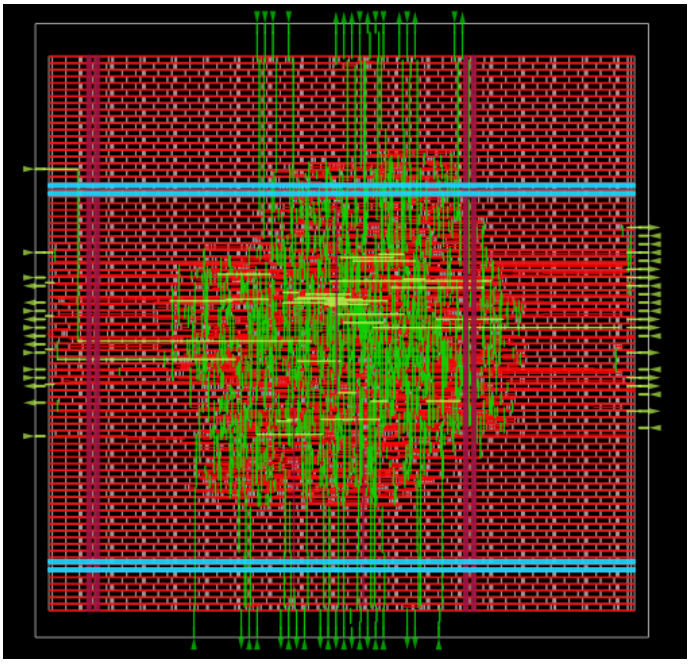

Figure 8: Layout of SRAM 32x4_8

4.1.2. Report

4.1.2.1. Area, Clock Frequency and Wiring Length

Core Area	(5.52, 10.88), 244.26, 236.64 = 0.054 mm^2
Die Area	(0, 0), (300, 300) = 0.0625 mm^2
Clock Frequency	125 MHz
Wiring Length	27154 um

4.1.2.2. Power

Group	Internal Power	Switching Power	Leakage Power	Total Power
Sequential	1.22e-03	6.98e-05	1.68e-09	1.29e-03 30.8%
Combinational	5.52e-04	2.31e-04	2.43e-09	7.84e-04 18.7%
Clock	1.44e-03	6.75e-04	1.97e-09	2.11e-03 50.5%
Macro	0.00e+00	0.00e+00	0.00e+00	0.00e+00 0.0%
Pad	0.00e+00	0.00e+00	0.00e+00	0.00e+00 0.0%
Total	3.21e-03	9.76e-04	6.08e-09	4.19e-03 100.0%

4.2. SRAM 64x4_8

4.2.1 Layout

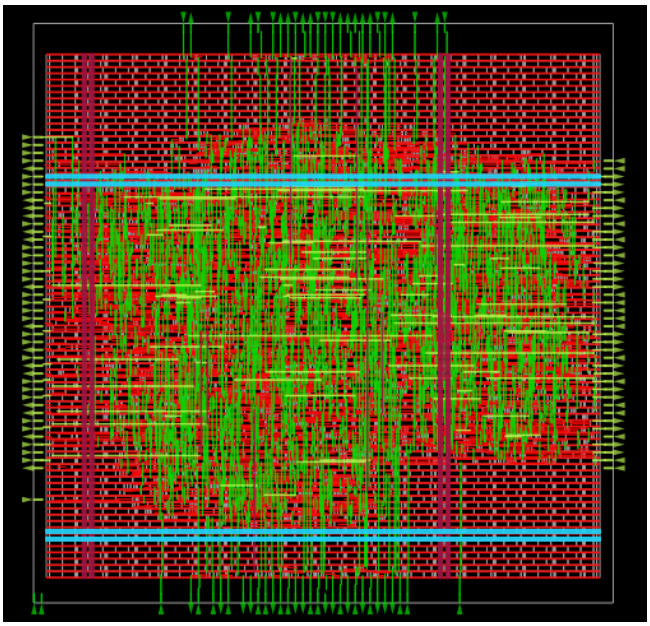

Figure 9: Layout of SRAM 64x4_8

4.2.2. Report

4.2.2.1. Area, Clock Frequency and Wiring Length

Core Area	(5.52, 10.88), (244.26, 236.64) = 0.54 mm^2
Die Area	(0, 0), (250, 250) = 0.625 mm^2

Clock Frequency	125 MHz
Wiring Length	55468 um

4.2.2.2. Power

Group	Internal Power	Switching Power	Leakage Power	Total Power
Sequential	2.41e-03	1.46e-04	3.30e-09	2.56e-03 30.6%
Combinational	1.12e-03	4.64e-04	4.89e-09	1.58e-03 18.9%
Clock	2.86e-03	1.37e-03	3.39e-09	4.22e-03 50.5%
Macro	0.00e+00	0.00e+00	0.00e+00	0.00e+00 0.0%
Pad	0.00e+00	0.00e+00	0.00e+00	0.00e+00 0.0%
Total	6.39e-03	1.98e-03	1.16e-08	8.37e-03 100.0%

4.3. SRAM 72x4_8

4.3.1 Layout

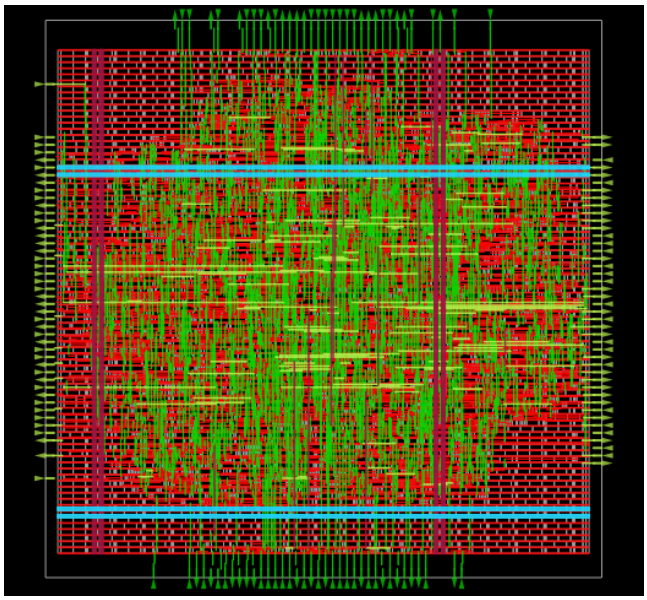

Figure 10: Layout of SRAM 72x4_8

4.3.2. Report

4.3.2.1. Area, Clock Frequency and Wiring Length

Core Area	(5.52, 10.88), (244.26, 236.64) = 0.54 mm^2
Die Area	(0, 0), (250, 250) = 0.625 mm^2
Clock Frequency	125 MHz
Wiring Length	63163 um

4.3.2.2. Power

Group	**Internal Power**	**Switching Power**	**Leakage Power**	**Total Power**
Sequential	2.17e-03	1.29e-04	3.73e-09	2.30e-03 30.8%
Combinational	9.95e-04	4.15e-04	5.73e-09	1.41e-03 18.8%
Clock	2.53e-03	1.24e-03	3.71e-09	3.77e-03 50.4%
Macro	0.00e+00	0.00e+00	0.00e+00	0.00e+00 0.0%
Pad	0.00e+00	0.00e+00	0.00e+00	0.00e+00 0.0%
Total	5.70e-03	1.78e-03	1.32e-08	7.48e-03 100.0%

4.4. SRAM 96x4_8

4.4.1 Layout

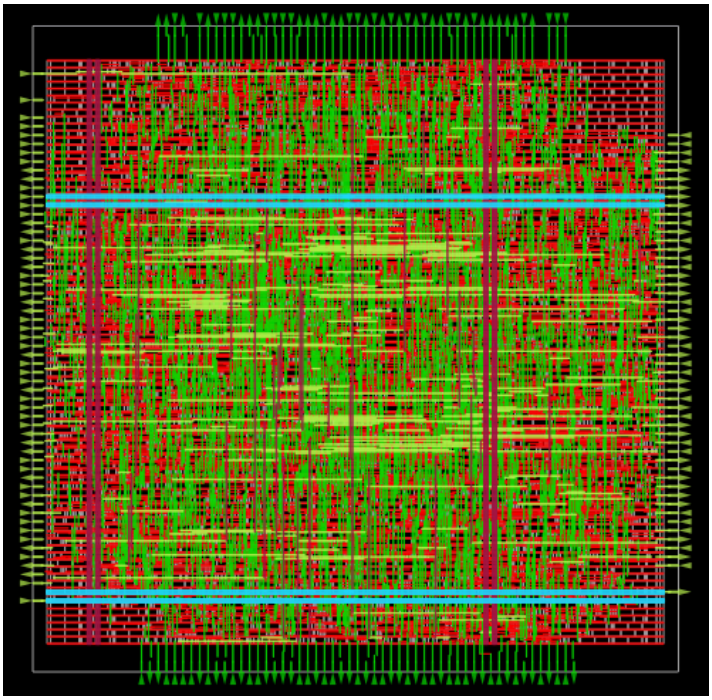

Figure 11: Layout of SRAM 96x4_8

4.4.2. Report

4.4.2.1. Area, Clock Frequency and Wiring Length

Core Area	(5.52, 10.88), (244.26, 236.64) = 0.54 mm^2
Die Area	(0, 0), (250, 250) = 0.625 mm^2
Clock Frequency	125 MHz
Wiring Length	87696 um

4.4.2.2. Power

Group	**Internal Power**	**Switching Power**	**Leakage Power**	**Total Power**
Sequential	3.61e-03	2.10e-04	4.97e-09	3.82e-03 31.1%
Combinational	1.61e-03	6.84e-04	7.51e-09	2.30e-03 18.7%
Clock	4.13e-03	2.03e-03	4.73e-09	6.16e-03 50.2%
Macro	0.00e+00	0.00e+00	0.00e+00	0.00e+00 0.0%

Pad	0.00e+00	0.00e+00	0.00e+00	0.00e+00 0.0%
Total	9.35e-03	2.92e-03	1.72e-08	1.23e-02 100.0%

4.5. SRAM 128x4_8

4.5.1.Layout

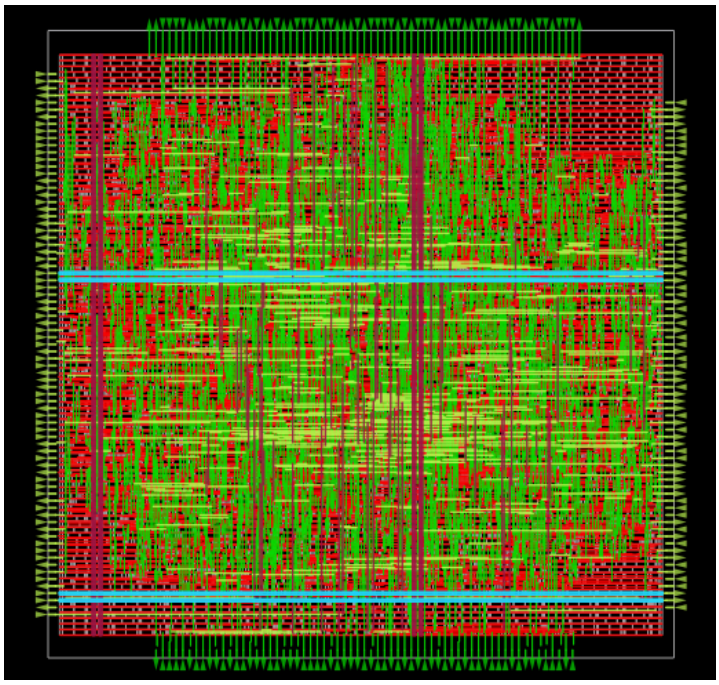

Figure 12: Layout of SRAM 128x4_8

4.5.2.Report

4.5.2.1. Area, Clock Frequency and Wiring Length

Core Area	(5.52, 10.88), (294.4, 288.32) = 0.8 mm^2
Die Area	(0, 0), (300, 300) = 0.9 mm^2
Clock Frequency	125 MHz
Wiring Length	131542 um

4.5.2.2. Power

Group	**Internal Power**	**Switching Power**	**Leakage Power**	**Total Power**

Sequential	4.81e-03	2.77e-04	6.61e-09	5.08e-03 31.0%
Combinational	2.19e-03	9.21e-04	1.04e-08	3.11e-03 19.0%
Clock	5.47e-03	2.75e-03	6.23e-09	8.22e-03 50%
Macro	0.00e+00	0.00e+00	0.00e+00	0.00e+00 0.0%
Pad	0.00e+00	0.00e+00	0.00e+00	0.00e+00 0.0%
Total	1.25e-02	3.95e-03	2.32e-08	1.64e-02 100.0%

5. Summary

Macro	**Clock Frequency**	**Area**	**Power**
tcam_32x28	125 MHz	1.44 mm^2	(2.45 + 2*12.5) = 27.45 mW
tcam_64x28	125 MHz	3.24 mm^2	(15.1 + 4*12.5) = 65.1 mW
tcam_32x56	125 MHz	2.56 mm^2	(15.2 + 4*12.5) = 65.2 mW
sky130_sram_16byte_1rw_32x4_8	125 MHz	0.0625 mm^2	4.19 mW
sky130_sram_32byte_1rw_64x4_8	125 MHz	0.0625 mm^2	8.37 mW
sky130_sram_36byte_1rw_72x4_8	125 MHz	0.0625 mm^2	7.48 mW
sky130_sram_48byte_1rw_96x4_8	125 MHz	0.0625 mm^2	12.23 mW
sky130_sram_64byte_1rw_128x4_8	125 MHz	0.09 mm^2	16.4 mW

Note: While calculating power, the power consumption of each SRAM is taken into account and added to the total power.

6. Weekly tasks

Phase	**Task**	**Week**
1. Community Bonding Period		
1.1	Went through code base to understand TCAM to SRAM mapping	Week 1: May 1 - May 8
1.2	Understanding different TCAM configurations	Week 2: May 8 - May 15

1.3	Environment setup to run the ASIC Design Flow using OpenLane	Week 3: May 15 - May 22
2. RTL and ASIC Design of TCAM IPs using OpenLane and Sky130		
2.1	• Modified RTL Code for some existing configurations such as TCAM 28x32 and TCAM 7x64 • Built a framework to do the ASIC Design Flow	Week 4: May 23 - May 31
2.2	Initial Layout of TCAM 7x64 using Sky130 PDK and Sky130 SRAM Macro	Week 5: Jun 1 - Jun 8 [Branch: dev_layout] Commit #195 :70b177a
2.3	RTL for the following configurations were coded: • TCAM 28x32 • TCAM 56x32	Week 6 - 8: Jun 9 - Jun 30 [Branch: dev_layout] Commit #196: d1dfde3 Commit: #197: 2786e32 Commit #198: 46afdba Commit #199: 5537ae3
2.4	ASIC Design of the above TCAM IPs	
3. OpenLane Flow Automation		
3.1	Python script to instantiate SRAM macros based on TCAM configuration	Week 10: Jul 1 - Jul 8 Commit #200: 8d84508
3.2	Modified the code to correct errors due to area and macro placement	Week 11: Jul 9 - Jul 16 Commit #201: 22b877b
4. RTL and ASIC Design of SRAM Macros using OpenLane and Sky130		
4.1	Code for custom SRAM configuration SRAM 72x4_8 and OpenLane flow	Week 12: Jul 16 - Jul 23 Commit #202: c78638ae Commit #203: 450fd1f
4.2	RTL code and OpenLane flow for following custom SRAM configurations: • SRAM 32x4_8 • SRAM 64x4_8 • SRAM 96x4_8 • SRAM 128x4_8	
5. Report of various parameters for different configurations		
5.1	• Optimising the OpenLane flow for area, power, routing and macro placement • Making a report of core area, die area, clock frequency, power consumption and routing length	Week 13 - 14: Jul 24 - Aug 8

6. Documentation		
6.1	• Documentation of RTL code and ASIC Design of TCAM IPs • Documentation of RTL code and ASIC Design of SRAM macros	Week 14 - 15: Aug 9 - Aug 19

7. Future Work

ASIC Design of TCAM IPs is done using Sky130 PDK currently. It should be run using GF180 PDK also. SRAM macros instantiated are from Sky130 PDK which has a limited number of configurations. This has been increased by writing custom RTL for other SRAM configurations. However, this limits the number of TCAM configurations. DFFRAM can be used for custom configurations. Increase in clock frequency can be tried to achieve and proper timing closure for a flip flop based RAM structure.