INTERNATIONAL JOURNAL OF CIRCUITS, SYSTEMS AND SIGNAL PROCESSING
DOI: 10.46300/9106.2021.15.65 Volume 15, 2021

Design and Implementation of 64-bit SRAM and CAM on Cadence and Open-source environment

N. Shylashree[1], Yatish D. Vahvale[2], N. Praveena[3], A. S. Mamatha[4]

[1]Department of Electronics and communication Engineering, RV College of Engineering, Bengaluru, India.
[2]Department of Electronics and communication Engineering, RV College of Engineering, Bengaluru, India
[3]Department of Electronics and communication Engineering, RV College of Engineering, Bengaluru -560059 and affiliated to VTU, Belagavi, Karnataka, India.
[4]Department of Electronics & Communication Engineering, St.Joseph Engineering College, Mangaluru-5750 28, India.

Received: January 30, 2021. Revised: July 21, 2021. Accepted: July 10, 2021. Published: July 14, 2021.

Abstract— **Low-power IC design has become a priority in recent years because of the growing proliferation of portable battery-operated devices, bringing Static Random-Access Memory (SRAM) and Content Addressable Memory (CAM) into play. In today's SoCs, embedded SRAM units have become a necessary component. There is a lack of chips in the current world and to manufacture chips there is the requirement of Electronic Design Automation(EDA) tools that can perform better. In this paper, the main motive is to showcase the performance of open-source tools available currently which can still generate the required output with no cost. In this new era of fast mobile computing, traditional SRAM cell designs are power-demanding and underperforming. Rather than lowering manufacturing costs through high-volume production, specialty memory give cost-effective alternatives through architecture. Specialty memory devices enable the designer to address issues like board area, important timing, data flow bottlenecks, and so on in ways that high-volume regular memory devices cannot. Implementation of memory devices on Cadence environment and open-source environment to check the compatibility and compare the power, area, and delay of both 64-bit SRAM and CAM also analysing and validating the results of both the memory devices in this paper. For SRAM in a cadence environment, the calculated power, area, and slack have improved values, namely 0.145mW, 1104.3µm^2, and positive slack of 6636 for pre-layout analysis. Furthermore, the power for 64-bit CAM in a cadence context is nearly identical to those for an open-source environment ~0.8mW. In an open-source environment, the calculated slack for CAM is 4.74.**

Keywords—Yosys, OpenSta, genus, Graywolf, innovus, qflow tool

I. Introduction

Computer memory refers to any physical device capable of temporarily storing information, such as RAM (random access memory), or permanently storing information, such as ROM (read-only memory) (read-only memory). Memory devices use integrated circuits (ICs), which are used by operating systems, software, and hardware. A CAM is a memory that implements the lookup-table function in a single clock cycle using dedicated comparison circuitry. CAMs are often found in network routers for packet forwarding and categorization, but they can also be found in a wide range of other applications that require quick table searching. The key CAM design problem is to limit the amount of power consumed by the vast quantity of parallel active circuitry without losing speed or memory density. Large integrated storage is required for image processing and other multimedia applications. Approximate memory has been proposed as a possible energy-efficient solution for such error-tolerant applications in some previous studies. When compared to normal 6-T SRAM cells with the same bit error rate, a single-ended 6-T (SE6T) static random-access memory (SRAM) cell has around 50% less dynamic power[1]. In the XNOR-SRAM bitcell, the ternary XNOR operations are accumulated on the read bit line (RBL) by turning on all 256 rows at the same time, resulting in a resistive voltage divider. The analog RBL voltage is digitized by a column-multiplexed 11-level flash analog-to-digital converter (ADC) at the XNOR-SRAM peripheral[2]. Yosys for Verilog synthesis and nextpnr for placement, routing, and bitstream generation is introduced in the [3] paper as a fully free and open-source software (FOSS) architecture-neutral FPGA framework. A novel task-based parallel incremental timing analysis engine to overcome the performance bottleneck of traditional loop-based methods, a new application programming interface (API) approach to utilize high degrees of parallelism, and improved support for industry-standard design formats to increase user experience [4]. The high dynamic power consumption associated with traditional CAM design's massive and active parallel hardware has long been a problem. However, with multigate devices replacing planar MOSFETs, deeply scaled technology nodes are projected to provide additional tradeoffs to CAM design [5-8]. By increasing the ratio to 4, read stability is improved, and write stability is improved by charging and discharging the storage node using two transistors. The proposed 8T SRAM architecture is compared to standard 6T, decoupled 8T, and 10T SRAM cells. To write data into a cell at 1.2V supply voltage, a 6T SRAM cell required 584.1mV word line voltage, but the suggested SRAM cell requires just only 512mV [9]. The suggested placement tool [10] begins by running a cutting-edge CMOS placer, which arranges rows of fixed-height but variable-width cells on the chip. Each row's cells are then clustered together in groups of at least k cells with the same logic level. Increasing k reduces chip area while also potentially lowering performance. Place-and-route results of a 32-bit Kogge-Stone [10] adder for various values of k is provided to evaluate the effectiveness of the suggested methodology. When compared to the results of a standard CMOS placement with an H-tree clock net, the overall chip size can be decreased by 27% employing this innovative design process. Because of its parallel search accessibility, content addressable memory (CAM) is one of the most promising

HSEs. However, it suffers from significant dissipation, which is exacerbated when several components, such as cells and accompanying match lines (MLs), are accessed during each search [11-14]. A ternary content-addressable memory (TCAM) cell with 12 transistors and two magnetic tunneling junctions (MTJs) is presented. The suggested TCAM cell consumes no static power during the search process, resulting in a very energy-efficient operation. The resistance of an MTJ in the anti-parallel state is compared to that of an MTJ in the parallel state for search operations [15-17]. papers [18-20] discuss the implementation of memory devices and their reaction to internal noise. In the paper [20-22] the authors discuss the various techniques used and the basic memory-based explanations required to understand the GDS flow. The ASIC understanding and its flow has been clearly explained in [23-25] these explanations have been used in this project. Based on the above analysis and understanding, the project has been implemented and exploring of new open-source tools is done.

Through the continuous review of different journals and papers, concluding the behavior of memory architectures in not yet been explored on open-source tools. So, the comparison can be done on cadence environment and open-source environment. The final values are to be expected to have almost nearest value with respect to each other. This validation of the resulting power, area and delay of memory device on open-source environment is the new approach to view the results in a perspective of VLSI EDA tools.

II. VLSI TOOLS

A. *Open source VLSI tools*

First, the different open source tools available in the market for generating gds2 files from scratch are yosys, magic, gray wolf, qflow, and openSta to calculate the timing, area, and power parameters. The magic tool is used for the placement of the cells also qrouter is to route the cell nets to connect to different parts of the design. The qflow is used to process different backend aspects of design to generate a gds2 file. These are the tools used to produce gds2 for memory design CAM and SRAM.

B. *Cadence environment for the development of the design*

Cadence's toolkit includes several programs for a variety of tasks, including schematic sketching, layout, verification, and simulation. These programs work on a variety of computer platforms. The open architecture also enables the incorporation of third-party or custom-built solutions. An application named Design Framework II is responsible for integrating all of these tools (DFW). The layout is created using the Virtuoso Layout Editor. A layout is made up of geometrical figures in various colors. It is then feasible to generate the final mask layers that are employed in the production of the design based on the size and color of these figures. Other cells can be included by instantiating their layout views. PNR is the final stage in constructing a huge design. This is when all of the chip's various components are positioned in their proper locations and connected. Because a design can easily have hundreds of connection points, manually connecting them would be difficult and time-consuming. The designer may also want to experiment with other layouts for the components, output buffers, memory structures, amplifiers, and so on.

III. METHODOLOGY

Before implementing the tool, it is of prime importance to first understand the methodology of the design movement from Verilog code to generating synthesized netlist and then from the .sdc file to control the timing parameters. Finally, generating gds2 on both the environment. In the previous section, the learning about SRAM and CAM and also the tools used in both the environments is done. This section demonstration of a better methodology of looking into the flow of the project.

The process of converting a design into manufacturable geometry is known as physical design. Floorplanning, placement, clock tree synthesis, and routing are among the steps involved. The physical design process begins with the creation of a netlist from RTL. The netlist is a diagram that shows how a circuit's components are connected. The first and most important stage is to plan the layout of the space. It entails determining which buildings should be put next to one another while taking into account space constraints, speed, and the numerous constraints imposed by components. Figure 1 shows the flow chart for generating GDSII in a cadence environment. Figure 2 shows the flow chart for generating GDSII in an open-source environment. The open-source tool “qrourter” is used for the placement and detailed routing of the cells. To have a proper layout the tool used is “magic”.

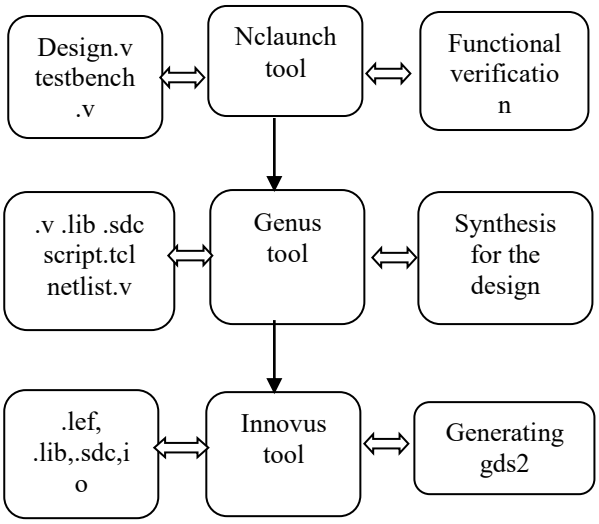

Fig. 1. The flow of cadence environment to generate gds2 from design.

The “open timer” tool is utilized for the final timing analysis; this tool uses C++ programming because it is an object-oriented technique to solve highly complex numerical problems. In a cadence set, the blocks above illustrate the basic flow of code to the final GDS II of digital design. The Nclaunch tool is used to verify the functionality of a Verilog code with the DUT and testbench as inputs. The Genus tool is used to build a netlist from a design, whereas Innovus is used to build a final GDS II file. “Yosys” tool is used to verify the code, “BliFanout" is used for synthesis, Graywolf is used for placement as shown in fig 2. A chip is partitioned into functional blocks. The location of each component or block on the die is determined by placement, which takes into account timing and connection length. Inserting buffers or inverters into a clock tree allows the clock to be distributed equally across consecutive parts in a design while minimizing skew and latency. The different tools used in the

cadence environment are as shown in fig 1 i.e., genus and Innovus tools. The genus tool takes few files such as the Verilog design file, testbench file, constraint files to generate synthesis files. This generated netlist from synthesis is then used in the physical design of SRAM and CAM. The generated constraints file from synthesis is also used to clock constraint and liberty files and the lef file is used for floor planning and power planning. From the obtained files from synthesis and physical design to calculate area, power, and timing report. The available open-source tools Icarus Verilog is a simulation and synthesis tool for Verilog. It functions as a compiler, converting Verilog (IEEE-1364) source code to a target format. The compiler can generate an intermediate form called vvp assembly for batch simulation. The vvp command is used to execute this intermediate form. The compiler generates netlists in the desired format for synthesis. Magic is a VLSI layout tool with a long history. Universities and small businesses continue to use Magic VLSI. Even for folks who eventually rely on commercial tools for their product design flow, Magic is frequently rated as the easiest tool to use for circuit designing. Netgen is a program that compares netlists, a process known as LVS (Layout vs. Schematic).

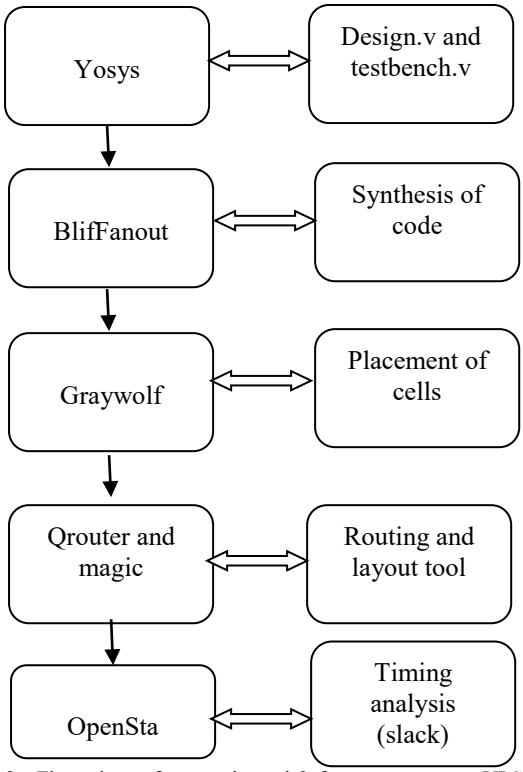

Fig. 2. Flow chart of generating gds2 from open-source VLSI tools for both SRAM and CAM

This is a crucial phase in the integrated circuit design process because it ensures that the laid-out geometry matches the predicted circuit. By proving circuit operation through extraction and simulation, very tiny circuits can skip this stage. Large digital circuits are typically created using tools that generate high-level descriptions and compilers that ensure proper layout geometry. Large analog or mixed-signal circuits that cannot be simulated in a reasonable amount of time are the most in need of LVS. LVS can be done considerably faster than simulation, even for small circuits, and offers feedback that makes it quicker to discover a mistake than simulation. Interconnect pathways, including conventional cell and macro pins, are determined by routing. This stage completes all of the connections defined in the netlist in the most efficient and time-constrained manner possible. GDSII, a data format that represents layout information, is usually the final result of the physical design process. Qrouter is a VLSI fabrication tool that generates metal layers and vias to physically connect a netlist. It's a maze router, sometimes known as a "sea-of-gates" router or an "over-the-cell" router. That is, unlike a channel router, it starts with a description of where standard cells should be located, usually at the smallest possible spacing, and then builds metal routes over the standard cells.

IV. Implementation on VLSI Environment

A. 64-bit SRAM on cadence environment

In this section, the discussion about the implementation of 64-bit SRAM in a cadence environment. Fig 3 shows the generated netlist for SRAM on the genus tool of cadence environment. The connection of the input registers to the internal flip flops and LUT (lookup table) of the design is done in the netlist. The netlist contains details such as the number of inputs, outputs and intermediate gates, and flip flops that are used to produce the functionality of the desired design. Similarly, the design on the backend has been continued on the innovus tool.

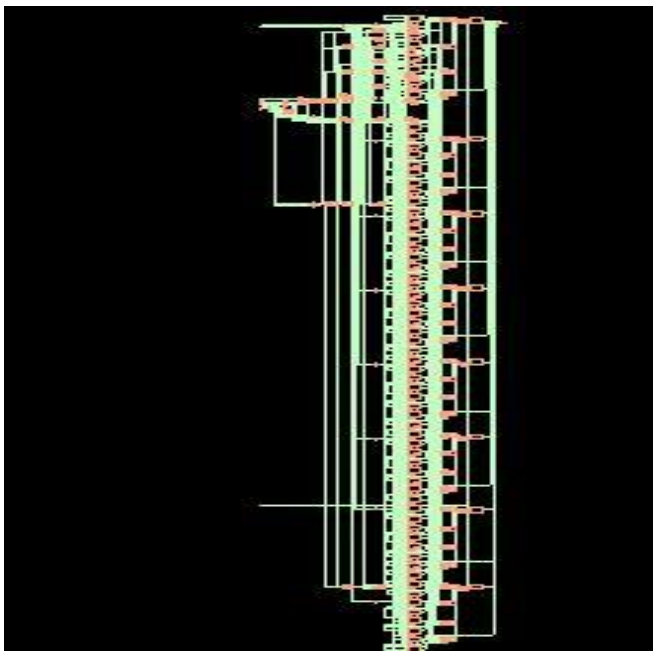

Fig. 3. Generated netlist of 64 bit SRAM using genus tool.

Using the innovus tool, the placement of cells and routing of wires for the design are done. Also, the power and clock arrangements of the design will be performed on innovus tools themselves. The 64-bit SRAM has been developed from innovus tool s shown in fig 4 & fig 5. Fig 5 shows the final arrangements and placements of the clock, wires, power grid, input and output location on the floor planning of the allotted area.

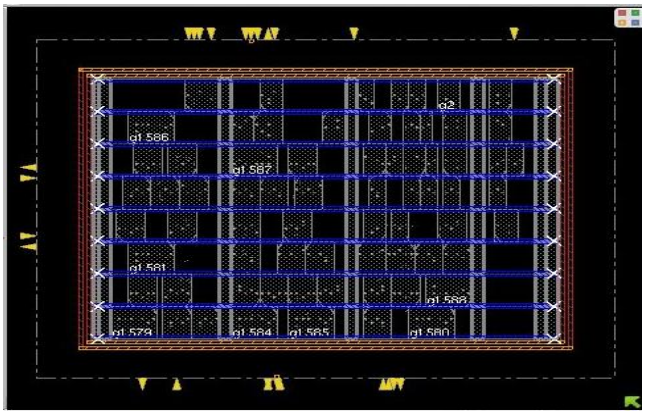

Fig. 4. Placement of 64 bit SRAM cells using Innovus tool.

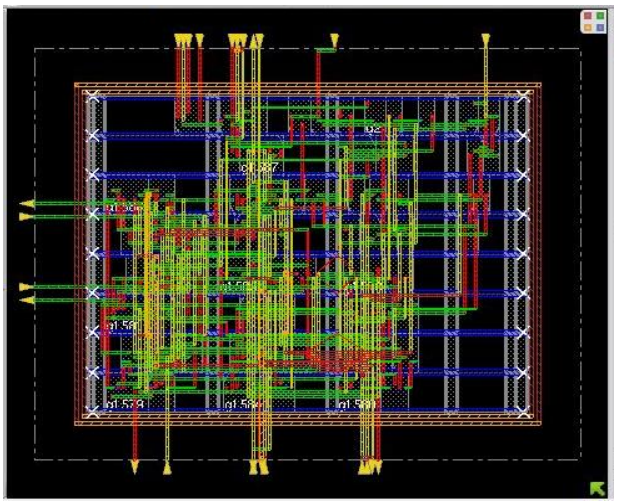

Fig. 5. Placement and routing did generate final gds2 for SRAM on the Innovus tool.

The fig 5 shows the final placement and routing defined for the SRAM on cadence environment. The section is ended with the final implementation of SRAM on cadence environment, the final power, area and slack are discussed in the Result section.

B. 64-bit CAM on cadence environment

This section discusses the implementation of 64-bit CAM on cadence environment with the tools shown in the fig1.The 64-bit CAM netlist has been shown in fig 6. The netlist shows the connection of wires, gates, flip flops, lookup tables from input to output of CAM design. The arrangements of clock given to different parts of the design and power grids to connect to different parts to distribute the power equally without any loss of power over the design without changing the functionality have been connected as shown in fig 7 for CAM. Similarly, from fig 8 the placement and routing of the CAM cells have been shown. In this section, the conclusion of the implementation of SRAM and CAM on cadence environment using genus and innovus tool. From the latter and former section implementation of the backend process for the design such as generating a netlist file, sdc file, and placing the cells in the proper place.

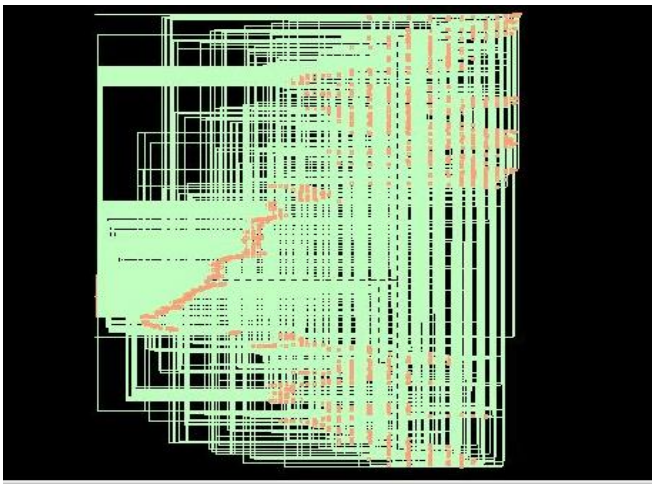

Fig. 6. 64 bit CAM netlist on genus tool of cadence environment.

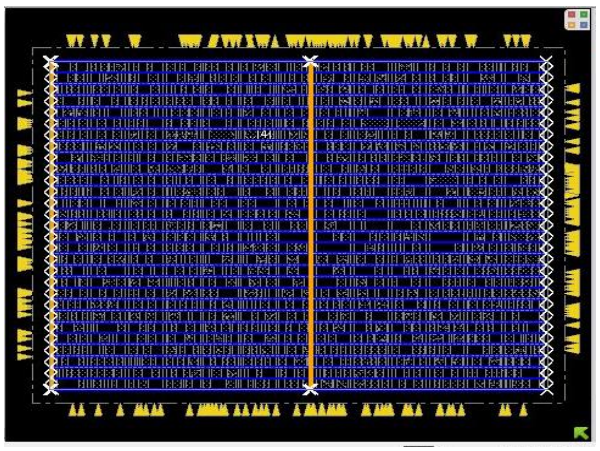

Fig. 7. 64 bit CAM placement of cells using innovus tool.

Also, generated a gds2 file for both SRAM and CAM. The calculation of power, area, and slack for the design is done on both genus and innovus tools of cadence environment for pre and post-validation. In the upcoming section, a discussion about the implementation of open source tools for both SRAM and CAM is shown.

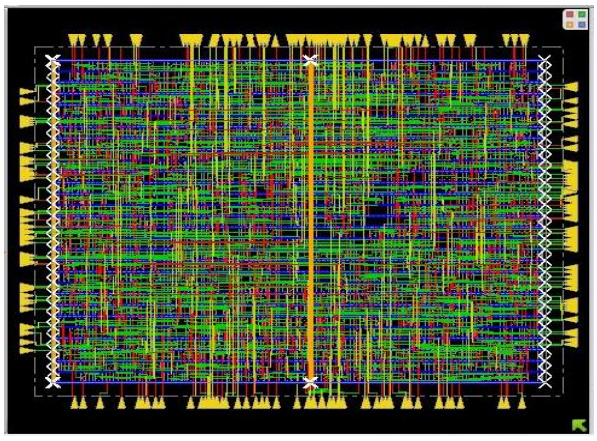

Fig. 8. Placement and Routing(PNR) done for 64 bit CAM on Innovus tool.

C. 64-bit SRAM on the open-source environment

Open-source tools are mainly a solution to the new VLSI generation as they can be downloaded free of cost and there is always a community working on the development of the tools. The open-source environment has been a newly emerging tool in the VLSI field to generate gds2 for the design to overcome the commercial problem of money to students and design understanding enthusiasts. In this section the discussion of implementation of 64-bit SRAM on open-source environment using tools such as Yosys, OpenSta, Qflow and Magic.

Fig 9 shows the cell placement of 64-bit SRAM on the GRAYWOLF tool available as an open-source for generating the area layout of the design SRAM. The cell placement is done following the rules of the design so that, it doesn't encounter any DRC and LVS. Thus, the tool is efficient in understanding of basic tool flow of the design.

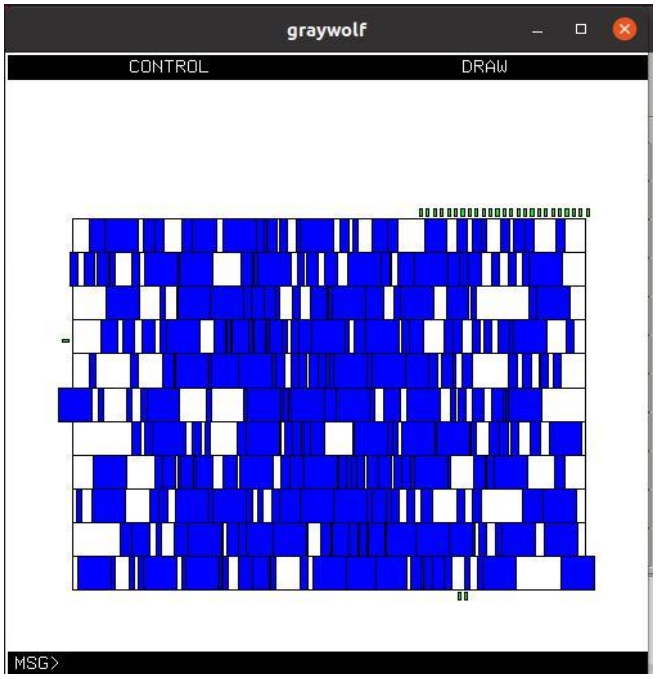

Fig. 9. Cell placement of 64 bit SRAM on Graywolf tool.

The yosys tool is used to generate synth.v file which will later be used to calculate power and area report for SRAM design. The SRAM.ys file should contain a design file, technology file path so that it can generate synth.v file. This file is later used for calculating the power and area of the design.

Fig 10 shows the placement and routing for SRAM on the MAGIC tool. The placement of different cells and routing the wires to proper cells and gates have been shown in fig 10. This tool has explored the new connectivity for the open-source environment and gives easy privilege to users. Fig 11 shows the QFLOW tool used to generate gds2 for SRAM, the process begins from selecting the design file and technology osu035 to conduct. The process begins from preparation, synthesis, placement, static timing analysis, routing, post-STA, migration, DRC, LVS, and then final generating gds2 file.

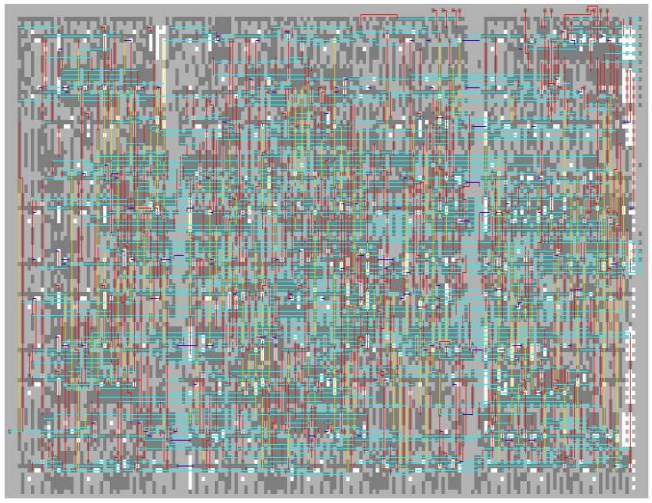

Fig. 10. Placement and Routing did for SRAM using the magic tool.

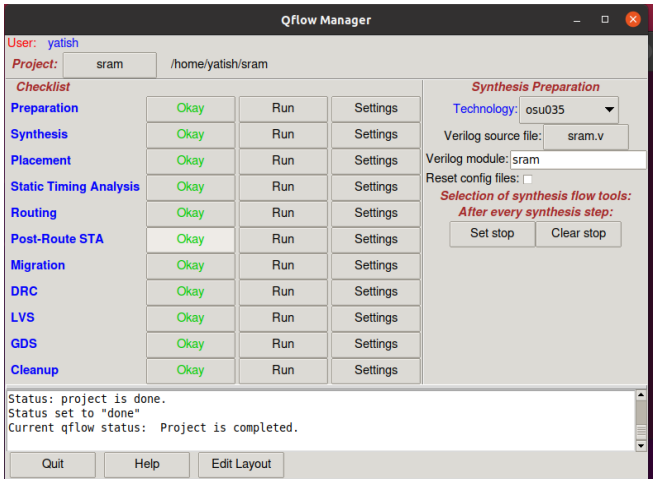

Fig. 11. Generating gds2 using qflow tool for 64 bit SRAM.

D. 64-bit CAM on the open-source environment

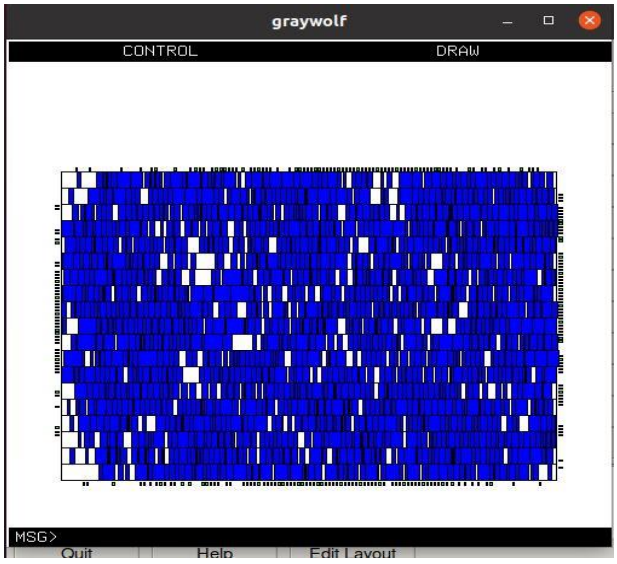

Fig. 12. Placement of cells done on Graywolf tool for 64 bit CAM.

The implementation of CAM in an open-source environment is discussed in this section. fig 12 shows the arrangement of CAM cells in the layout. The arrangement is done using the GRAYWOLF tool for 64-bit CAM. The PnR of CAM is shown in figure 13 using the MAGIC tool of open-source environment.

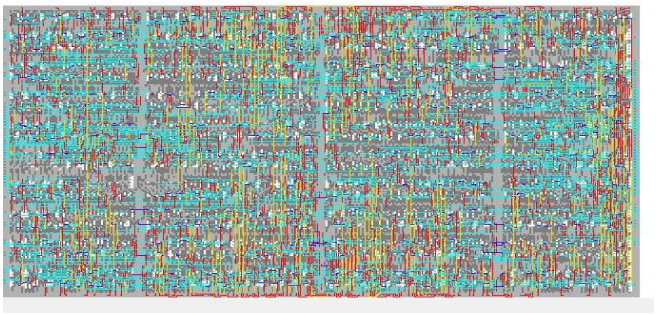

Fig. 13. Placement and Routing(PnR) done for 64 bit CAM using a magic tool.

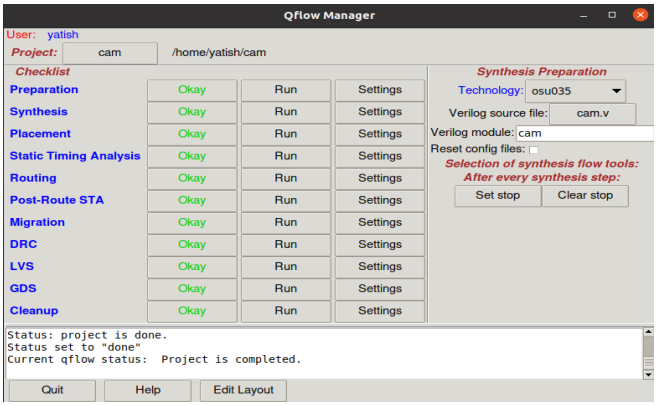

Fig. 14. Generation of gds2 for 64 bit CAM using qflow tool.

The QFLOW tool is used to process the technology osu035 for the design and then synthesis of the CAM is also done using the same design file. fig 14 shows the process that happens in the qflow tool to generate the gds2 file for CAM. Once, the placement is done i.e the allotment of the pins on the different sides of the chip to reduce the connectivity and unnecessary bias. The DRC(design rule check) and layout versus schematic are checked if the design is anyway violating. Then as the last step gds2 file is generated.

Qrouter accepts and outputs files in the open standard LEF and DEF formats. It examines the geometry for each cell to detect contact points and route impediments, using cell definitions from a LEF file. It then reads a DEF file for cell location, pin placement, and netlist, runs the detailed route, and outputs an annotated DEF file. Qflow is a complete toolchain for synthesizing digital circuits, from Verilog source through physical layout for a specific fabrication method. Digital synthesis with a goal application of a chip design is frequently incorporated into large EDA software packages like Cadence or Synopsys in the commercial electronics market. Commercial toolchains are becoming increasingly expensive as commercial electronics designers strive for cutting-edge performance, and have largely priced themselves out of all but the most established integrated circuit makers. This creates an undesirable void in which small enterprises and startups cannot afford to perform any type of integrated circuit design. Digital synthesis algorithms are completely locked up in closed-source software, and development is monopolized by a few EDA software companies.

V. Results and Discussion

As in the previous discussion of implementing SRAM and CAM on both cadence and open-source tool, got the values of power, area, and delay. Table I and Table II depicts the values obtained from both the environment for 64-bit SRAM and CAM. The pre-layout values of power, area, and slack for SRAM and CAM are calculated using the GENUS tool of cadence environment. The post-layout is calculated after generating gds2 using the INNOVUS tool from cadence.

For calculating the power, area, and slack from an open-source environment, using OPENSTA tool for calculated the power and slack of the designs. YOSYS tool is used to calculate the area of the chip in μm^2.

TABLE I. SRAM VALUES COMPARISON OF POWER, AREA, AND SLACK

Sl. No.	SRAM	Cadence environment		Open-source environment
		Pre-layout	Post-layout	
1.	Power(mW)	0.145	0.431	1.78
2.	Area(μm^2)	1104.3648	1868	55808
3.	slack	6636	5.672	17.84

. Table I shows the power values in mW for 64-bit SRAM. The power value is comparatively less in the cadence tool with a ~10% saving to that of the open-source tool. Similarly, the area can also be saved by ~50% when cadence tools are used for demonstrating. As both power and area have the upper hand in using cadence tools, therefore it is always preferable to use cadence tools to develop gds2 for 64-bit SRAM for their performance analysis. The osu035 technology is used to generate the report for 64-bit SRAM on the open-source tools. From the report we can conclude that the wires/nets used to connect the internal blocks of SRAM is 1259 and wire bits is 1559. The number of public wires is 6 and the total number of public wire nets is 299. The obtained internal power is 85.0497% of the total obtained power. It is better to reduce the internal power to reduce the total power. Also, the sequential power in the block takes the highest power i.e., 95.98% so reducing the power used by the sequential circuits will definitely be able to reduce the overall power used by SRAM after post-layout on cadence environment. The power obtained for SRAM is 1.78mW which comprises of internal power of 1.5mW and switching power of 0.286mW. The most of the power is used from the internal circuits. So, overall power of the block can be reduced if the internal power is taken care because it takes the power of almost 84% of the total power. Similarly, the power taken from the sequential block is 75% as obtained on open-source tools has the internal power is more. So, it better

to take care of the internal power that is generated from the sequential block.

Table II shows the different values obtained for the parameters such as power, area, and slack for 64-bit CAM. The power values are almost the same for CAM on both the environment with just 2% variation for the post-layout value of cadence environment to that of one obtained from the OpenSta tool. The area of CAM is increased ~18% in open-source tools to that compared to the value obtained using the innovus tool. Also, the timing values that are obtained have met the slack in both the environment. The timing reports for CAM in an open-source environment are measured by the OpenSta tool. The slack is positive as shown i.e., 2. The input delay is 2000ns and the data path delay is 7852ns. Fortunately, the setup time for the system is 136ns. The elements are considered to be on the critical path when the delay between them exceeds the clock cycle time. When the path delay exceeds the clock cycle delay, the circuit will not work, therefore the logic design engineer's responsibility is to redesign the circuit to eliminate the timing failure (and therefore the critical path). The maximum delay in all of the many register-to-register paths is likewise defined by the critical path, and it does not have to be bigger than the clock cycle time. So carefully the paths must be chosen to calculate the timing of the overall circuit.

TABLE II. CAM VALUES COMPARISON OF POWER, AREA, AND SLACK

Sl. No.	CAM	Cadence environment		Open-source environment
		Pre-layout	Post-layout	
1.	Power(mW)	0.202	0.8287	0.832
2.	Area(μm^2)	13861.1	13814.15	152316
3.	slack	2	2.65	4.74

VI. CONCLUSION AND FUTURE SCOPE

After implementing the design on both the environment of VLSI tools concluding the following. The power, area, and slack calculated have better values i.e., 0.145mW, 1104.3μm^2, and positive slack of 6636 for SRAM in cadence environment. Also, the power values for 64-bit CAM in cadence environment are almost the same as that of open-source environment i.e., 0.8mW. The calculated slack for CAM has a better value on the open-source environment of 4.74 value. Cadence environment has the upper hand when it comes to implementing the design and generating gds2 file as the environment is too old and well-experienced in handling the tough corners of the design irrespective of the number of bits used.

The open-source tool as of today has given good results for the same design. In near future, there is a chance of having better results because of the continuous improvement happening on the open-source tools in the VLSI market. Thus, appreciating the open source tools such as yosys, OpenSta, qflow, Graywolf, and magic tools currently competing with the commercial tools available in the industry.

VII. ACKNOWLEDGEMENT

The authors thank the Principal and HOD of the Dept. of ECE for the kind support extended to us on the project completion.

REFERENCES

[1] N. Surana and J. Mekie, "Energy Efficient Single-Ended 6-T SRAM for Multimedia Applications," in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 66, no. 6, pp. 1023-1027, June 2019, doi: 10.1109/TCSII.2018.2869945.

[2] D. Shah, E. Hung, C. Wolf, S. Bazanski, D. Gisselquist and M. Milanovic, "Yosys+nextpnr: An Open Source Framework from Verilog to Bitstream for Commercial FPGAs," 2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), 2019, pp. 1-4, doi: 10.1109/FCCM.2019.00010.

[3] Z. Jiang, S. Yin, M. Seok and J. Seo, "XNOR-SRAM: In-Memory Computing SRAM Macro for Binary/Ternary Deep Neural Networks," 2018 IEEE Symposium on VLSI Technology, 2018, pp. 173-174, doi: 10.1109/VLSIT.2018.8510687.

[4] T. -W. Huang, G. Guo, C. -X. Lin and M. D. F. Wong, "OpenTimer v2: A New Parallel Incremental Timing Analysis Engine," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 40, no. 4, pp. 776-789, April 2021, doi: 10.1109/TCAD.2020.3007319.

[5] E. V. Kuliev, V. V. Kureichik and I. O. Kursitys, "Decision making in VLSI components placement problem based on grey wolf optimization," 2019 IEEE East-West Design & Test Symposium (EWDTS), 2019, pp. 1-4, doi: 10.1109/EWDTS.2019.8884371.

[6] D. Bhattacharya, A. N. Bhoj and N. K. Jha, "Design of Efficient Content Addressable Memories in High-Performance FinFET Technology," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 23, no. 5, pp. 963-967, May 2015, doi: 10.1109/TVLSI.2014.2319192.

[7] S. Jeloka, N. B. Akesh, D. Sylvester and D. Blaauw, "A 28 nm Configurable Memory (TCAM/BCAM/SRAM) Using Push-Rule 6T Bit Cell Enabling Logic-in-Memory," in IEEE Journal of Solid-State Circuits, vol. 51, no. 4, pp. 1009-1021, April 2016, doi: 10.1109/JSSC.2016.2515510.

[8] L. Sterpone and A. Ullah, "On the optimal reconfiguration times for TMR circuits on SRAM based FPGAs," 2013 NASA/ESA Conference on Adaptive Hardware and Systems (AHS-2013), 2013, pp. 9-14, doi: 10.1109/AHS.2013.6604220.

[9] B. N. K. Reddy, K. Sarangam, T. Veeraiah and R. Cheruku, "SRAM cell with better read and write stability with Minimum area," TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON), 2019, pp. 2164-2167, doi: 10.1109/TENCON.2019.8929593.

[10] S. N. Shahsavani, T. Lin, A. Shafaei, C. J. Fourie and M. Pedram, "An Integrated Row-Based Cell Placement and Interconnect Synthesis Tool for Large SFQ Logic Circuits," in IEEE Transactions on Applied Superconductivity, vol. 27, no. 4, pp. 1-8, June 2017, Art no. 1302008, doi: 10.1109/TASC.2017.2675889.

[11] J. Lu and B. Taskin, "From RTL to GDSII: An ASIC design course development using Synopsys® University Program," 2011 IEEE International Conference on Microelectronic Systems Education, 2011, pp. 72-75, doi: 10.1109/MSE.2011.5937096.

[12] T. Venkata Mahendra, S. Wasmir Hussain, S. Mishra and A. Dandapat, "Energy-Efficient Precharge-Free Ternary Content Addressable Memory (TCAM) for High Search Rate Applications," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 67, no. 7, pp. 2345-2357, July 2020, doi: 10.1109/TCSI.2020.2978295.

[13] Y. Jinsong, W. Chunlu and L. Chuanyi, "Performance Comparisons of a Content-Addressable Storage Network System and Other Typical IP-SAN Based Storage Systems," 2011 Fourth International Conference on Intelligent Computation Technology and Automation, 2011, pp. 1142-1145, doi: 10.1109/ICICTA.2011.571.

[14] V. M. Telajala, W. H. Sheikh, S. Mishra and A. Dandapat, "A Low-Power Split-Controlled Single Ended Storage Content Addressable Memory," 2019 IEEE International Symposium on Smart Electronic Systems (iSES) (Formerly iNiS), 2019, pp. 369-372, doi: 10.1109/iSES47678.2019.00091.

[15] M. Irfan and A. Ahmad, "Impact of Initialization on Gate-Based Area Efficient Ternary Content-Addressable Memory," 2018 International Conference on Computing, Electronics & Communications Engineering (iCCECE), 2018, pp. 328-332, doi: 10.1109/iCCECOME.2018.8658961.

[16] K. Hinsen, "The Magic of Content-Addressable Storage," in Computing in Science & Engineering, vol. 22, no. 3, pp. 113-119, 1 May-June 2020, doi: 10.1109/MCSE.2019.2949441.

[17] D. Cho, K. Kim and C. Yoo, "A Non-Volatile Ternary Content-Addressable Memory Cell for Low-Power and Variation-Toleration Operation," in IEEE Transactions on Magnetics, vol. 54, no. 2, pp. 1-3, Feb. 2018, Art no. 9300103, doi: 10.1109/TMAG.2017.2763579.

[18] J. -W. Lee et al., "A 1.8 Gb/s/pin 16Tb NAND Flash Memory Multi-Chip Package with F-Chip of Toggle 4.0 Specification for High Performance and High Capacity Storage Systems," 2020 IEEE Symposium on VLSI Circuits, 2020, pp. 1-2, doi: 10.1109/VLSICircuits18222.2020.9163052.

[19] M. Saitoh et al., "Emerging Non-Volatile Memory and Thin-Film Transistor Technologies for Future 3D-LSI," 2018 48th European Solid-State Device Research Conference (ESSDERC), 2018, pp. 138-141, doi: 10.1109/ESSDERC.2018.8486916.

[20] M. Janai and I. Bloom, "Charge Gain, NBTI, and Random Telegraph Noise in EEPROM Flash Memory Devices," in IEEE Electron Device Letters, vol. 31, no. 9, pp. 1038-1040, Sept. 2010, doi: 10.1109/LED.2010.2058088.

[21] Cherry Bhargava; Gaurav Mani Khanal, "Basic VLSI Design Technology: Technical Questions and Solutions," in Basic VLSI Design Technology: Technical Questions and Solutions , River Publishers, 2020, pp.i-xx.

[22] Cherry Bhargava; Gaurav Mani Khanal, "2 IC Fabrication Technology," in Basic VLSI Design Technology: Technical Questions and Solutions , River Publishers, 2020, pp.91-112.

[23] Cherry Bhargava; Gaurav Mani Khanal, "4 Verilog Hardware Descriptive Language," in Basic VLSI Design Technology: Technical Questions and Solutions , River Publishers, 2020, pp.153-200.

[24] H. Chu et al., "An ASIC Design of Multi-Electrode Digital Basket Catheter Systems with Reconfigurable Compressed Sampling," 2018 31st IEEE International System-on-Chip Conference (SOCC), 2018, pp. 124-129, doi: 10.1109/SOCC.2018.8618535.

[25] C. Zhao, Y. Yan and W. Li, "An efficient ASIC Implementation of QARMA Lightweight Algorithm," 2019 IEEE 13th International Conference on ASIC (ASICON), 2019, pp. 1-4, doi: 10.1109/ASICON47005.2019.8983618.

Dr. Shylashree N is currently working as Associate Professor in the Department of Electronics and Communication Engineering at RV College of Engineering, Bengaluru. She is having 15 years of teaching experience. She was a recipient of the best PhD thesis award for the year 2016-2017 in Electronics and Communication Engineering from BITES. She has received the best IEEE researcher award in IEEE-AGM meeting held during 2021 from Bangalore IEEE section. She has also received the best paper award in IEEE-ICERECT held during 2015 at Mandya. She has research publication in 18 International Journals (out of which 5 journals are Springer-SCI journals), 4 Springer book chapters and 9 International conferences. She has published 2 patents and filed 1 German patent in the area of cryptography. She has also published 2 patents in the area of VLSI out of which one of them got a grant. She is also the author of the Network Theory textbook, Engineering Statistics & Linear Algebra text book and Control Engineering textbook. She has funded project on chalcogenide materials & consultancy projects on FPGA and has delivered many technical talks on VLSI. She has delivered lectures as a subject matter expert in VTU e-shikshana and EDUSAT program. She is a recipient of international travel grant under SERB young research scholar category. She is a life member of ISTE, IETE, fellow member of ISVE, Senior member of IEEE and IEEE-CAS Bangalore section execom member

Mr. Yatish D Vahvale is a post graduate student of VLSI Design and Embedded systems. He is currently pursuing his final year in Master of Technology degree at RV College of Engineering, Bengaluru, India. His areas of interest are VLSI, Analog and Digital design and IC manufacturing.

Prof. Praveena N is a part-time research scholar in the Department of Electronics and Communication Engineering at RV College of Engineering, Bengaluru and also she is currently working as an Assistant Professor in the Department of Electronics and Communication Engineering at Sir MVIT, Bangalore. She has 11 years of teaching experience. Her areas of interest are VLSI, Embedded systems, HDL and Nano electronics etc. She has completed BE and M.Tech from VTU in the year 2004 and 2008 respectively.

Dr. Mamatha A.S is currently working as Associate Professor in the Department of Electronics and Communication Engineering at St.Joseph Engineering College, Mangalore. She has 22 years of teaching experience. She is the author of four international journals and six international conferences in the field of Multispectral Image Compression. She is the author of the Network Theory textbook, Engineering Statistics & Linear Algebra textbook and Control Engineering text book. Her areas of interest are Signal Processing, Image Compression, and Control Engineering etc. She is a Senior IEEE member