IIIT-B Chip Design Studio Report

Team Name/Project:
ASIC-Based VLSI Architecture for Support Vector Machine

Team Members:

Name	Email ID	Current Status	Institute
Arun M	arun.m004@iiitb.ac.in	Student(MS by Research)	IIITB
Vaishnavi Sharma	vaishnavi.sharma@iiitb.ac.in	Student(M.Tech)	IIITB

Supervisor: Prof. Madhav Rao, IIITB

1. Problem Statement

The increasing prevalence of sleep apnea necessitates efficient and real-time detection methods, as traditional polysomnography is expensive and impractical for continuous monitoring. This project presents an ASIC-based Support Vector Machine (SVM) inference engine designed to classify apnea events using physiological signals such as ECG. Unlike software-based implementations, which are computationally expensive and power-hungry, this ASIC provides a low-power, high-speed alternative tailored for real-time classification. The architecture is optimized for efficient SVM kernel computation and decision functions, ensuring minimal latency while maintaining high classification accuracy. A key feature of this design is its scalability, enabling the same hardware framework to be extended beyond sleep apnea detection to classify multiple datasets from different domains. The ASIC efficiently processes input features, performs classification using the SVM model, and outputs real-time predictions, making it suitable for deployment in embedded and wearable healthcare devices. Key challenges addressed in this project include hardware optimization, feature extraction, resource-efficient implementation of SVM computations, and scalability. The inference engine has been tested with different datasets to demonstrate its flexibility, showing promising results in medical and non-medical classification tasks.

2. GitHub Link

Repository: https://github.com/ArunM2402/svm$_a$*rchitecture.git*

3. Introduction

Support Vector Machine:

A Support Vector Machine (SVM) is a powerful supervised machine learning algorithm used for classification and regression tasks, particularly effective in high-dimensional spaces. It operates by finding an optimal hyperplane that best separates data points belonging to different classes while maximizing the margin between them. The key idea behind SVM is to transform input data into a higher-dimensional space using a kernel function, such as linear, polynomial, or radial basis function (RBF), to make complex patterns more separable. The algorithm selects a subset of critical data points, called support vectors, which define the decision boundary, ensuring robustness and generalization to unseen data. Unlike traditional classifiers, SVM is highly effective in handling non-linearly separable data and avoids overfitting, making it ideal for

applications like medical diagnosis, image recognition, and financial forecasting. The training phase involves solving a convex optimization problem to determine the optimal hyperplane, while the inference phase involves simple dot-product operations, making it suitable for efficient hardware implementation in ASIC or FPGA-based systems. Additionally, the use of kernel trick allows SVM to model complex decision boundaries without explicitly computing high-dimensional feature transformations.
Kernel:

In Support Vector Machines (SVMs), a kernel function is used to transform input data into a higher-dimensional space where it becomes easier to classify using a linear boundary. This transformation, known as the "kernel trick," allows SVM to handle non-linearly separable data efficiently without explicitly computing high-dimensional transformations. Common kernel functions include the linear kernel for simple separable data, polynomial kernel for capturing complex feature interactions, Radial Basis Function (RBF) kernel for mapping data into an infinite-dimensional space, and sigmoid kernel, which mimics neural network activation functions.
Support Vectors:

Support vectors play a crucial role in defining the optimal hyperplane. These are the data points closest to the decision boundary, forming the margin that SVM maximizes for better classification. Only these points influence the hyperplane, making SVM a sparse model that efficiently generalizes to unseen data. The presence of more support vectors indicates a complex model, while fewer support vectors lead to simpler and more generalized decision boundaries. Since support vectors lie on or near the margin boundaries, they ensure robust decision-making even in noisy datasets.

4. Methodology and Implementation

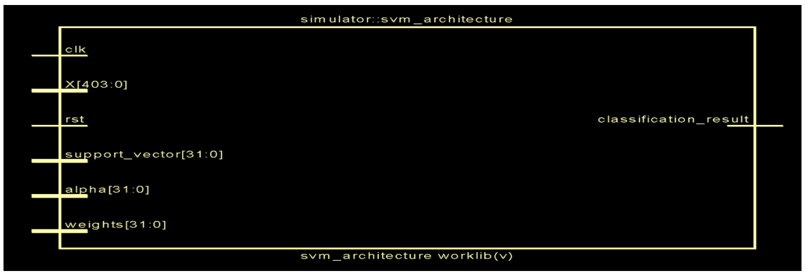
The entire project can be divided into two parts: software and hardware.
Software:

- In the software phase,the training of the SVM engine is done.
- The ECG dataset was obtained from the the standard Physionet dataset which comprises of data recordings of various ECG signals.
- Data pre-processing was performed to reduce the number of features. Initially, a repeated correlation matrix elimination method was used to systematically eliminate features having correlation value of more than 0.75.
- Later, PCA (Principal Component Analysis) is performed to select the best features required for the model.
- This helped in reducing the number of features from 66 to 13.
- A SVM model using a simple linear kernel was constructed.The kernel was chosen in accordance to maintaining less hardware complexity at the cost of some accuracy (negligible loss).
- Finally, the dataset was split into training and test datasets and accuracy is measured.

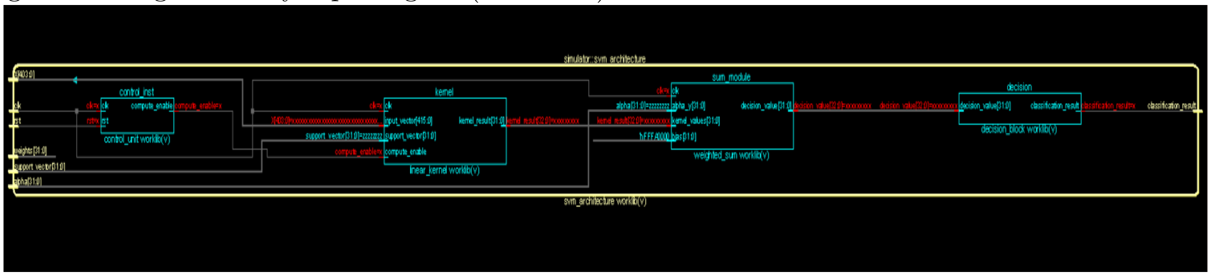
Hardware:

- The hardware phase mainly comprises of building the inference engine.

- It consists of a kernel block, weighted sum block, control unit and a decision maker block.
- The linear kernel block basically transforms the data into a higher dimension for easier classification.
- The weighted sum module is a MAC unit which accounts for the effect of support vectors, weights and alphas on the input.
- The decision block is a simple comparator which sets the classification output based on the threshold value.
- The control unit makes sure that proper synchronization takes place between the blocks during data transfer.
- The support vectors, alphas and weights obtained from the training phase are fed as inputs to the inference engine along with the test inputs.

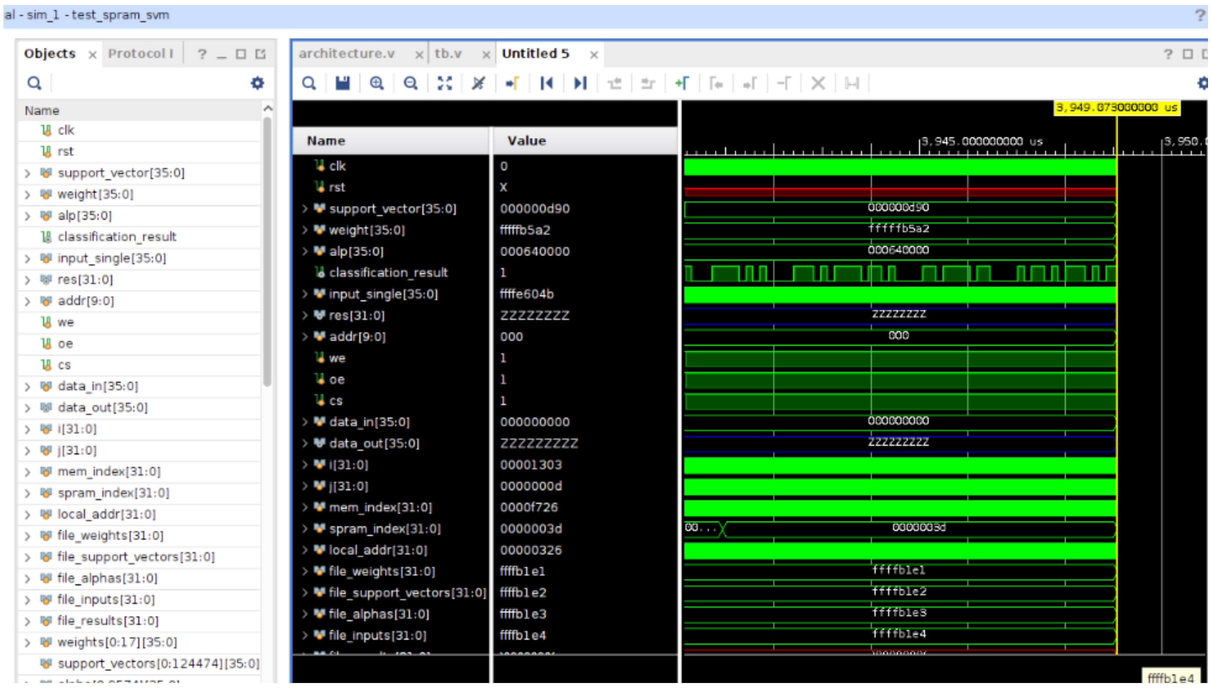


The above blackbox can be elaborated as shown below. Note that the 'x' signals are due to the design not being driven by input signals(idle state)



5. Simulation Results:

The design was tested in Vivado for a input dataset of shape (4867,13) and the waveforms are shown below:



```
Name                              Value
clk                               0
rst                               X
support_vector[35:0]              000000d90
weight[35:0]                      fffffb5a2
alp[35:0]                         000640000
classification_result             1
input_single[35:0]                ffffe604b
res[31:0]                         ZZZZZZZZ
addr[9:0]                         000
we                                1
oe                                1
cs                                1
data_in[35:0]                     000000000
data_out[35:0]                    ZZZZZZZZZ
i[31:0]                           00001303
j[31:0]                           0000000d
mem_index[31:0]                   0000f726
spram_index[31:0]                 0000003d
local_addr[31:0]                  00000326
file_weights[31:0]                ffffb1e1
file_support_vectors[31:0]        ffffb1e2
file_alphas[31:0]                 ffffb1e3
file_inputs[31:0]                 ffffb1e4
```

The values are fed serially hence a valid classification output is obtained after dividing by number of features. The results were extracted and the accuracy was found to be 77.4 percent.

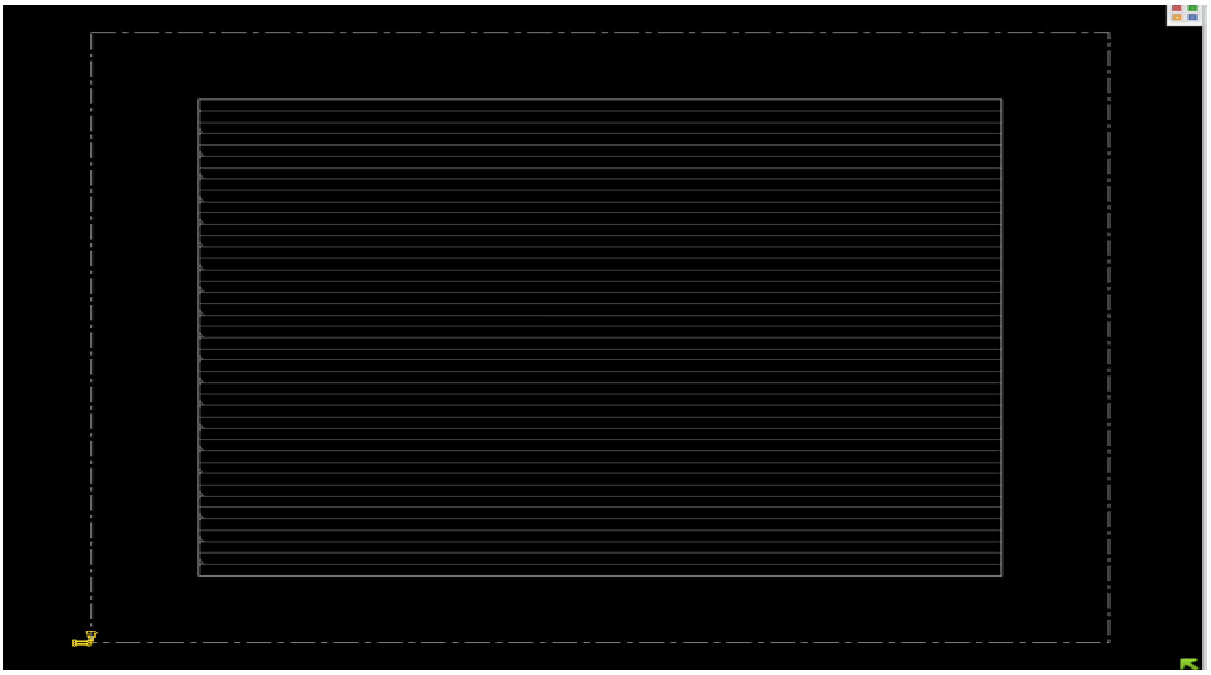
6. Synthesis Results:

The design was synthesized in Cadence Genus using 45nm technology node and the following results were observed.

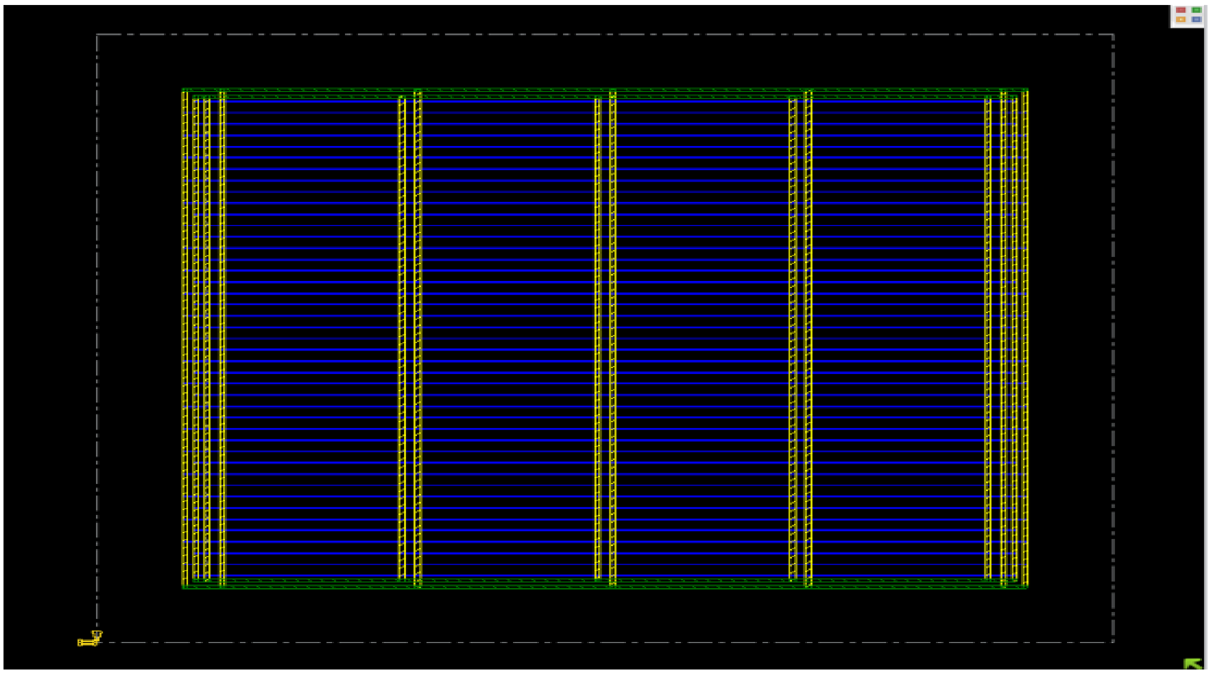
Parameter	Value
Area	5762.152 um^2
Gates	1349
Power	234.968 nW

7. Physical Design:

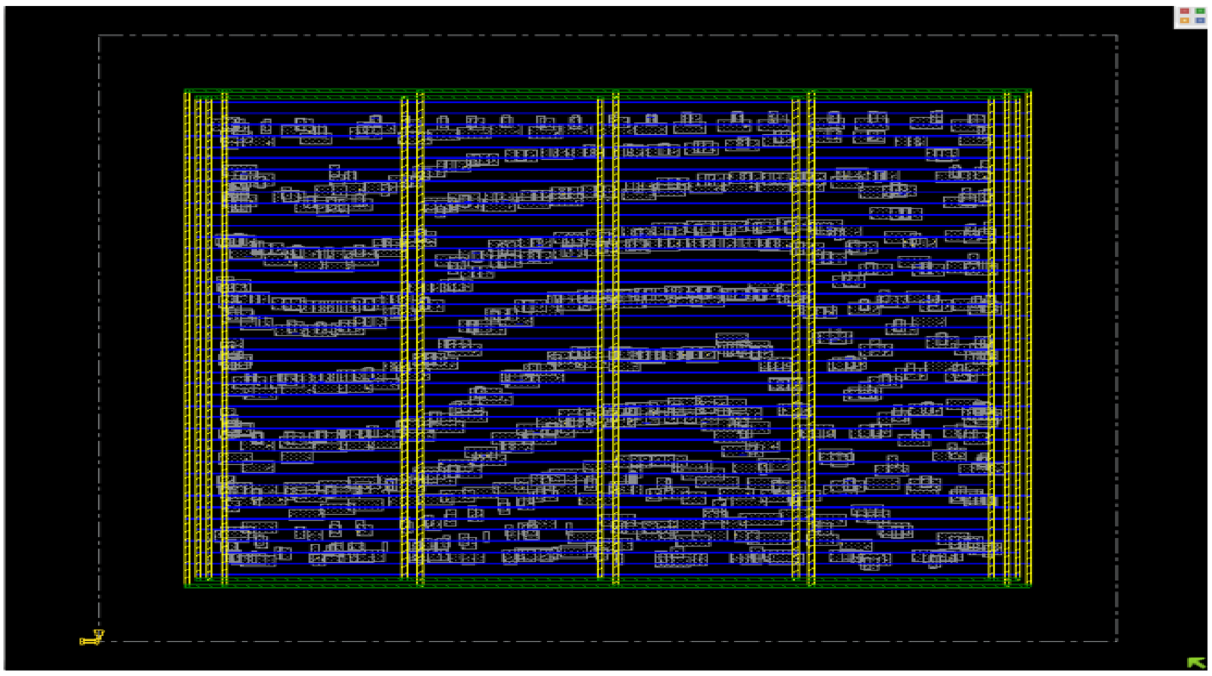
- The optimized netlist file after synthesis is read for this part of the project.
- The .lef and .lib files are read from the 45nm technology directory.
- For setup and hold analysis, RC corners are created for fast and slow cells. For this step, the corresponding capTable and QRC files are read.
- The floorplan is then created for a aspect ratio of 1 and core utilization of 0.7.

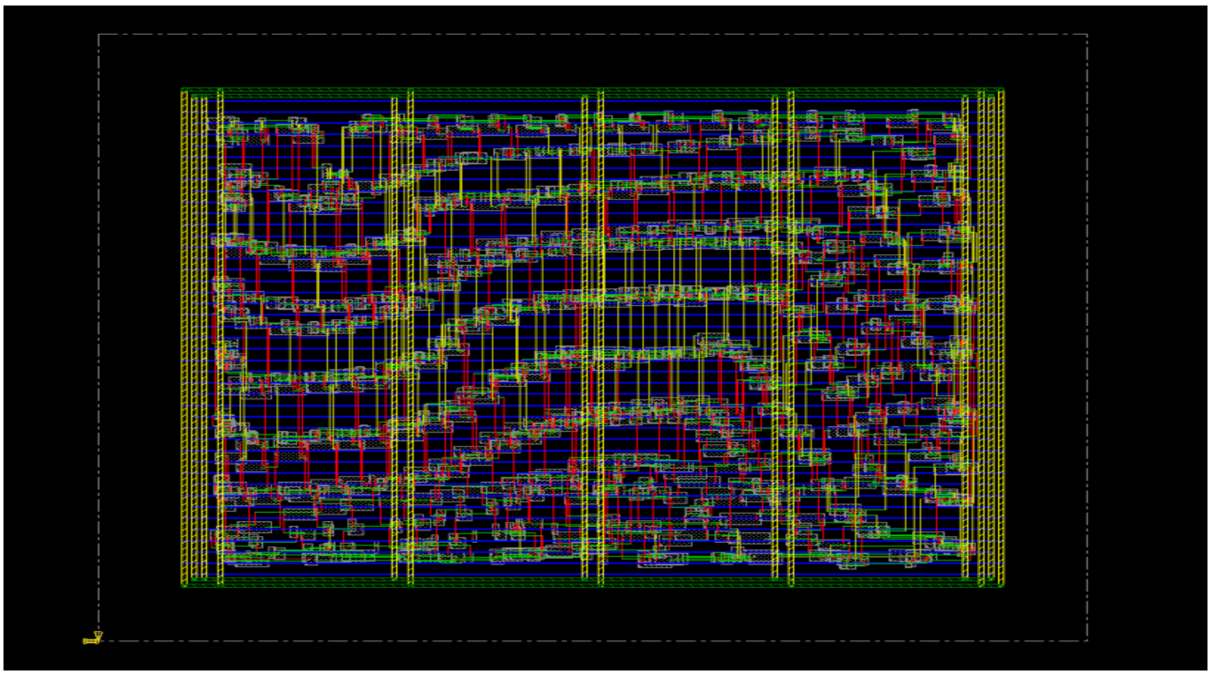


- The next step is Power Planning which is done to ensure efficient power distribution, prevent excessive voltage drops, and manage thermal dissipation. This is done by adding core rings and stripes of power pins (VDD and VSS).

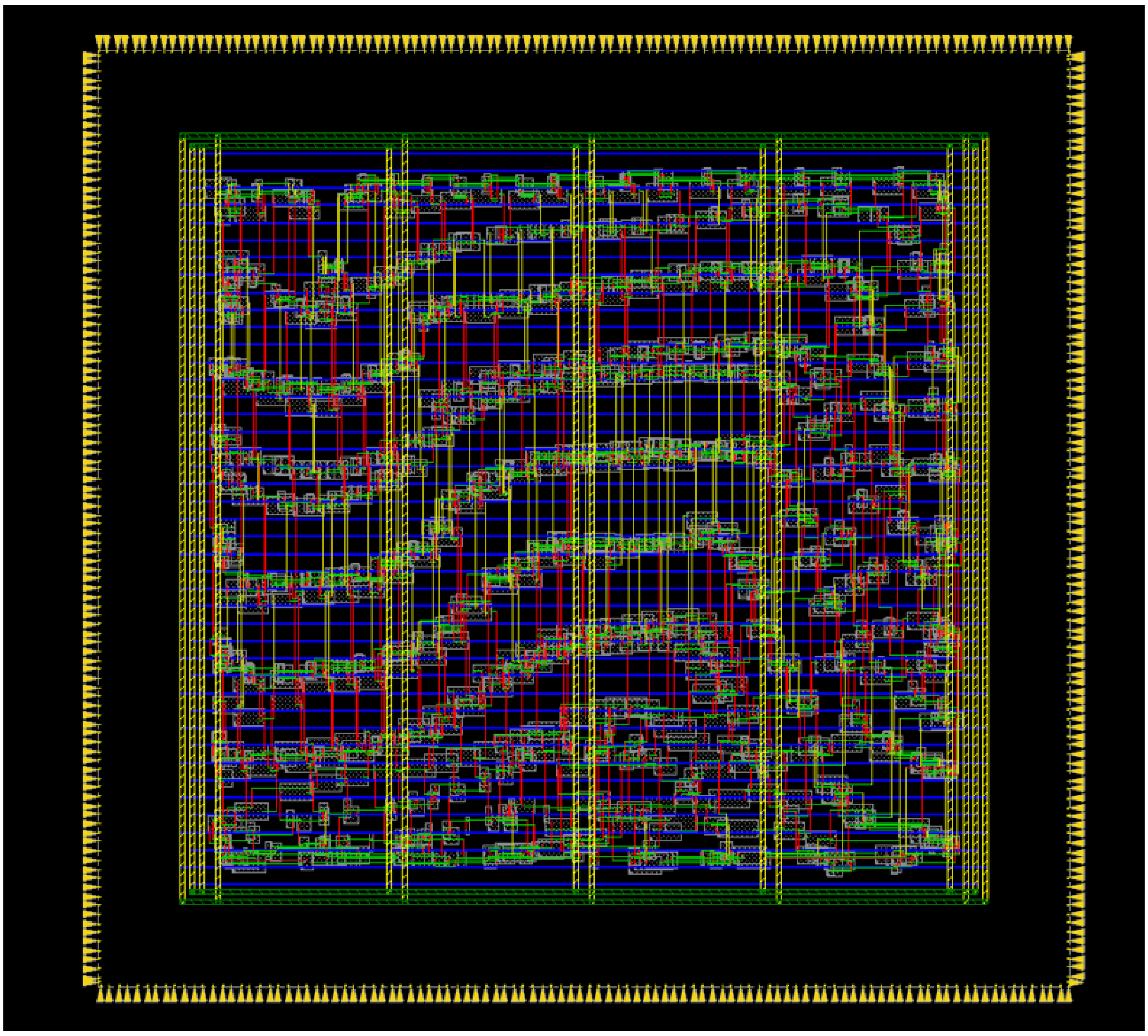


- The standard cells are then placed by referring to the synthesized netlist and the attached .lef file of the technology node.

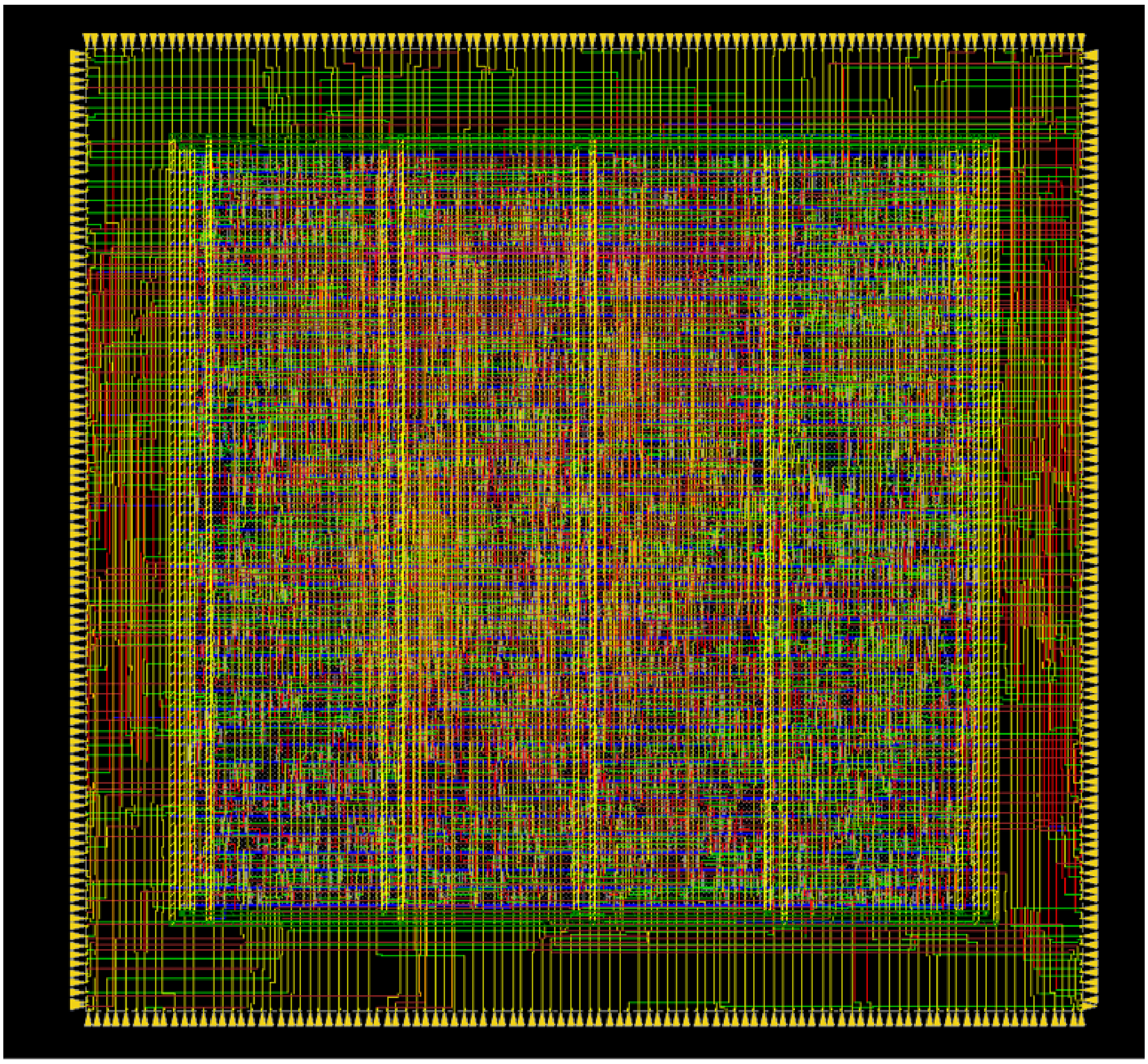




- The pins are then found by parsing the netlist and spread in an optimized manner around the core.



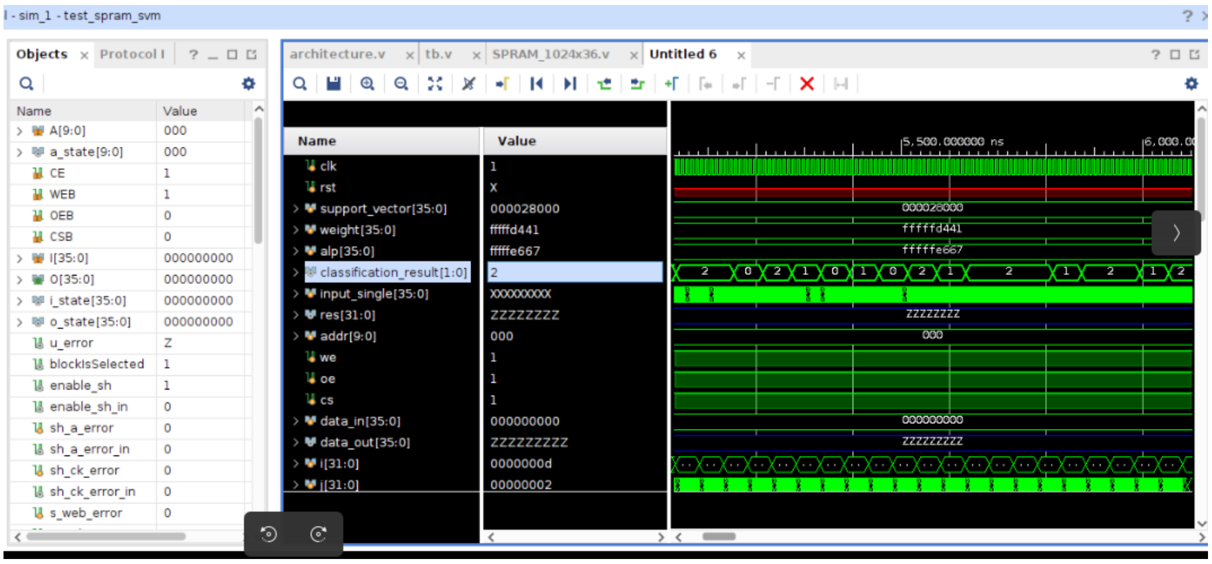
- Optimization and Clock Tree Synthesis (CTS) is done at each stage.CTS is a critical step in ASIC physical design that ensures a well-balanced and optimized clock distribution across the entire chip. The primary goal of CTS is to minimize clock skew and latency, ensuring that all sequential elements (flip-flops and registers) receive the clock signal in a synchronized manner.
- Placement of clock buffers and inverters is done to drive the clock signal efficiently while balancing the load and reducing clock jitter.
- The next stop is global routing. Global Routing determines the approximate paths for interconnections while optimizing congestion and minimizing detours.



- After checks, the final GDS file is extracted. It is attached in the github link mentioned earlier.

8. Extension of the work

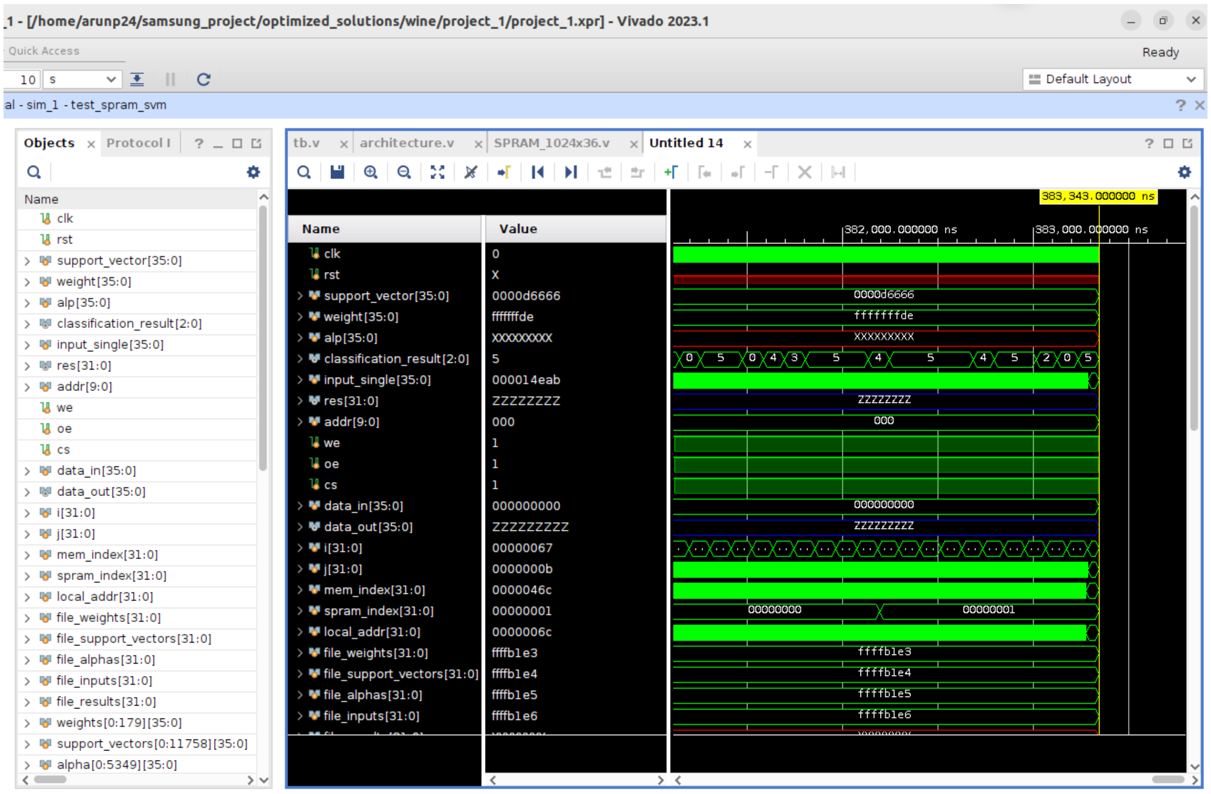
The same SVM engine was tried on various datasets so as to check the generic property of the inference engine. A few example datasets are shown below:
1) Iris dataset: The simulation results are as shown:



Name	Value
A[9:0]	000
a_state[9:0]	000
CE	1
WEB	1
OEB	0
CSB	0
I[35:0]	000000000
O[35:0]	000000000
i_state[35:0]	000000000
o_state[35:0]	000000000
u_error	Z
blockIsSelected	1
enable_sh	1
enable_sh_in	0
sh_a_error	0
sh_a_error_in	0
sh_ck_error	0
sh_ck_error_in	0
s_web_error	0

Name	Value
clk	1
rst	X
support_vector[35:0]	000028000
weight[35:0]	fffffd441
alp[35:0]	fffffe667
classification_result[1:0]	2
input_single[35:0]	XXXXXXXXX
res[31:0]	ZZZZZZZZ
addr[9:0]	000
we	1
oe	1
cs	1
data_in[35:0]	000000000
data_out[35:0]	ZZZZZZZZZ
i[31:0]	0000000d
j[31:0]	00000002

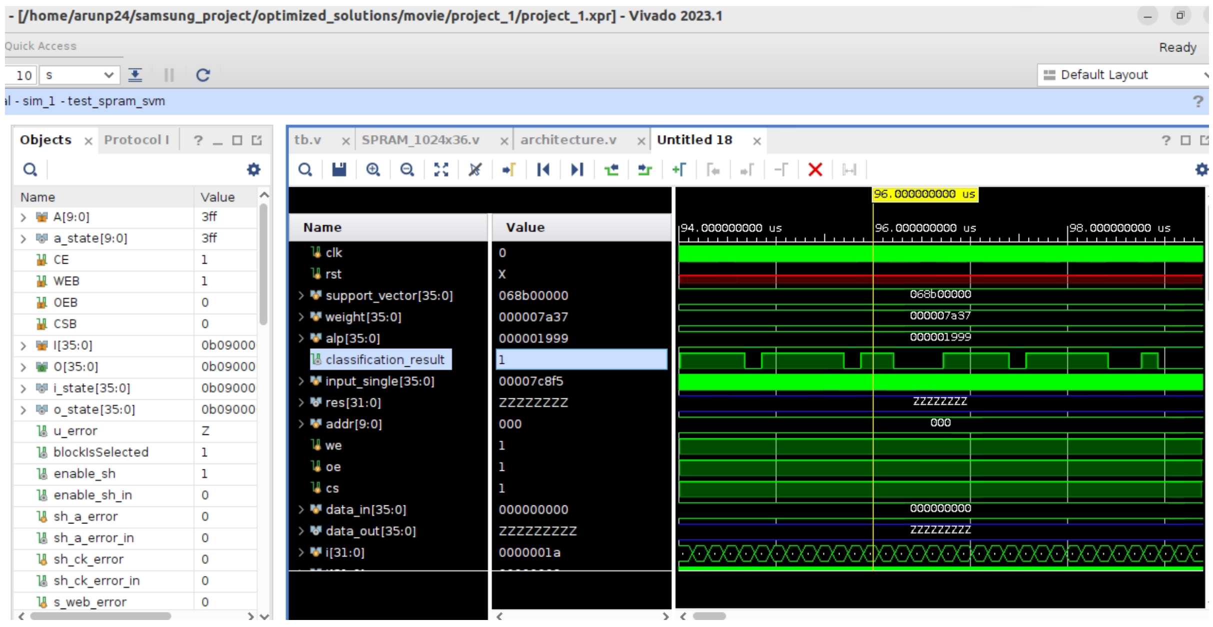
2) Wine dataset: The simulation results are as shown:



Name
clk
rst
support_vector[35:0]
weight[35:0]
alp[35:0]
classification_result[2:0]
input_single[35:0]
res[31:0]
addr[9:0]
we
oe
cs
data_in[35:0]
data_out[35:0]
i[31:0]
j[31:0]
mem_index[31:0]
spram_index[31:0]
local_addr[31:0]
file_weights[31:0]
file_support_vectors[31:0]
file_alphas[31:0]
file_inputs[31:0]
file_results[31:0]
weights[0:179][35:0]
support_vectors[0:11758][35:0]
alpha[0:5349][35:0]

Name	Value
clk	0
rst	X
support_vector[35:0]	0000d6666
weight[35:0]	fffffffde
alp[35:0]	XXXXXXXXX
classification_result[2:0]	5
input_single[35:0]	000014eab
res[31:0]	ZZZZZZZZ
addr[9:0]	000
we	1
oe	1
cs	1
data_in[35:0]	000000000
data_out[35:0]	ZZZZZZZZZ
i[31:0]	00000067
j[31:0]	0000000b
mem_index[31:0]	0000046c
spram_index[31:0]	00000001
local_addr[31:0]	0000006c
file_weights[31:0]	ffffb1e3
file_support_vectors[31:0]	ffffb1e4
file_alphas[31:0]	ffffb1e5
file_inputs[31:0]	ffffb1e6

3) Movie dataset: The simulation results are as shown:


- [/home/arunp24/samsung_project/optimized_solutions/movie/project_1/project_1.xpr] - Vivado 2023.1

l - sim_1 - test_spram_svm

Name	Value
A[9:0]	3ff
a_state[9:0]	3ff
CE	1
WEB	1
OEB	0
CSB	0
I[35:0]	0b09000
O[35:0]	0b09000
i_state[35:0]	0b09000
o_state[35:0]	0b09000
u_error	Z
blockIsSelected	1
enable_sh	1
enable_sh_in	0
sh_a_error	0
sh_a_error_in	0
sh_ck_error	0
sh_ck_error_in	0
s_web_error	0

Name	Value
clk	0
rst	X
support_vector[35:0]	068b00000
weight[35:0]	000007a37
alp[35:0]	000001999
classification_result	1
input_single[35:0]	00007c8f5
res[31:0]	ZZZZZZZZ
addr[9:0]	000
we	1
oe	1
cs	1
data_in[35:0]	000000000
data_out[35:0]	ZZZZZZZZZ
i[31:0]	0000001a

Comparison:
Below shown is a table comprising of the above datasets and their test accuracy calculated using Python and the design in use.

Dataset	Support Vectors	No of features	No of Classes	Software accuracy	Hardware accuracy
Sleep Apnea	9575	13	2	79.8	77.4
Iris	9	5	3	100	73.1
Wine Quality	1069	11	6	56	42
Movie	216	16	2	69.7	58.65

From the table, it is clear the design does work for various classification datasets. However, it depends on number of support vectors since they decide the decision boundaries and therefore right number of classifications. It also needs a lot of improvements when it comes to multiclass classifications.

9. Future scope

We plan to improve the present design by :

- Remodeling the design to work for Posit format. This helps in working with large datasets using lesser bits and hence, lesser hardware resources.
- Improving the kernel computation block to support polynomial, RBF and sigmoid functions.
- Improving the design's performance with multi-class classification problems.

10. Acknowledgment

We thank our supervisor Prof. Madhav Rao for his contribution and timely guidance. We thank the entire team (Samsung IIIT, Bangalore) associated with Chip Design Studio for giving us this opportunity.