Parasitic RC Estimation and Defect Prediction for Embedded Memory using Machine Learning

Venkatesham Maddela
Lovely Professional University
Sanjeet Kumar Sinha (sanjeetksinha@gmail.com)
Lovely Professional University
Muddapu Parvathi
BVRIT HYDERABAD College of Engineering for Women
Sweta Chander
Lovely Professional University

Research Article

Keywords: Parasitic Extraction Method, Open/Short Faults, Linear Regression, Decision Tree, Machine Learning

Posted Date: January 10th, 2024

DOI: https://doi.org/10.21203/rs.3.rs-3843032/v1

Additional Declarations: No competing interests reported.

Parasitic RC Estimation and Defect Prediction for Embedded Memory using Machine Learning

Venkatesham Maddela[1], *Sanjeet Kumar Sinha[2], Muddapu Parvathi[3] and Sweta Chander[4]
[1,2,4]Dept. of School of Electronics and Electrical Engineering, Lovely Professional University, Phagwara, India
[1,3]Department of Electronics and Communication Engineering, BVRIT HYDERABAD College of Engineering for Women, Hyderabad, India
E-mail: mvenkateshkp@gmail.com[1] , sanjeetksinha@gmail.com[2], pbmuddapu@gmail.com[3] & sweta.chander@gmail.com[4]

Abstract

In today's rapidly scaling-down technological environment, identifying the best-fit algorithms for evaluating complicated circuits such as SRAMs is a difficult issue. Many fault models have developed, however their flexibility of use is limited by the restrictions and constraints of the provided test environment. The majority of existing fault models have been studied in terms of well-known March algorithms, which simply provide fault detection information. Scaled-down technologies have an impact on parasitic effects as well, resulting in an extra source of defective behavior and making current test algorithms vulnerable to them. Recent work that uses method of parasitic extraction for fault detection have addressed the problem of limitation due to scale down technologies. However, as the circuit complexity increases the estimation of RC would be tedious. Hence in this paper machine learning based parasitic RC extraction is proposed. Also, as an extension to that, proposed ML based fault detection using extracted parasitic RCs as dataset. The proposed machine learning based fault prediction uses extracted parasitic RCs as dataset. The parasitic RC values are extracted for each fault model using technologies of 120nm down to deep submicron 7nm. Regression algorithm is used for modeling the machine for extraction of RCs and observed that 88% of prediction accuracy. Decision tree modeling is used for fault detection and observed 91.7% of accuracy in prediction of fault.

Key Words: Parasitic Extraction Method; Open/Short Faults; Linear Regression; Decision Tree, Machine Learning

1. Introduction

Memory devices are essential from a quality standpoint as well due to the huge area that SRAMs occupy and their high level of integration. For these reasons, manufacturing tests must be done quickly and accurately in order to find any internal system faults and keep costs in control. The majority of the time, these abnormalities take place inside the memory cell. They might result from resistance or parasitic capacitance between the paths [1, 2].

*Corresponding Author:

Sanjeet Kumar Sinha
E-mail: sanjeetksinha@gmail.com

On the other hand, faults in low-power designs involve behaviors that are challenging for standard March tests to decide up on [3]. The system experiences different impacts depending on how short or open defects between the nodes. The effect of such short/open faults on the behavior of deep submicron 6T-SRAM cells is explored in this work [4-6].

Modern technologies heavily rely on memory testing. The maximal storage density in the smallest possible space is frequently required in memory architecture. Therefore, as technology advances, data storage requirements and system complexity rise, increasing the probability that a system would have defects during manufacturing.

Previous research took into account the various faults that might exist in each SRAM cell [12]; resistive defects are the common defects occur in the memory cell. There are various deviations that may disrupt the memory cell. Some of these can be represented as bridging defects and resistive-opens defects [7-9] on a circuit model. The functional model of a defect is referred to as a fault.

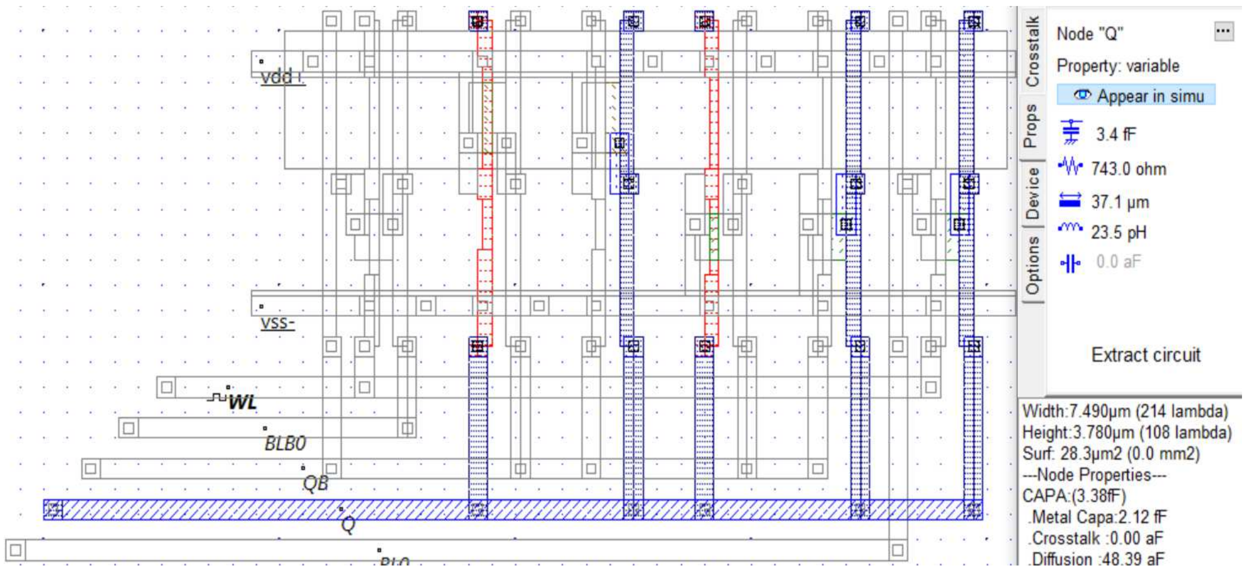
Existing testing methods to test the embedded memory defects during physical design are well established to find the faults described by the fault primitives. But they did not consider the parasitic effects and fault masking. Therefore we proposed Parasitic R, C extraction method, which gives the 100% fault detection. Recently, there has been increased lot of interest in the use of machine learning-based modeling tools.t. In this work, we explore Linear Regression machine learning techniques to estimate the parasitic R, C values for different technologies (120nm, 90nm, 45nm, 32nm and 7nm) and decision tree algorithm to detect the fault in SRAM cell. The major contributions of our work are as follows:

- We have analyzed the 6T-SRAM cell for all possible open/short (Section II).
- We conducted experiments to identify the various faults occur due to the short defects or open defects between the nodes (Section II).
- We have investigated the Machine learning techniques to estimate the parasitic R, C values and to detect and locate the defect in embedded memory(Section III)

This paper is organized as follows: in section III we discussed about proposed parasitic extraction method for all open and short faults. Section IV we have discussed different machine learning techniques used in VLSI in section V draws some conclusions.

3. Proposed Parasitic Extraction Method:

As technology continues to move in the direction of scaling down, dense eSRAMs may be produced by high error-prone designs. As a result, memory and SoC yield are decreased. As a result, a solution is needed, and it needs to be free of technological variations [13, 14]. Another drawback in the most recent testing methodology is that it does not account for the parasitic memory effect, leaving the test uncompleted. With this in mind, we suggested a testing approach for eSRAM that enables an extremely accurate fault identification via parasitic R, C extraction from a fault-induced architecture



Node "Q"
Property: variable
3.4 fF
743.0 ohm
37.1 µm
23.5 pH
0.0 aF

Width:7.490µm (214 lambda)
Height:3.780µm (108 lambda)
Surf: 28.3µm2 (0.0 mm2)
---Node Properties---
CAPA:(3.38fF)
.Metal Capa:2.12 fF
.Crosstalk :0.00 aF
.Diffusion :48.39 aF

.

Fig 1. Extraction of parasitic R, C values

Fig. 1, shows the layout diagram of 6T-SRAM cell. In the proposed method we extracted the parasitic R, C values at each node. The Parasitic capacitance is the sum of Metal, Diffusion, and Gate and cross talk capacitances. And the parasitic resistance is the sum of via resistance, poly resistance, diffusion resistance and Metal resistance.

3.1 Effect of open defects in 6T SRAM Cell

In this are article we have consider the node to node open/short faults. In Fig 2. We have imposed all possible open defects and then we have analyzed the memory cell for all possible open defects.

There are totally 25 open defects are possible as shown in the fig 2. The simulation results and different types faults occurs for all open defects are shown in table 1

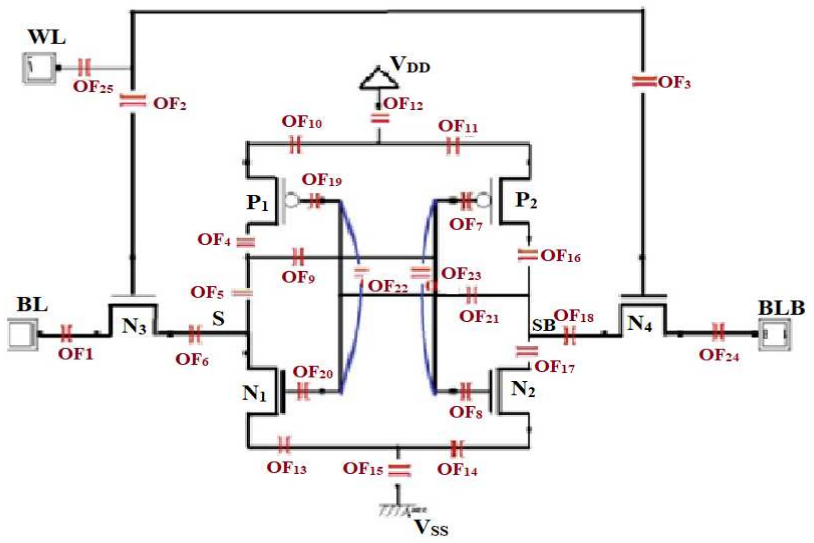


Fig2: Fault model for Open Defects in 6T-SRAM Cell

Table 1. 6T SRAM Cell open defect list for different technologies

Defect Representation	Open Defect at nodes	Technology	
		7nm	32nm
OF_1	$BL-N_3S$	NAF	NAF
OF_2	$WL-N_3G$	NAF	NAF
OF_3	$WL-N_4G$	URF	URF
OF_4	$Q-P_1D$	UWF1	UWF1
OF_5	$Q-N_1D$	UWF0	UWF0
OF_6	$Q-P_1DN_1D$	NAF	NAF
OF_7	$Q-P_2G$	UWF0, URF0	TF
OF_8	$Q-N_2G$	UWF1, URF1	TF
OF_9	$Q-P_2GN_2G$	NAF	NAF
OF_{10}	$VDD-P_1S$	UWF1	UWF1
OF_{11}	$VDD-P_2S$	UWF0, URF0	TF
OF_{12}	$VDD-P_1SP_2S$	UWF,URF0	UWF,URF0
OF_{13}	$VSS-N_1S$	UWF0	UWF0
OF_{14}	$VSS-N_2S$	UWF1, URF1	TF
OF_{15}	$VSS-N_1SN_2S$	UWF, URF1	UWF, URF1
OF_{16}	$QB-P_2D$	UWF0, URF0	TF
OF_{17}	$QB-N_2D$	UWF1,URF1	UWF1,URF1
OF_{18}	$QB-P_2DN_2D$	URF, UWF0	URF0, UWF
OF_{19}	$QB-P_1G$	UWF1	UWF1
OF_{20}	$QB-N_1G$	UWF0	UWF0
OF_{21}	$QB-P_1GN_1G$	UWF	UWF
OF_{22}	P_1G-N_1G	UWF	UWF

Defect Representation	Open Defect at nodes	Technology	
		7nm	32nm
OF_{23}	P_2G-N_2G	NAF	NAF
OF_{24}	BLB-N_4S	URF	URF
OF_{25}	WL-N_3GN_4G	NAF	NAF

Table 2.Variation in Parasitic R, C values to detect Open Faults

Node	Fault Free		NAF (BL-N_3S)		URF (WL- N_4G)		TF (Q-P_2G)		UWF (P_1G_N_1G)	
	C(aF)	R(Ω)	C(aF)	R(Ω)	C(aF)	R(Ω)	C(aF)	R(Ω)	C(aF)	R(Ω)
Q	1700	800	1800	805	1800	813	1600	527	1700	803
QB	1500	498	1500	498	1500	498	1500	498	1300	239
WL	770	296	780	296	520	155	780	296	780	296
BL	626	71	NA	NA	630	71	630	71	630	71
BLB	815	91	820	91	820	91	820	91	820	91
VDD	310	13	310	13	310	13	310	13	310	13
VSS	310	13	310	13	310	13	310	13	310	13

Table 2 shows the extracted parasitic R, and C values of different faults occurs for the open defects between the nodes.

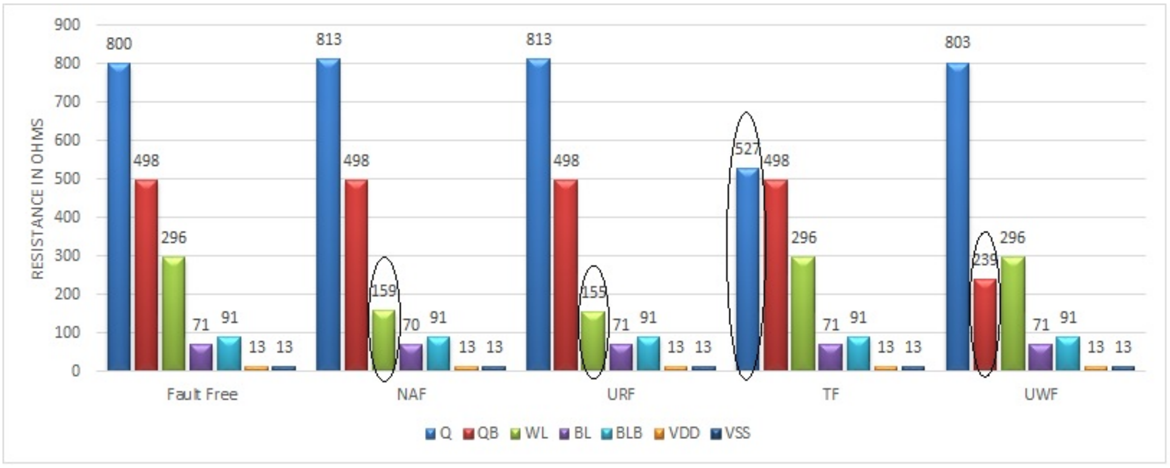


Fig 3. Using variation of parasitic R Value Detection of open faults

Fig 3. Shows the extracted parasitic R values for fault free SRAM cell. When we impose the open defect, we have observed the different the faults like No Access fault, Undefined Read Fault, Transition Fault and Undefined Write Faults. As shown in the figure for No Access Fault the resistance value at node WL changes from 296Ω to 159Ω, thus we can conclude that open defect at WL will cause for the No Access Fault. Similarly for transition faults at node BLB, the resistance

value changes from 800 ohm to 527 ohms, for Undefined Write Fault at node QB the resistance changes from 498 ohms to 239 ohms. Therefore changes in the resistance at particular node indicates the defect at the node. Same explanation true for the parasitic capacitance. It means the change in the capacitance value at a node indicates the fault at that node.

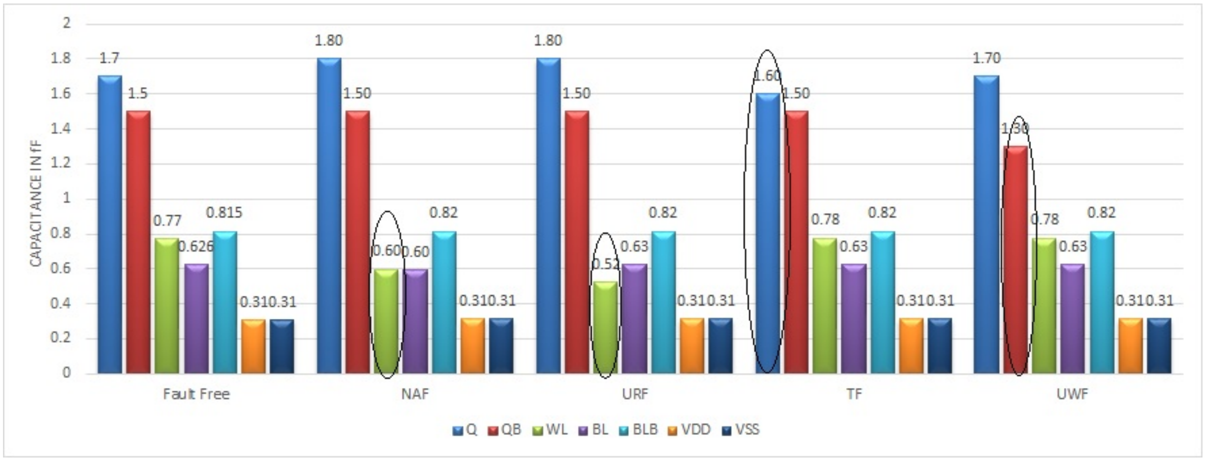


Fig 4. Using variation of parasitic C Value Detection of open faults

Table 3. Extracted Parasitic R, C values for all open defects

S. No	Open Defect	at QB		at Q		At WL		at BL		at BLB		at VDD		at VSS	
		C(aF)	R(Ω)	C(aF)	R(Ω)	C(aF)	R(Ω)	C(aF)	R(Ω)	C(aF)	R(Ω)	C(aF)	R(Ω)	C(aF)	R(Ω)
	Fault Free	4470	6728	4660	7185	1990	370	930	1160	1030	2100	8250	7060	2040	3610
1	BL-N_4S	4470	6730	4670	7190	1990	370	NA	NA	1030	2100	8250	7060	2040	3610
2	WL- N_4G	4470	6730	4670	7190	1230	190	910	1160	1030	2100	8250	7060	2040	3610
3	WL- N_5G	4470	6730	4660	7190	1150	190	930	1160	1030	2100	8250	7060	2040	3610
4	Q-P_1D	4480	6730	3700	4730	1990	370	930	1160	1030	2100	8250	7060	2040	3610
5	Q-N_1D	4470	6730	4140	5090	1990	370	930	1160	1030	2100	8250	7060	2040	3610
6	Q- P_1D N_1D	4470	6730	930	2090	1990	370	930	1160	1030	2100	8250	7060	2040	3610
7	Q- P_2G	4470	6730	3140	6830	1990	370	930	1160	1030	2100	8250	7060	2040	3610
8	Q- N_2G	4470	6730	3900	7000	1990	370	930	1160	1030	2100	8250	7060	2040	3610
9	Q- P_2G N_2G	4470	6730	2380	6650	1990	370	930	1160	1030	2100	8250	7060	2040	3610
10	VDD- P_1S	4470	6730	4660	7190	1990	370	930	1160	1030	2100	7670	4620	2040	3610
11	VDD- P_2S	4470	6730	4660	7190	1990	370	930	1160	1030	2100	7640	4400	2020	3610
12	VDD- P_1S P_2S	4470	6730	4660	7190	1990	370	930	1160	1030	2100	7060	1960	2010	3610
13	VSS- N_1S	4470	6730	4660	7190	1990	370	930	1160	1030	2100	8250	7060	1720	2460

14	VSS- N_2S	4470	6730	4660	7190	1990	370	930	1160	1030	2100	8250	7060	1720	2460
15	VSS N_1S N_2S	4470	6730	4660	7190	1990	370	930	1160	1030	2100	8250	7060	1390	1320
16	QB - P_2D	3510	4050	4660	7190	1990	370	930	1160	1030	2100	8250	7060	2040	3610
17	QB - N_2D	3970	4640	4660	7190	1990	370	930	1160	1030	2100	8250	7060	2040	3610
18	QB_ P_2D N_2D	1000	1160	4660	7190	1990	370	930	1160	1030	2100	8250	7060	2040	3610
19	QB_ P_1G	3200	6170	4660	7190	1990	370	930	1160	NA	NA	8250	7060	2040	3610
20	QB_ N_1G	3730	6490	4660	7190	1990	370	930	1160	1030	2100	8250	7060	2040	3610
21	QB_ P_1G N_1G	2240	5930	4660	7190	1990	370	930	1160	1030	2100	8250	7060	2040	3610
22	P_1G_ N_1G	2440	5930	4660	7190	1990	370	930	1160	1030	2100	8250	7060	2040	3610
23	P_2G_ N_2G	4470	6730	2380	6650	1990	370	930	1160	1030	2100	8250	7060	2040	3610
24	BLB - N_5S	4470	6730	4660	7190	1990	370	930	1160	NA	NA	8250	7060	2040	3610
25	WL- N_4G N_5G	4470	6730	4660	7190	NA	NA	930	1160	1030	2100	8250	7060	2040	3610

3.2 Effect of Short defects in 6T SRAM Cell

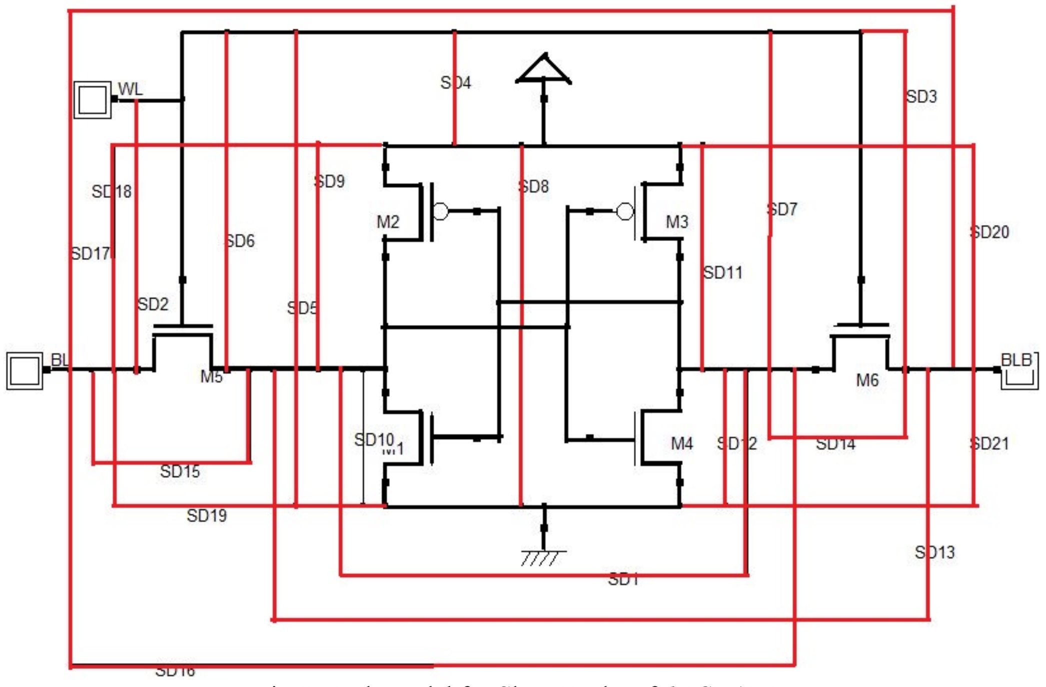


Fig 5. Fault model for Short Faults of 6T SRAM

Fig 5. Depicted the all possible short defects for the single 6T-SRAM cell. As shown in the fig 5. There are totally 21 short defects are possible between the nodes excluding the equivalent nodes. These defects will have the effect on the functional behavior of the cell. Table 4. Shows the different faults occur for the short faults.

Table 4. 6T SRAM Cell short defect list for different technologies

S.No	Fault Representation	Short between Nodes	Technology		
			45nm	32nm	7nm
1	SF_1	$S\text{-}S_B$	UWF, URF	USWF, URF	USWF, URF
2	SF_2	WL-BL	SA1	TF	WBAF, TF
3	SF_3	WL-BLB	USF	USRF-1	WBAF, USRF-1
4	SF_4	$WL\text{-}V_{DD}$	Error(NAF)	Error(NAF)	Error(NAF)
5	SF_5	$WL\text{-}V_{SS}$	Error(NAF)	Error(NAF)	Error(NAF)
6	SF_6	WL-S	SA-0, URF	SA-0, URF	SA-0, URF
7	SF_7	$WL\text{-}S_B$	SA-1,URF	SA-1, URF	SA-1, URF
8	SF_8	$V_{DD}\text{-}V_{SS}$	UWF, URF-0	UWF, URF-0	UWF, URF-0
9	SF_9	$S\text{-}V_{DD}$	URF, UWF	URF-0, UWF-0	URF-0, UWF-0
10	SF_{10}	$S\text{-}V_{SS}$	URF, UWF	URF-1, UWF-1	URF-1, UWF-1
11	SF_{11}	$S_B\text{-}V_{DD}$	IOF	IOF	IOF
12	SF_{12}	$S_B\text{-}V_{SS}$	UWF, URF-0	TF, URF-0	TF, URF-0
13	SF_{13}	S-BLB	URF	URF	URF
14	SF_{14}	$S_B\text{-}BLB$	WBAF	WBAF, USWF0, USRF0	USWF-0, USRF-0
15	SF_{15}	S-BL	SA-0	WBAF, SA-0	SA-0
16	SF_{16}	$S_B\text{-}BL$	USWF, USRF	WBAF, USWF, USRF	USWF, USRF
17	SF_{17}	BL-BLB	USWF, USRF	USWF, USRF	USWF, USRF
18	SF_{18}	$BL\text{-}V_{DD}$	Error	Error	Error
19	SF_{19}	$BL\text{-}V_{SS}$	Error	Error	Error
20	SF_{20}	$BLB\text{-}V_{DD}$	Error	Error	Error
21	SF_{21}	$BLB\text{-}V_{SS}$	Error	Error	Error

Table 5. Extracted Parasitic R, C values for all short defects

S.No	Short Defect	at Q		at Q_B		at WL		at BL		at BLB		at V_{DD}		at V_{SS}	
		C in fF	R in Ω	C in fF	R in Ω	C in fF	R in Ω	C in fF	R in Ω	C in fF	R in Ω	C in fF	R in Ω	C in fF	R in Ω
1	**Fault Free**	**2.9**	**433**	**3.1**	**1170**	**1.8**	**180**	**1.1**	**158**	**0.78**	**54**	**2.7**	**2071**	**1.7**	**402**
2	$Q\text{-}Q_B$	5.50	1583	NO	NO	1.80	178	1.00	157	0.75	53	2.70	2071	1.70	402
3	WL-BL	2.90	433	3.10	1170	NO	NO	1.60	236	0.78	54	2.70	2071	1.70	402
4	WL-BLB	2.90	433	3.10	1170	2.10	219	1.00	159	NO	NO	2.70	2071	1.70	402
5	$WL\text{-}V_{DD}$	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	2.80	2164	1.70	402

6	WL-V_{SS}	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	2.70	2071	2.70	553
7	Q-WL	4.00	565	3.10	803	NO	NO	1.00	157	0.75	54	2.70	2071	1.70	402
8	Q_B-WL	2.90	433	NO	NO	4.30	1331	1.00	157	0.75	54	2.70	2071	1.70	402
9	V_{DD}-V_{SS}	2.90	433	3.10	1170	1.80	180	1.10	158	0.78	54	2.40	1670	2.00	805
10	Q-V_{DD}	NO	NO	3.00	971	1.80	178	1.000	158	0.78	54	3.60	2409	1.70	402
11	Q-V_{SS}	NO	NO	3.00	971	1.80	178	1.00	158	0.75	53	2.70	2071	3.10	743
12	Q_B-V_{DD}	2.90	407	NO	NO	1.80	178	1.00	157	0.75	54	4.00	2787	1.70	402
13	Q_B-V_{SS}	2.90	407	NO	NO	1.80	178	1.00	157	0.75	53	2.70	2071	3.50	1146
14	Q-BLB	3.10	445	3.10	803	1.80	180	1.00	157	NO	NO	2.70	2071	1.70	402
15	Q_B-BLB	2.90	407	3.40	842	1.80	180	1.00	157	NO	NO	2.70	2071	1.70	402
16	Q-BL	NO	NO	3.10	1170	1.80	180	2.90	529	0.78	54	2.70	2071	1.70	402
17	Q_B-BL	2.90	407	NO	NO	1.80	180	3.50	941	0.78	54	2.70	2071	1.70	402
18	BL-BLB	2.90	407	3.10	803	1.80	180	1.30	196	NO	NO	2.70	2071	1.70	402
19	BL-V_{DD}	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	2.80	2198	1.70	402
20	BL-V_{SS}	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	2.70	2071	1.80	528
21	BLB-V_{DD}	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	2.80	2101	1.70	402
22	BLB-V_{SS}	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	2.70	2071	1.70	430

Table 5 shows the extracted parasitic R, and C values of different faults. These faults occurs for the short defects between the nodes. In the above table NO is the abbreviation for the Node Absorbed. When we short two nodes one node will become the equivalent to another node. In this case one node will be absorbed

Table 6. Variation of parasitic R, C values for SRAM short defect model

	Fault Free		WL-BL (WBAF, TF)		V_{DD}-V_{SS} (UWF, URF0)		Q_B-V_{DD} (IoF)		Q-BL (SA0)		Q_B-BL (USWF, USRF)	
			Effected Node WL		Effected Node V_{DD} & V_{SS}		Effected Node V_{DD}		Effected Node BL		Effected Node BL	
	C(aF)	R(Ω)	C(aF)	R(Ω)	C(aF)	R(Ω)	C(aF)	R(Ω)	C(aF)	R(Ω)	C(aF)	R(Ω)
Q	2900	433	2900	433	2900	433	2900	407	NO	NO	2900	407
QB	3100	1170	3100	1170	3100	1170	NO	NO	3100	1170	NO	NO
WL	1800	180	NO	NO	1800	180	1800	178	1800	180	1800	180
BL	1800	158	1600	236	1100	158	1000	157	2900	529	3500	941
BLB	783	54	783	54	783	54	753	54	783	54	783	54

VDD	2700	2071	2700	2071	2400	1670	4000	2787	2700	2071	2700	2071
VSS	1700	402	1700	402	2000	805	1700	402	1700	402	1700	402

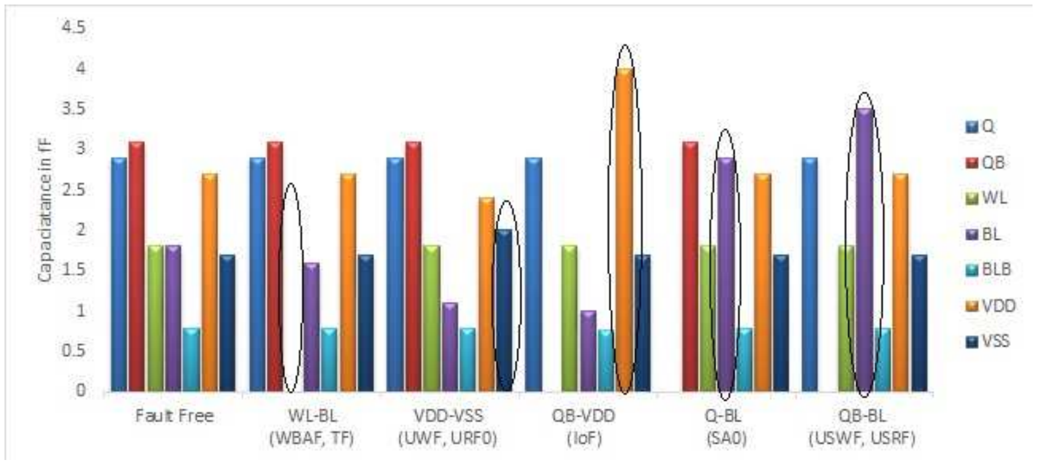


Fig.6 Fault detection based on parasitic capacitance variation for short defects

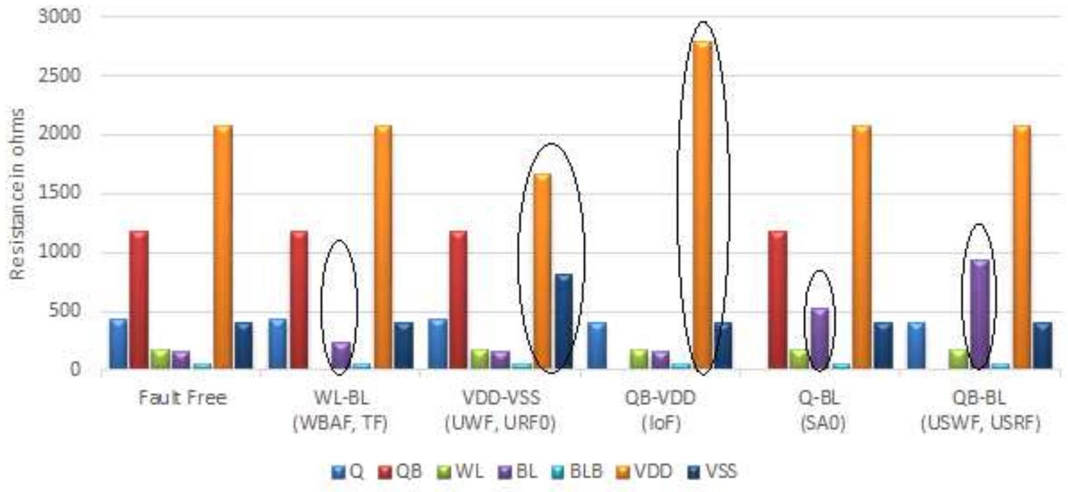


Fig.7 Fault detection based on parasitic resistance variation for short faults

Fig 7. Shows the extracted parasitic R values for fault free SRAM cell. When we impose the short defect, we have observed the different the faults like Write Before Access Fault, Initialization order Fault, Stuck at Fault. As shown in the figure for IoF the resistance value at node VDD changes from 2071Ω to 2787Ω, thus we can conclude that short defect at VDD will cause for the, Initialization order Fault. Similarly for transition faults at node BLB, the resistance value changes from 800 ohm to 527 ohms, for Undefined Write Fault at node QB the resistance changes from 498 ohms to 239 ohms. Therefore changes in the resistance at particular node indicates the defect at the node. Same explanation true for the parasitic capacitance. It means the change in the capacitance value at a node indicates the fault at that node.

3. Results and Comparison

3.1 Machine Learning Techniques in embedded Memory

Recent developments in the application of machine learning approaches to design research challenges have generated a lot of interest [10, 11]. A model is trained or guided by the actual application of a process or phenomena, and then it is used to predict the same metric for new input data. The training set refers to the data used to develop the model initially. It should be evaluated using an entirely new set, known as the testing set, in order to determine the goodness of the developed model. If the actual set of inputs chosen is highly linked with the expected output, it is crucial to consider the fitness value of the training set.

We must have a solid and broad training set from real data obtained through operations in order to have a good model for variation estimates. To achieve this, we have obtain huge data set for the short and open faults. For short faults totally we got 21 defects at 7 nodes. Each node will have different resistance and capacitance values for different faults. Similarly we have calculated the parasitic R, C value for 25 different open faults at 7 nodes. table 3 and table 5 shows the obtained values. This will provide a large dataset for training and testing. One of the key features of our work is the use of actual layouts to extract parasitic R, C values, then we impose the short/open defects then calculated the Parasitic R, C values, these values are used find the defects of the SRAM cell.

3.2 Machine Learning Design Methodology

Machine learning is a branch of artificial intelligence that allows systems to learn from large amounts of data and address certain issues. It makes use of computer algorithms whose effectiveness is automatically improved through practice.

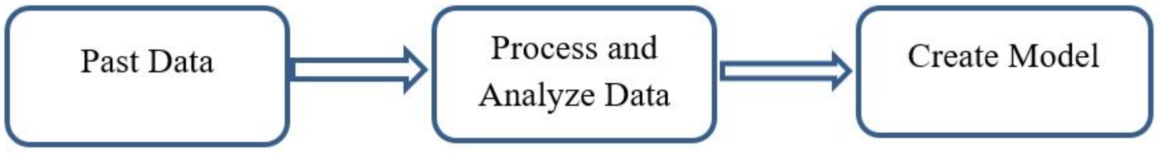


Fig 8. Machine learning model

There are primarily three types of machine learning: Supervised, Unsupervised, and Reinforcement Learning.

Supervised Learning: In supervised learning, machine learning models are trained using labeled data. The outcome in labeled data is already known. The model only needs to map the inputs to the corresponding outputs. Algorithms for supervised learning are frequently employed to solve classification and regression issues.

Linear Regression, Logistic Regression, SVM algorithm, KNN algorithm, Decision Tree, Random Forest are supervised learning algorithms

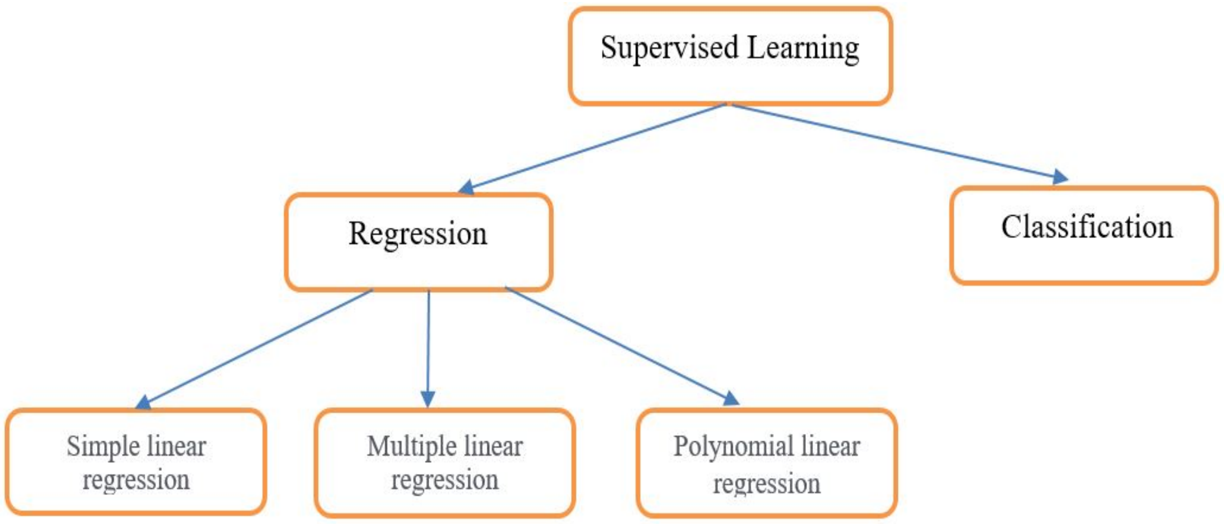


Fig 9. Types of Supervised Learning

Unsupervised Learning: Machines are trained with unlabeled data using a technique called unsupervised learning. No fixed output variable exists for unlabeled data. The model takes in the information from the data, looks for patterns and features, and then outputs the results. For the purpose of resolving clustering and association issues, unsupervised learning is employed.

Reinforcement Learning: Reinforcement Learning enables a machine to respond appropriately and maximize its benefits in a certain circumstance. To generate actions and rewards, it makes use of an agent and an environment. The agent has a beginning state and a conclusion state. However, there could be numerous routes leading to the goal, much like a maze. There is no fixed target variable in this learning method.

In our proposed method we have used multiple linear regression to predict the parasitic R, C values. The regression method explained in the following section.

Simple Linear Regression:

The relationship between independent and dependent variables can be predicted using a statistical model called linear regression by looking at two aspects:

1. Specifically, which variables are capable of accurately predicting the outcome variable?
2. In terms of creating predictions with the highest degree of accuracy, how significant is the regression line?

An independent variable's value is unaffected by the effects of other variables. It is frequently indicated with a "x."

The dependent variable is affected by an independent variable. When the values of the independent variables change, the dependent variable's value also changes. It is frequently indicated by a "y".

The linear regression represented by the equation of

$y = m*x + c$

Where x → independent variable, y → dependent variable, m → slope

Multiple Linear Regression

The multiple linear regression, represented by the equation of $y = m_1x_1 + m_2x_2 + m_3x_3 + ... + c$

Where x_1, x_2 and $x_{3...}$ are the independent variables. m_1, m_2, m_3 indicates the slopes.

3.2.1 Determination of Parasitic R, C values by using Multiple Linear Regression:

Table 7. Shows the extracted R, C values for the different technologies from the layout diagram of the 6T-SRAM Cell at each node as shown in the fig 1. In the table shown we have used multiple linear regression to determine the R, C values. In this process we have used technology and length as the independent variables and Resistance and Capacitance are the dependent variables.

Table 7. Extracted R and C values for different technologies

Node	Technology(nm)	L(um)	R(ohms)	C(fF)
Q	120	69.2	1336	7.1
QB	120	78.3	1415	7.5
WL	120	31.3	371	4

BL	120	20.7	146	0.973
BLB	120	28.3	243	1.2
VDD	120	12.6	2	0.604
VSS	120	12.6	2	0.604
Q	90	64.7	1013	6.8
QB	90	57.8	949	6.5
WL	90	21.1	337	2.9
BL	90	18.1	99	1.2
BLB	90	10.2	188	0.753
VDD	90	9.4	6	0.537
VSS	90	9.4	6	0.537
Q	45	29.9	1518	3.6
QB	45	25.8	1128	3.3
WL	45	9.6	415	1.4
BL	45	8.1	152	0.816
BLB	45	4.6	247	0.484
VDD	45	4.2	8	0.404
VSS	45	4.2	8	0.404
Q	32	19.4	818	2
QB	32	17	682	1.8
WL	32	7.3	335	0.692
BL	32	6.3	75	0.701
BLB	32	3.9	67	0.459
VDD	32	2.9	13	0.314
VSS	32	2.9	13	0.314
Q	7	3.3	7417	0.642
QB	7	3.5	7077	0.681
WL	7	1.7	3553	0.34
BL	7	1.3	951	0.195
BLB	7	1.7	1371	0.256
VDD	7	0.65	23	0.081
VSS	7	0.65	23	0.081

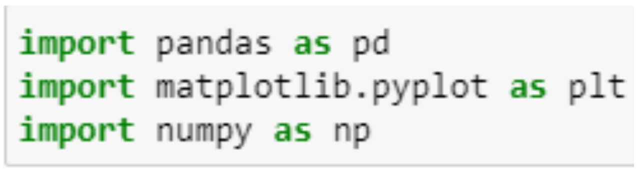
The main steps involved in the multiple linear regression to determine the R and C values are as follows:

1. Importing the libraries
2. Load the data set and extract independent and dependent variable
3. Data Visualization
4. Encoding the Data
5. Splitting the data into train and test set

6. Fitting the Multiple Linear Regression to training set
7. Predicting the test results

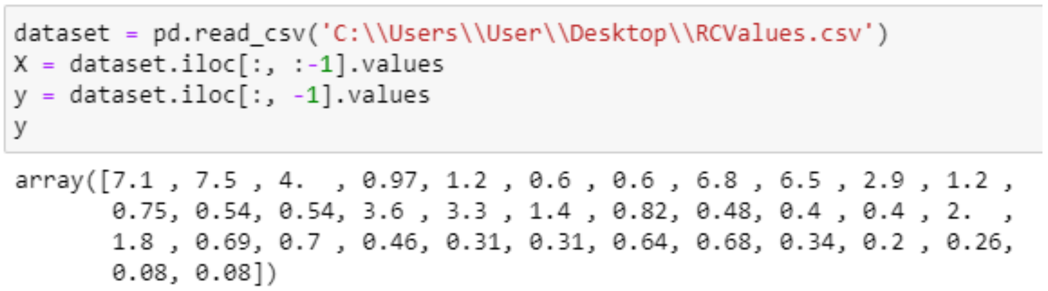
Simulation Results:

1. Importing the libraries: we have imported the Pandas and Numpy libraries for the data processing and perform the numerical operations respectively.



```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

2. Load the data set and extract independent and dependent variable

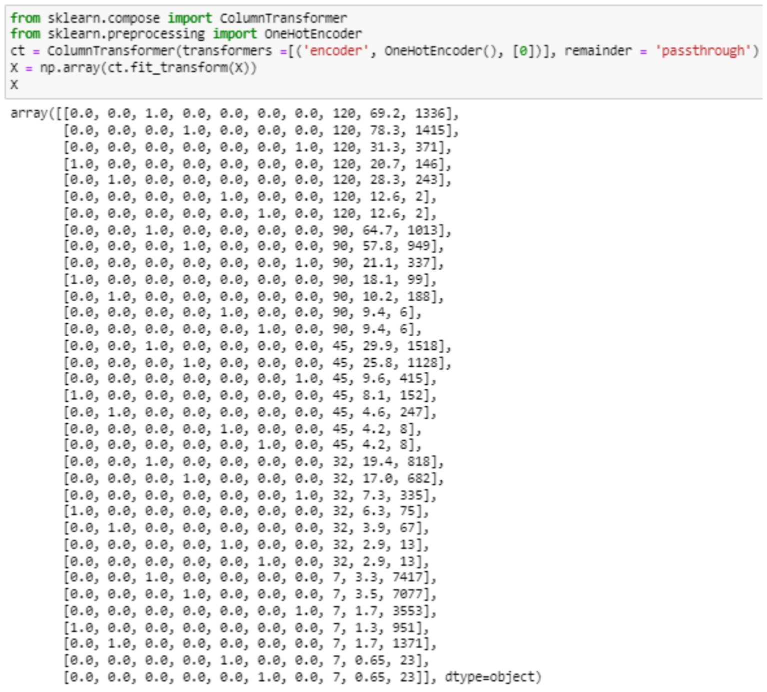


```
dataset = pd.read_csv('C:\\Users\\User\\Desktop\\RCValues.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
y
```

```
array([7.1 , 7.5 , 4.  , 0.97, 1.2 , 0.6 , 0.6 , 6.8 , 6.5 , 2.9 , 1.2 ,
       0.75, 0.54, 0.54, 3.6 , 3.3 , 1.4 , 0.82, 0.48, 0.4 , 0.4 , 2.  ,
       1.8 , 0.69, 0.7 , 0.46, 0.31, 0.31, 0.64, 0.68, 0.34, 0.2 , 0.26,
       0.08, 0.08])
```

pd.read_csv is used to load the data. Dataset.iloc is used to select the particular row and column to determine the dependent and independent variables. In the given dataset we have made resistance column as the dependent variable and technology and length as independent variables.

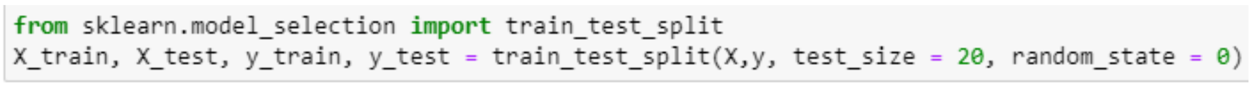
3. Data Visualization and Encoding the data



```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers =[('encoder', OneHotEncoder(), [0])], remainder = 'passthrough')
X = np.array(ct.fit_transform(X))
X
```

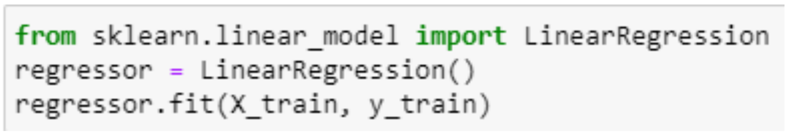
```
array([[0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 120, 69.2, 1336],
       [0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 120, 78.3, 1415],
       [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 120, 31.3, 371],
       [1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 120, 20.7, 146],
       [0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 120, 28.3, 243],
       [0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 120, 12.6, 2],
       [0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 120, 12.6, 2],
       [0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 90, 64.7, 1013],
       [0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 90, 57.8, 949],
       [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 90, 21.1, 337],
       [1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 90, 18.1, 99],
       [0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 90, 10.2, 188],
       [0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 90, 9.4, 6],
       [0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 90, 9.4, 6],
       [0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 45, 29.9, 1518],
       [0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 45, 25.8, 1128],
       [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 45, 9.6, 415],
       [1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 45, 8.1, 152],
       [0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 45, 4.6, 247],
       [0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 45, 4.2, 8],
       [0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 45, 4.2, 8],
       [0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 32, 19.4, 818],
       [0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 32, 17.0, 682],
       [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 32, 7.3, 335],
       [1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 32, 6.3, 75],
       [0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 32, 3.9, 67],
       [0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 32, 2.9, 13],
       [0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 32, 2.9, 13],
       [0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 7, 3.3, 7417],
       [0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 7, 3.5, 7077],
       [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 7, 1.7, 3553],
       [1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 7, 1.3, 951],
       [0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 7, 1.7, 1371],
       [0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 7, 0.65, 23],
       [0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 7, 0.65, 23]], dtype=object)
```

4. Splitting the data into train and test set



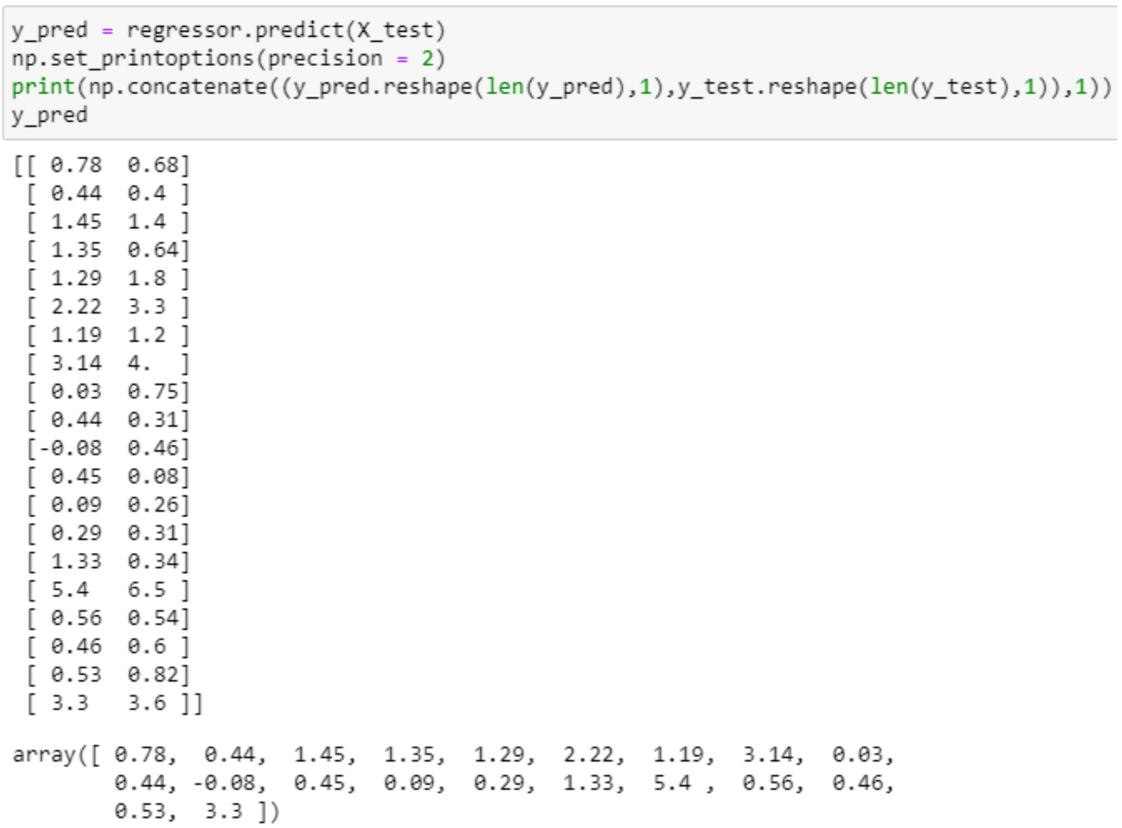
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 20, random_state = 0)
```

5. Fitting the Multiple Linear Regression to training set



```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

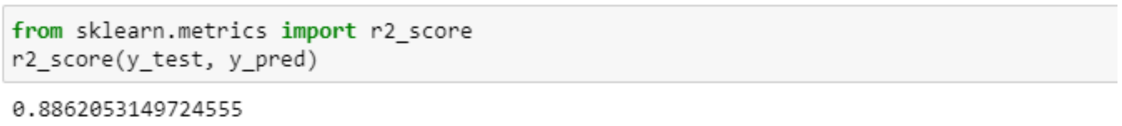
6. Predicting the test results



```
y_pred = regressor.predict(X_test)
np.set_printoptions(precision = 2)
print(np.concatenate((y_pred.reshape(len(y_pred),1),y_test.reshape(len(y_test),1)),1))
y_pred
```

```
[[ 0.78  0.68]
 [ 0.44  0.4 ]
 [ 1.45  1.4 ]
 [ 1.35  0.64]
 [ 1.29  1.8 ]
 [ 2.22  3.3 ]
 [ 1.19  1.2 ]
 [ 3.14  4.  ]
 [ 0.03  0.75]
 [ 0.44  0.31]
 [-0.08  0.46]
 [ 0.45  0.08]
 [ 0.09  0.26]
 [ 0.29  0.31]
 [ 1.33  0.34]
 [ 5.4   6.5 ]
 [ 0.56  0.54]
 [ 0.46  0.6 ]
 [ 0.53  0.82]
 [ 3.3   3.6 ]]

array([ 0.78,  0.44,  1.45,  1.35,  1.29,  2.22,  1.19,  3.14,  0.03,
        0.44, -0.08,  0.45,  0.09,  0.29,  1.33,  5.4 ,  0.56,  0.46,
        0.53,  3.3 ])
```



```
from sklearn.metrics import r2_score
r2_score(y_test, y_pred)
```

```
0.8862053149724555
```

Thus our proposed model gives the 88.62% accuracy to determine the parasitic C values for fault free SRAM Cell. After extracted the Parasitic R, C values, we will use these values to find the defects and location of the faulty SRAM cell. Table 1 shows fault model dictionary for all open faults. Table 4 shows fault model dictionary for all short faults.

After estimation of the parasitic R, C Values, We have used Decision Tree algorithm to find the fault and its location, A decision tree is a tree-based supervised learning technique used to forecast a target variable's result. With the support of regression and classification algorithms, supervised learning employs labeled data information with known output variables to create predictions. Using different data features, it learns from basic decision-making guidelines. Python decision trees are widely used to calculate probabilities because they may be utilized to handle classification and regression issues.

Important Terms Used in Decision Trees:

1. **Entropy:** The amount of uncertainty or randomness in a set of data is measured by entropy. How a decision tree divides the data depends on entropy. The following formula is used to calculate the uncertainty.

$$\sum_{i=1}^{k} P(value_i).\log_2(P(value_i)$$

2. **Information Gain:** After the data set is divided, the information gain calculates the reduction in entropy.
   IG(Y, X) = Entropy (Y) - Entropy (Y | X) formula is used to calculate the information gain
3. **Gini Index:** To select the appropriate variable for splitting nodes, the Gini Index is used. It assesses the frequency of inaccurate identification of a randomly selected variable.
4. **Root Node:** The top node of a decision tree is always the root node. It can be further split into various sets and represents the total population or data sample.
5. **Decision Node:** Decision nodes are subnodes that can be divided into other subnodes and include two or more branches.
6. **Leaf Node:** A leaf carries the final results. These nodes, are also known as terminal nodes, and these nodes further cannot be split any further

We will now predict if the memory cell is faulty cell or fault free cell using the decision tree algorithm in machine learning. In order to make the prediction, the data set includes a variety of information, such as the capacitance and resistance values for defective and fault-free SRAM cells at each node, as well as fault information.. Table shows the extracted capacitance and resistance values for the open fault detection at the nodes QB, Q, WL, BL, BLB and VDD.

Table 8. Extracted R and C values for faulty and fault free SRAM at different nodes

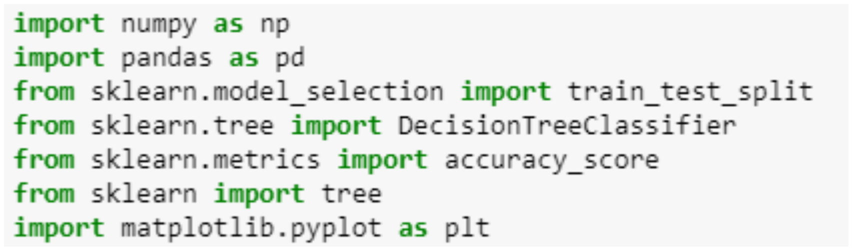
C in fF	R in KΩ	Faulty or Not	C in fF	R in KΩ	Faulty or Not	C in fF	R in KΩ	Faulty or Not
2.9	18.96	Fault Free	3.3	20.18	Fault Free	0.88	2.66	Fault Free
2.9	18.96	Fault Free	3.3	20.18	Fault Free	0.88	2.66	Fault Free
2.9	18.96	Fault Free	3.3	20.18	Fault Free	0.26	1.36	Fault at WL
2.9	18.96	Fault Free	3.3	20.18	Fault Free	0.8	1.36	Fault at WL
2.9	18.96	Fault Free	2.4	12.26	Fault at Q	0.88	2.66	Fault Free
2.9	18.96	Fault Free	2.7	16.74	Fault at Q	0.88	2.66	Fault Free
2.9	18.96	Fault Free	1.2	3.62	Fault at Q	0.88	2.66	Fault Free
2.9	18.96	Fault Free	2.8	16.13	Fault at Q	0.88	2.66	Fault Free

2.9	18.96	Fault Free	3	18.94	Fault at Q	0.88	2.66	Fault Free
2.9	18.96	Fault Free	2.6	14.91	Fault at Q	0.88	2.66	Fault Free
2.9	18.96	Fault Free	3.3	20.18	Fault Free	0.88	2.66	Fault Free
2.9	18.96	Fault Free	3.3	20.18	Fault Free	0.88	2.66	Fault Free
2.9	18.96	Fault Free	3.3	20.18	Fault Free	0.88	2.66	Fault Free
2.9	18.96	Fault Free	3.3	20.18	Fault Free	0.88	2.66	Fault Free
2.9	18.96	Fault Free	3.3	20.18	Fault Free	0.88	2.66	Fault Free
2.9	18.96	Fault Free	3.3	20.18	Fault Free	0.88	2.66	Fault Free
2.1	11.06	Fault at QB	3.3	20.18	Fault Free	0.88	2.66	Fault Free
2.4	15.51	Fault at QB	3.3	20.18	Fault Free	0.88	2.66	Fault Free
0.52	2.31	Fault at QB	3.3	20.18	Fault Free	0.88	2.66	Fault Free
2.5	14.94	Fault at QB	3.3	20.18	Fault Free	0.88	2.66	Fault Free
2.7	17.74	Fault at QB	3.3	20.18	Fault Free	0.88	2.66	Fault Free
2.7	17.74	Fault at QB	3.3	20.18	Fault Free	0.88	2.66	Fault Free
2.3	13.72	Fault at QB	3.3	20.18	Fault Free	0.88	2.66	Fault Free
2.9	18.96	Fault Free	2.6	14.91	Fault at Q	0.88	2.66	Fault Free
2.9	18.96	Fault Free	3.3	20.18	Fault Free	0.88	2.66	Fault Free
2.9	18.96	Fault Free	3.3	20.18	Fault Free	0	0	Fault at WL
1	2.6	Fault Free	0.61	3.29	Fault Free	2.4	12.03	Fault Free
0	0	Fault at BL	0.61	3.29	Fault Free	2.4	12.03	Fault Free
1	2.6	Fault Free	0.61	3.29	Fault Free	2.4	12.03	Fault Free
1	2.6	Fault Free	0.61	3.29	Fault Free	2.4	12.03	Fault Free
1	2.6	Fault Free	0.61	3.29	Fault Free	2.4	12.03	Fault Free
1	2.6	Fault Free	0.61	3.29	Fault Free	2.4	12.03	Fault Free
1	2.6	Fault Free	0.61	3.29	Fault Free	2.4	12.03	Fault Free
1	2.6	Fault Free	0.61	3.29	Fault Free	2.4	12.03	Fault Free
1	2.6	Fault Free	0.61	3.29	Fault Free	2.4	12.03	Fault Free
1	2.6	Fault Free	0.61	3.29	Fault Free	2.4	12.03	Fault Free
1	2.6	Fault Free	0.61	3.29	Fault Free	2.1	8.65	Fault at VDD
1	2.6	Fault Free	0.61	3.29	Fault Free	2.1	8.65	Fault at VDD
1	2.6	Fault Free	0.61	3.29	Fault Free	1.9	5.27	Fault at VDD
1	2.6	Fault Free	0.61	3.29	Fault Free	2.4	12.03	Fault Free
1	2.6	Fault Free	0.61	3.29	Fault Free	2.4	12.03	Fault Free
1	2.6	Fault Free	0.61	3.29	Fault Free	2.4	12.03	Fault Free
1	2.6	Fault Free	0.61	3.29	Fault Free	2.4	12.03	Fault Free
1	2.6	Fault Free	0.61	3.29	Fault Free	2.4	12.03	Fault Free
1	2.6	Fault Free	0.61	3.29	Fault Free	2.4	12.03	Fault Free
1	2.6	Fault Free	0.61	3.29	Fault Free	2.4	12.03	Fault Free
1	2.6	Fault Free	0.61	3.29	Fault Free	2.4	12.03	Fault Free
1	2.6	Fault Free	0.61	3.29	Fault Free	2.4	12.03	Fault Free
1	2.6	Fault Free	0.61	3.29	Fault Free	2.4	12.03	Fault Free

1	2.6	Fault Free	0.61	3.29	Fault Free	2.4	12.03	Fault Free
1	2.6	Fault Free	0	0	Fault at BLB	2.4	12.03	Fault Free
1	2.6	Fault Free	0.61	3.3	Fault Free	2.4	12.03	Fault Free

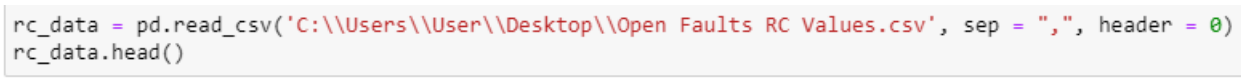
Building a Decision Tree for fault detection in SRAM Cell

1. Import the libraries for Decision Tree.

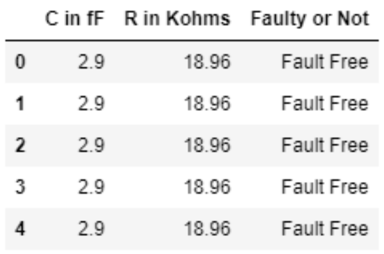


```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn import tree
import matplotlib.pyplot as plt
```

2. Load the data using Pandas

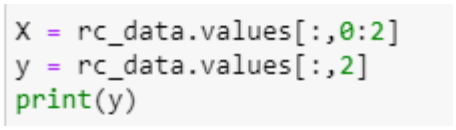


```
rc_data = pd.read_csv('C:\\Users\\User\\Desktop\\Open Faults RC Values.csv', sep = ",", header = 0)
rc_data.head()
```



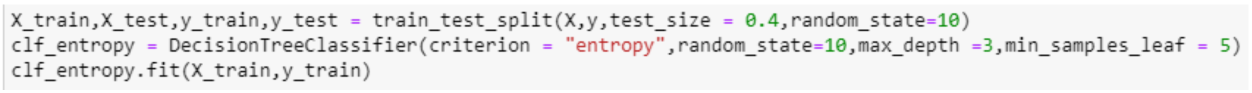
	C in fF	R in Kohms	Faulty or Not
0	2.9	18.96	Fault Free
1	2.9	18.96	Fault Free
2	2.9	18.96	Fault Free
3	2.9	18.96	Fault Free
4	2.9	18.96	Fault Free

3. Slicing method separate dependent and independent variables.



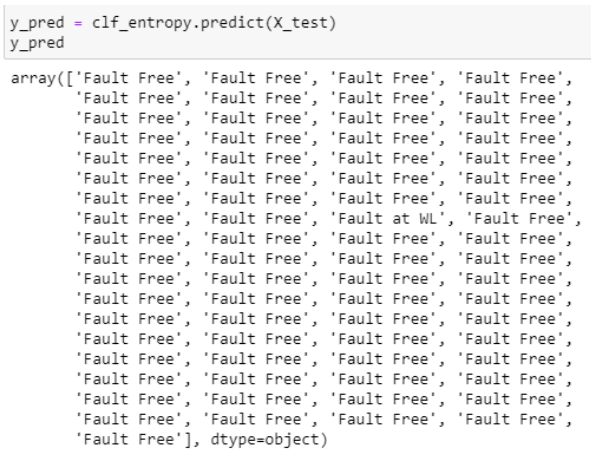
```
X = rc_data.values[:,0:2]
y = rc_data.values[:,2]
print(y)
```

4. Using the decision tree classifier split the train and test data



```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.4,random_state=10)
clf_entropy = DecisionTreeClassifier(criterion = "entropy",random_state=10,max_depth =3,min_samples_leaf = 5)
clf_entropy.fit(X_train,y_train)
```

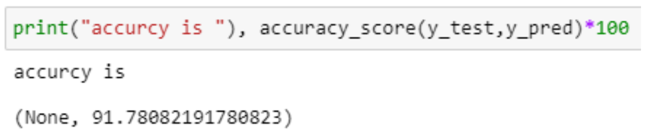
5. Predict the test data set values.



```
y_pred = clf_entropy.predict(X_test)
y_pred
```

```
array(['Fault Free', 'Fault Free', 'Fault Free', 'Fault Free',
       'Fault Free', 'Fault Free', 'Fault Free', 'Fault Free',
       'Fault Free', 'Fault Free', 'Fault Free', 'Fault Free',
       'Fault Free', 'Fault Free', 'Fault Free', 'Fault Free',
       'Fault Free', 'Fault Free', 'Fault Free', 'Fault Free',
       'Fault Free', 'Fault Free', 'Fault Free', 'Fault Free',
       'Fault Free', 'Fault Free', 'Fault Free', 'Fault Free',
       'Fault Free', 'Fault Free', 'Fault at WL', 'Fault Free',
       'Fault Free', 'Fault Free', 'Fault Free', 'Fault Free',
       'Fault Free', 'Fault Free', 'Fault Free', 'Fault Free',
       'Fault Free', 'Fault Free', 'Fault Free', 'Fault Free',
       'Fault Free', 'Fault Free', 'Fault Free', 'Fault Free',
       'Fault Free', 'Fault Free', 'Fault Free', 'Fault Free',
       'Fault Free', 'Fault Free', 'Fault Free', 'Fault Free',
       'Fault Free', 'Fault Free', 'Fault Free', 'Fault Free',
       'Fault Free', 'Fault Free', 'Fault Free', 'Fault Free',
       'Fault Free', 'Fault Free', 'Fault Free', 'Fault Free',
       'Fault Free', 'Fault Free', 'Fault Free', 'Fault Free',
       'Fault Free'], dtype=object)
```

6. Calculate the accuracy of the model.



```
print("accurcy is "), accuracy_score(y_test,y_pred)*100
```

```
accurcy is

(None, 91.78082191780823)
```

Therefore our prediction model shows that there is an excellent accuracy score of 91.78 percent to separate faulty memory cells and also locate the position of the defect irrespective of the technology variation.

Conclusion:

In this paper, we proposed a machine learning based parasitic R, C estimation technique for embedded SRAMs obtaining maximum defect coverage for short and open defects. The proposed method we have used multiple linear regression method to determine the parasitic resistance and capacitance values, and we have used decision tree algorithm to find the faulty and fault free memory cell. The proposed method is implemented to detect the open and short faults which are independent of the technology variation. Using the proposed method we found existing fault models along with an undetectable faults. To get the parasitic R, C values using Microwind 3.9 simulation tool. The experimental results shows excellent accuracy to calculate the parasitic R, C values and also to predict the faults in the SRAM memory cell by using machine learning algorithms.

Declarations:

Funding: Not Applicable

Conflicts of interest/Competing interests: No Conflicts of Interest

Availability of data and material: All simulation results are available with authors.

Code availability: All codes are available with authors.

Authors' contributions: Venkatesham Maddela simulated all the results, Sanjeet Kumar Sinha analyzed the theory behind the simulation result, Muddapu Parvathi formatted the results and conclusion, and Sweta Chander analyzed the results.

References:

1. Semiconductor Industry Association (SIA), "International Technology Road map for semiconductors (ITRS)" 2023.
2. M. Klaus and A. J. Van de Goor, "Test for resistive and capacitive defects in address decoders", Proceedings of IEEE Asian Test Symposium, pp. 31–36, 2001.
3. J. F. Li, K. L. Cheng, C. T. Huang and C. W. Wu, "March-based RAM diagnosis algorithms for stuck-at and coupling faults", Proceedings of IEEE International Test Conference, pp. 758–767, 2001.
4. M.Venkatesham, S.K.Sinha, M Parvathi "Analysis of Open Defect Faults in Single 6T SRAM Cell Using R and C Parasitic Extraction Method", IEEE International Conference on Disruptive Technologies for Multi-Disciplinary Research and Applications (CENTCON-2021) pp.213-217, 2021.
5. M.Venkatesham, S.K.Sinha, M.Parvathi, "Fault Detection and Analysis in embedded SRAM for sub nanometer technology" International Conference on Applied Artificial Intelligence and computing (ICAAIC) 2022.
6. M.Venkatesham, S.K.Sinha and M Parvathi "Extraction of Undetectable Faults in 6T-SRAM Cell", IEEE International Conference on Communication, Control and information Sciences(ICCISc),pp.13-17, 2021.
7. L. Dilillo, P. Girard, S. Pravossoudovitch, A. Virazel and M. Hage-Hassan, "Resistive-open defect injection in SRAM core-cell: analysis and comparison between 0.13 μm and 90 nm technologies", Proceedings of Association for Computing Machinery (ACM) IEEE Design Automation Conference, pp. 857– 862, 2005.
8. M.T.Martins,G.Medeiros,T.Copetti, F.Vargas and L.B.Poehls, "Analyzing NBTI Impact on SRAMs with Resistive- Open Defects", 17th IEEE Latin-American Test Symposium - LATS 2016.
9. C.James, M. Li, C.W.Tseng, and E.J. McCluskey, "Testing for Resistive Opens and Stuck Opens", IEEE ITC International Test Conference, 2001.

10. K. I. Gubbi et al., “Survey of machine learning for electronic design automation,” in Proc. GLSVLSI, 2022, pp. 513–518

11. S. K. Samal , G.Chen, S.K. Lim, “Machine Learning Based Variation Modeling and Optimization for 3D ICs”, Journal of Information and Communication Convergence Engineering, 14(4): 258-267, Dec. 2016.

12. M.Venkatesham, S. K.Sinha and M. Parvathi, “Study on Paradigm of Variable Length SRAM Embedded Memory Testing” Proceedings of the Fifth International Conference on Electronics, Communication and Aerospace Technology (ICECA) 2021.

13. M.Parvathi, K.Satya Prasad ,N. Vasantha, “Testing of Embedded SRAMs Using Parasitic Extraction Method” 9th International Conference on Robotic, Vision, SignalProcessing and Power Applications, 398,(2017).

14. M.Venkatesham, S.K.Sinha and M.Parvathi, V.Sharma, “Comparative Analysis of Open and Short Defects in embedded SRAM using Parasitic Extraction Method for Deep Submicron Technology” Wireless Personal Communications (2023) 132:2123–2141.

Mr. M. Venkatesham, having 14 years of teaching experience, currently working as Assistant Professor in the department of ECE, BVRITH. He is pursuing his PhD from Lovely Professional University, Phagwara, He has awarded with M.Tech from Malla Reddy Engineering College, in 2013. He has done his B.Tech from Aurora’s Engineering College in 2007. He has published papers in various reputed conferences. He is guiding projects for UG students. His research interests are Digital System Design, Low Power VLSI, and Embedded System Designs.

Dr. **Sanjeet Kumar Sinha** was born in Nalanda, India, in 1981. He received the B.Tech. degree in Electronics and Communication Engineering from Jaypee University of Information Technology (JUIT), Waknaghat, Solan, Himachal Pradesh, in 2006, M.Tech in Microelectronics & VLSI design and Ph.D with topic Carbon Nanotube and nanowire FET Devices and its Application in Future VLSI from National Institute of Technology, Silchar, Assam, in 2009 and 2015, respectively, Currently, he is working as Professor at the School of Electronics and Electrical Engineering, Lovely Professional University, Phagwara, Punjab. His research interests include Semiconductor devices and it application for future VLSI applications. Characterization and Analysis of CNTFET, NW-FET and TFET devices. CMOS VLSI circuit design for low power applications. He is a Senior Member of IEEE, IETE Fellow and life member of IEI.

Dr. M. Parvathi, having 21 years of teaching experience, currently working as Professor in the department of ECE, BVRITH. She has awarded with PhD in 2016 from JNTUK, Kakinada, and M.Tech from JNTUH, Hyderabad in 2002. She has done her B.Tech from NITW, Warangal in 1997. She has published papers in various reputed journals and conferences. She has conducted AICTE ATL FDP on Machine learning applications in Micro-Nano VLSI technologies. She has delivered guest lectures under IEEE, IETE centres. She is senior member of IEEE, Life member of ISTE, FIETE. She is guiding projects for UG and PG students. Her research interests are Digital System Design, Low Power VLSI, Testing Techniques, and Embedded Designs.

Dr. Sweta Chander, having 14 years of teaching experience. She has awarded with Ph.D in Engineering from ECE department NIT Silchar, and M.Tech in VLSI from SRM University, Chennai. She has published papers in various reputed journals and conferences. She is a Senior Member of IEEE, IETE Fellow and life member of IEI. Her research interests include semiconductor device physics and modeling as well as VLSI Circuits.