

Ultra Low Power Memory Design for Implantable Pacemaker Chip

M.Tech. Thesis

By

M CHANDU



DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY INDORE
MAY 2026

Ultra Low Power Memory Design for Implantable Pacemaker Chip

Thesis

*Submitted in partial fulfillment of the
requirements for the award of the degree*

of

Master of Technology

by

M CHANDU



DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY INDORE
MAY 2026



INDIAN INSTITUTE OF TECHNOLOGY INDORE

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the thesis entitled **Ultra Low Power Memory Design for Implantable Pacemaker Chip** in the partial fulfillment of the requirements for the award of the degree of **Master of Technology - VLSI Design And Nanoelectronics** and submitted in the Department of Electrical Engineering, Indian Institute of Technology Indore, is an authentic record of my own work carried out during the time period from July 2025 to June 2026 under the supervision of **Dr. Santosh Kumar Vishvakarma**, Professor, Indian Institute of Technology Indore, Indore, India.

The matter presented in this thesis has not been submitted by me for the award of any other degree of this or any other institute.

M Chandu

22/05/2026

Signature of the Student with Date

(M Chandu)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Santosh

May 22, 2026

Signature of the Supervisor of M.Tech Thesis with Date

(Prof. Santosh Kumar Vishvakarma)

M Chandu has successfully given his M.Tech Oral Examination held on **May 7, 2026**

Saptarshi Ghosh

Signature of Supervisor of M.Tech
Thesis

Date: *Santosh*
May 22, 2026

Signature of Convener, DPGC

Date: 22-05-2026

ACKNOWLEDGEMENTS

I am immensely grateful to my M.Tech. thesis supervisor and mentor, **Prof. Santosh Kumar Vishvakarma**, for his constant encouragement, valuable guidance, and continuous support throughout my research work. His insightful suggestions, technical expertise, and motivation greatly contributed to the successful completion of this thesis. I sincerely thank him for inspiring me to improve both academically and personally.

I would also like to express my sincere gratitude to all the members of my thesis evaluation committee for their valuable suggestions, constructive feedback, and thoughtful discussions which broadened my research perspective.

I would also like to acknowledge the support of the **Nanoscale Devices, VLSI Circuit and System Design (NSDCS) Lab** research group. I sincerely thank **Ms. Shivangi Mishra** for her valuable guidance, technical discussions, and continuous support during this work. I also express my gratitude to **Mr. Vijay Pratap Sharma**, **Ms. Akanksha Jain**, and **Mr. Anoop R** for their encouragement and insightful suggestions throughout this research work.

I would also like to express my sincere gratitude to the **Chips to Startup (C2S) Programme** for providing access to Cadence EDA tools and the necessary design infrastructure required for this research work. I am also thankful to the **ChipIN Centre, C-DAC Bangalore**, for their continuous technical support and tool assistance throughout this project.

I am grateful to my friends and labmates for their encouragement and memorable companionship throughout my time at the institute. Their support made this journey both productive and enjoyable.

I am deeply thankful to my family for their unconditional love, encouragement, and continuous support throughout my master's journey. Finally, I thank all those who directly or indirectly contributed to the successful completion of this thesis.

M Chandu

*This thesis is dedicated to
all the people who have supported me
throughout this journey*

ABSTRACT

Implantable pacemakers are life-critical biomedical devices that continuously monitor cardiac activity and regulate abnormal heart rhythms through electrical stimulation. Modern pacemaker systems require reliable and low-power memory architectures for buffering and storing electrogram (EGM) signals acquired from sensing channels. Existing pacemaker architectures mainly focus on sensing, stimulation, and analog front-end circuitry, while efficient large-scale digital memory organization and low-power external memory interfacing receive limited attention.

This thesis presents the design and implementation of a low-power SPI-enabled SRAM architecture for implantable pacemaker applications. The proposed architecture integrates a Serial Peripheral Interface (SPI) communication protocol with a banking-based SRAM organization to support efficient external memory access and EGM signal buffering. An 8192×8 single-port SRAM macro is implemented as the basic memory block, and eight SRAM macros are integrated to construct a 64 KB memory system using address partitioning and decoder-based bank selection.

The complete RTL-to-GDSII implementation flow is carried out using Cadence EDA tools in SCL 180 nm CMOS technology including synthesis, floorplanning, placement, CTS, routing, and physical verification. The final implementation achieves a maximum operating frequency of 48.95 MHz with total power consumption of approximately 2 mW.

The proposed SPI-enabled SRAM architecture demonstrates an efficient and scalable low-power memory solution suitable for implantable biomedical systems requiring continuous biomedical signal buffering.

Contents

Abstract	i
List of Figures	v
List of Tables	vii
List of Nomenclature & Acronyms	vii
1 Introduction	1
1.1 Introduction to Implantable Biomedical Devices . . .	1
1.2 Implantable Pacemaker System	3
1.2.1 Overview of Pacemaker	5
1.2.2 Working Principle of Pacemaker	5
1.2.3 Electrogram (EGM) Signal Monitoring	6
1.2.4 Memory Requirement in Pacemakers	8
1.3 Motivation for Low-Power Memory Design	10
1.4 Existing Memory Architectures and Limitations . . .	12
1.5 Problem Statement	14
1.6 Objectives of the Thesis	14
1.7 Organization of the Thesis	15
2 Literature Review	17
2.1 SRAM Basics	17
2.1.1 SRAM Architecture	17
2.1.2 Conventional 6T SRAM Cell	18
2.1.3 SRAM Hold Operation	19
2.1.4 SRAM Read Operation	19
2.1.5 SRAM Write Operation	21

2.1.6	Static Noise Margin (SNM)	22
2.1.7	Challenges in SRAM Design	23
2.1.8	Selection of SPI Interface for Biomedical Mem- ory Systems	24
2.2	SPI Communication Protocol	26
2.2.1	SPI Architecture	27
2.2.2	SPI Modes	27
2.2.3	SPI Timing Diagram	28
2.2.4	Advantages of SPI Communication	29
2.3	Literature Review	29
2.4	Existing SRAM Architectures	32
2.4.1	Research Gap	32
3	Proposed SPI Enabled SRAM Architecture	34
3.1	Proposed 64 KB SRAM Architecture	34
3.1.1	SRAM Read Operation	36
3.1.2	SRAM Write Operation	38
3.1.3	SRAM Hold Operation	38
3.2	SPI Interface Design	39
3.2.1	SPI Finite State Machine (FSM) Operation .	41
3.2.2	SPI Timing Operation	42
3.3	Integration of SPI with SRAM	43
3.4	Overall System Operation	44
4	RTL to GDSII Implementation Flow	46
4.1	ASIC Design Methodology for SPI-Enabled SRAM .	46
4.2	RTL to GDSII Implementation Flow	47
4.2.1	RTL Design	48
4.2.2	Design Specification	48
4.2.3	Architectural Definition	49
4.2.4	Micro-Architectural Definition	49
4.3	RTL Verification	49
4.4	Advanced RTL Verification Checks	50
4.4.1	Lint Checking	50
4.4.2	CDC and RDC Verification	50
4.4.3	X-Propagation Verification	51

4.5	Logic Synthesis	51
4.5.1	Synthesis Reports	52
4.6	Logic Equivalence Checking (LEC)	52
4.7	Post Synthesis Gate Level Simulation (GLS)	53
4.8	Floorplanning	53
4.8.1	Macro Planning	54
4.8.2	MMMC Configuration	54
4.9	Power Planning	54
4.9.1	Power Distribution Network	55
4.10	Placement	55
4.11	Clock Tree Synthesis (CTS)	56
4.12	Routing	56
4.13	Post-Route Logic Equivalence Checking	57
4.14	Post-Route Gate Level Simulation (GLS)	57
4.15	Signoff Verification	58
4.15.1	Physical Signoff	58
4.15.2	Timing Signoff	58
4.16	Physical Verification	59
4.16.1	Design Rule Check (DRC)	59
4.16.2	Antenna Checks	59
4.16.3	Layout Versus Schematic (LVS)	60
5	Results and Performance Analysis	61
5.1	64KB SRAM Results	61
5.1.1	SRAM Macro Verification	61
5.1.2	64 KB SRAM Verification	62
5.1.3	Synthesis Results	63
5.1.4	Synthesis Summary	64
5.1.5	Post Synthesis LEC Results	65
5.1.6	Post Synthesis GLS Results	66
5.1.7	Floor Planning and Macro Planning Results	66
5.1.8	Power Planning and Placement Results	67
5.1.9	CTS and Routing Results	68
5.1.10	Post Route LEC	68
5.1.11	Post Route GLS	69
5.1.12	Signoff & Verification	69

5.2	SPI Enabled SRAM Results	70
5.2.1	SPI Enabled SRAM Verification	70
5.2.2	Synthesis Results	71
5.2.3	Synthesis Summary	72
5.2.4	Post Synthesis LEC Results	73
5.2.5	Post Synthesis GLS Results	73
5.2.6	Physical Design Results	74
5.2.7	Post Route GLS Results	74
5.2.8	Post Route LEC Results	75
5.2.9	Signoff & Verification	75
5.3	Final Results Summary	76
6	Conclusion and Future Scope	78
6.1	Summary of Work	78
6.2	Conclusion	79
6.3	Limitations	79
6.4	Future Scope	80

List of Figures

1.1	Implantable Pacemaker System	3
1.2	Functional block diagram of implantable pacemaker system	4
1.3	Typical EGM waveform characteristics	7
1.4	Cardiac conduction system	8
2.1	SRAM Sub-array Diagram with Peripheral Circuits .	18
2.2	Conventional 6T SRAM Cell	19
2.3	SRAM Read Operation	20
2.4	SRAM Read Timing Diagram	20
2.5	SRAM Write Operation	21
2.6	SRAM Write Timing Diagram	21
2.7	Butterfly Curve for SRAM HSNM	22
2.8	Butterfly Curve for SRAM RSNM	23
2.9	Butterfly Curve for SRAM WSNM	23
2.10	Basic SPI communication interface between master and slave devices.	27
2.11	Timing diagrams of different SPI modes.	28
3.1	Proposed 64 KB SPRAM architecture using eight 8192×8 SRAM macros.	35
3.2	Read cycle timing diagram of the SRAM macro. . . .	37
3.3	Write cycle timing diagram of the SRAM macro. . . .	39
3.4	SPI interface connected with SRAM memory	40
3.5	SPI interface internal architecture	40
3.6	Finite State Machine (FSM) of the SPI memory controller	42
3.7	Write Timing Diagram	43

3.8	Read Timing Diagram	43
4.1	Complete RTL to GDSII implementation flow.	47
5.1	8192x8 (8KB) Macro Simulation	61
5.2	8192x8 (8KB) Macro Simulation Console Output	62
5.3	64KB SRAM Simulation	62
5.4	64KB SRAM Simulation Console Output	63
5.5	64KB SRAM Schematic	63
5.6	64KB SRAM Area Report	64
5.7	64KB SRAM Power Report	64
5.8	LEC Compare Results	65
5.9	LEC Comparison	65
5.10	Zero Delay Gate Level Simulation	66
5.11	Unit Delay Gate Level Simulation	66
5.12	SDF Gate Level Simulation	66
5.13	After FloorPlannig & Macro Placement	66
5.14	After Power planning & Placement	67
5.15	After CTS & Routing	68
5.16	Post Route LEC Comparison	68
5.17	Zero Delay Simulation	69
5.18	Unit Delay Simulation	69
5.19	Post Route GLS SDF Simulation	69
5.20	Signoff	69
5.21	DRC Report	70
5.22	SPI Enabled Memory Simulation	70
5.23	SPI SRAM Schematic	71
5.24	SPI SRAM Area Report	71
5.25	SPI SRAM Power Report	72
5.26	LEC Comparison	73
5.27	Gate Level Simulation	73
5.28	After Routing Stage	74
5.29	Gate Level Simulation	74
5.30	LEC Comparison	75
5.31	Signoff	75
5.32	DRC Report	75

List of Tables

2.1	Comparison of Communication Protocols	25
2.2	SPI Communication Modes	28
5.1	Performance Summary of Proposed 64KB SRAM Architecture	65
5.2	Performance Summary of Proposed 64KB SPI-Enabled SRAM Architecture	72
5.3	Comparison of 64 KB SRAM Architecture With and Without SPI Interface	76

List of Nomenclature & Acronyms

ADE	:	Analog Design Environment
ASIC	:	Application-Specific Integrated Circuit
CDC	:	Clock Domain Crossing
CPHA	:	Clock Phase
CPOL	:	Clock Polarity
CRT	:	Cardiac Resynchronization Therapy
CS	:	Chip Select
CTS	:	Clock Tree Synthesis
DDR	:	Double Data Rate
DRAM	:	Dynamic Random Access Memory
DRC	:	Design Rule Check
DRV	:	Design Rule Violation
ECG	:	Electrocardiogram
EDA	:	Electronic Design Automation
EGM	:	Electrogram
EMI	:	Electromagnetic Interference
ESD	:	Electrostatic Discharge
FIFO	:	First-In First-Out
FSM	:	Finite State Machine
GDSII	:	Graphic Data System II
GLS	:	Gate Level Simulation
HDL	:	Hardware Description Language
HSNM	:	Hold Static Noise Margin
ICD	:	Implantable Cardioverter Defibrillator
IO	:	Input Output

IoT	:	Internet of Things
IR	:	Infrared
LEC	:	Logic Equivalence Checking
LVS	:	Layout Versus Schematic
MISO	:	Master Input Slave Output
MMMC	:	Multi-Mode Multi-Corner
MOSI	:	Master Output Slave Input
NMOS	:	N-type Metal-Oxide-Semiconductor
OEB	:	Output Enable Bar
PDK	:	Process Design Kit
PISO	:	Parallel-In Serial-Out
PMOS	:	P-type Metal-Oxide-Semiconductor
PnR	:	Place and Route
PVT	:	Process Voltage Temperature
RCA	:	Ripple Carry Adder
RDC	:	Reset Domain Crossing
RSNM	:	Read Static Noise Margin
RTL	:	Register Transfer Level
SAR	:	Successive Approximation Register
SCLK	:	Serial Clock
SCL	:	Semiconductor Complex Limited
SDF	:	Standard Delay Format
SNM	:	Static Noise Margin
SoC	:	System-on-Chip
SPI	:	Serial Peripheral Interface
SPRAM	:	Single Port Random Access Memory
SRAM	:	Static Random Access Memory
STA	:	Static Timing Analysis
SIPO	:	Serial-In Parallel-Out
VLSI	:	Very Large Scale Integration
VTC	:	Voltage Transfer Characteristic
WEB	:	Write Enable Bar
WL	:	Word Line
WSNM	:	Write Static Noise Margin
XPROP	:	X-Propagation

Chapter 1

Introduction

1.1 Introduction to Implantable Biomedical Devices

Implantable biomedical devices have become an essential part of modern healthcare systems due to their ability to continuously monitor, diagnose, and assist critical physiological functions within the human body. These devices are designed to operate inside the body for extended durations while ensuring reliable performance, minimal power consumption, and patient safety. Rapid advancements in semiconductor technology, low-power integrated circuits, and wireless communication systems have significantly improved the functionality and efficiency of implantable medical electronics. As a result, implantable systems are increasingly being used for the treatment and monitoring of various chronic medical conditions.

Common examples of implantable biomedical devices include cardiac pacemakers, implantable cardioverter defibrillators (ICDs), cochlear implants, neurostimulators, insulin pumps, and biomedical monitoring systems. Among these, implantable cardiac pacemakers are one of the most widely used life-supporting devices for patients suffering from cardiac arrhythmias and abnormal heart rhythms. These devices continuously monitor the electrical activity of the heart and provide electrical stimulation whenever irregular or insufficient heartbeats are detected.

One of the most critical design requirements of implantable biomedical devices is ultra-low power operation. Since these systems are powered by compact batteries that are surgically implanted within the body, replacing the battery frequently is highly undesirable and may expose patients to additional surgical risks. Therefore, implantable devices are expected to operate reliably for several years while consuming minimal power. In addition to low power consumption, high reliability, low heat generation, compact chip area, and robust operation under varying environmental conditions are also important requirements for biomedical integrated circuits.

Modern implantable biomedical systems increasingly rely on embedded digital processing and memory units for storing physiological data, buffering sensed signals, and supporting communication with external monitoring equipment. In cardiac pacemakers, electrical signals generated by the heart, commonly referred to as electrogram (EGM) signals, must often be temporarily stored and processed for diagnostic and therapeutic purposes. The growing demand for continuous monitoring and multi-channel signal acquisition has increased the requirement for efficient on-chip memory architectures in biomedical devices.

Conventional memory systems used in general-purpose electronic applications primarily focus on high speed and large storage capacity. However, implantable biomedical devices require memory architectures optimized for low leakage power, stable low-voltage operation, and long-term reliability rather than extremely high operating speed. As a result, the design of energy-efficient memory subsystems has become an important research area in biomedical system-on-chip (SoC) development.

To address these challenges, low-power SRAM-based memory architectures integrated with compact communication interfaces have gained significant attention for implantable applications. Such architectures can provide reliable data storage, reduced power consumption, and efficient communication while satisfying the strict constraints imposed by biomedical devices.

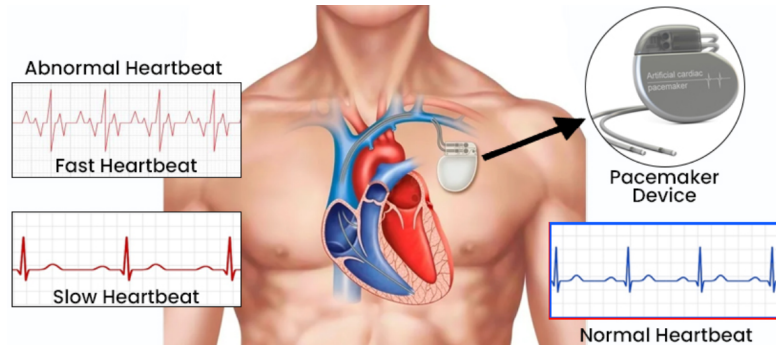


Figure 1.1: Implantable Pacemaker System

1.2 Implantable Pacemaker System

Implantable pacemakers are compact biomedical electronic devices designed to regulate abnormal heart rhythms by delivering controlled electrical stimulation to the heart muscles. These devices are primarily used for patients suffering from arrhythmias, bradycardia, heart block, and other cardiac conduction disorders in which the natural electrical pacing activity of the heart becomes irregular or insufficient. Modern pacemakers continuously monitor cardiac activity and provide pacing pulses only when necessary, thereby ensuring reliable and energy-efficient operation.

A typical implantable pacemaker consists of several functional blocks, including sensing circuitry, pacing circuitry, telemetry units, control logic, memory subsystems, battery management circuits, and electrode interfaces. These blocks work together to monitor cardiac signals, process physiological information, store diagnostic data, and generate appropriate pacing signals for maintaining normal heart rhythm.

The controller interface acts as the central processing unit of the pacemaker and coordinates communication among all internal functional blocks. It controls sensing operations, pacing decisions, telemetry communication, memory access, and battery management. The telemetry block enables wireless communication between the implanted pacemaker and external medical monitoring equipment, allowing physicians to configure device parameters and retrieve diagnostic information without invasive procedures.

1.2.1 Overview of Pacemaker

The human heart functions through synchronized electrical impulses that control the contraction and relaxation of cardiac muscles. Under normal conditions, these electrical impulses originate from the sinoatrial (SA) node, which acts as the natural pacemaker of the heart. However, due to aging, cardiovascular diseases, or abnormalities in the cardiac conduction system, the heart may fail to maintain an appropriate rhythm. Such conditions can lead to irregular heartbeats, reduced blood circulation, fatigue, dizziness, and potentially life-threatening complications.

An implantable pacemaker is designed to compensate for these abnormalities by monitoring the heart's electrical activity and generating artificial pacing pulses whenever the intrinsic heart rate falls below a predefined threshold. The pacemaker is surgically implanted beneath the skin near the chest region and connected to the heart through insulated leads and electrodes.

Modern pacemakers support various pacing modes depending on the patient's medical condition. Single-chamber pacemakers stimulate either the atrium or ventricle, whereas dual-chamber pacemakers coordinate stimulation between both chambers to maintain synchronized cardiac activity. Advanced cardiac resynchronization therapy (CRT) pacemakers additionally stimulate the left ventricle to improve cardiac efficiency in patients with heart failure.

In addition to pacing functionality, modern implantable pacemakers incorporate sensing, data logging, telemetry communication, and diagnostic monitoring capabilities. These additional functionalities require efficient digital processing and memory support while maintaining ultra-low power consumption.

1.2.2 Working Principle of Pacemaker

The operation of an implantable pacemaker is based on continuous monitoring and regulation of the electrical activity of the heart. The sensing electrodes placed inside the heart chambers detect intrinsic cardiac signals generated during each heartbeat. These signals are

processed by the sensing circuitry and forwarded to the controller unit for analysis.

The controller compares the detected heart rate and rhythm against predefined pacing thresholds programmed by medical professionals. If the controller determines that the heart is beating normally, no pacing action is initiated. However, when the heart rate falls below the programmed limit or irregular activity is detected, the pacing block generates electrical stimulation pulses that are delivered to the heart through pacing electrodes.

The pacing pulses depolarize cardiac muscle tissue and trigger controlled heart contractions, thereby restoring a normal rhythm. This closed-loop monitoring and stimulation process continues continuously throughout the operation of the device.

The telemetry block enables bidirectional wireless communication between the implanted device and external programmers. Physicians can use telemetry communication to modify pacing parameters, retrieve stored patient data, monitor battery status, and analyze recorded electrogram signals.

Battery management circuitry continuously monitors supply voltage and power consumption to maximize operational lifetime. Since the device must operate reliably for many years using a limited battery source, all functional blocks are carefully optimized for low-power operation.

1.2.3 Electrogram (EGM) Signal Monitoring

Electrogram (EGM) signals are intracardiac electrical signals sensed directly from the heart chambers through implanted electrodes. These signals contain valuable diagnostic information regarding cardiac rhythm, conduction abnormalities, arrhythmia events, and pacing effectiveness. Continuous monitoring of EGM signals enables the pacemaker to make accurate pacing decisions and helps physicians analyze the long-term cardiac condition of patients.

Modern implantable pacemakers often support multiple sensing channels for monitoring signals from different chambers of the heart, including the atrium, ventricle, and left ventricle. The sensed EGM

signals are amplified, filtered, digitized, and processed within the device. These signals may also be temporarily buffered or stored for diagnostic analysis and telemetry transmission.

Accurate EGM monitoring requires reliable sensing circuitry and efficient signal storage mechanisms. Since cardiac signals are continuously generated during device operation, the pacemaker must support low-power data buffering and memory management techniques to avoid excessive energy consumption. Furthermore, the memory architecture must provide stable operation and reliable data retention under low-voltage operating conditions.

The increasing requirement for continuous monitoring and multi-channel signal acquisition has significantly increased the importance of efficient memory subsystems in modern implantable pacemakers.

Figure 1.3 illustrates the cardiac conduction system and a typical electrocardiogram (ECG/EGM) waveform. The P-wave represents atrial depolarization caused by electrical activation of the atria through the sinoatrial (SA) node. The PR interval indicates signal propagation from the atria to the ventricles through the atrioventricular (AV) node.

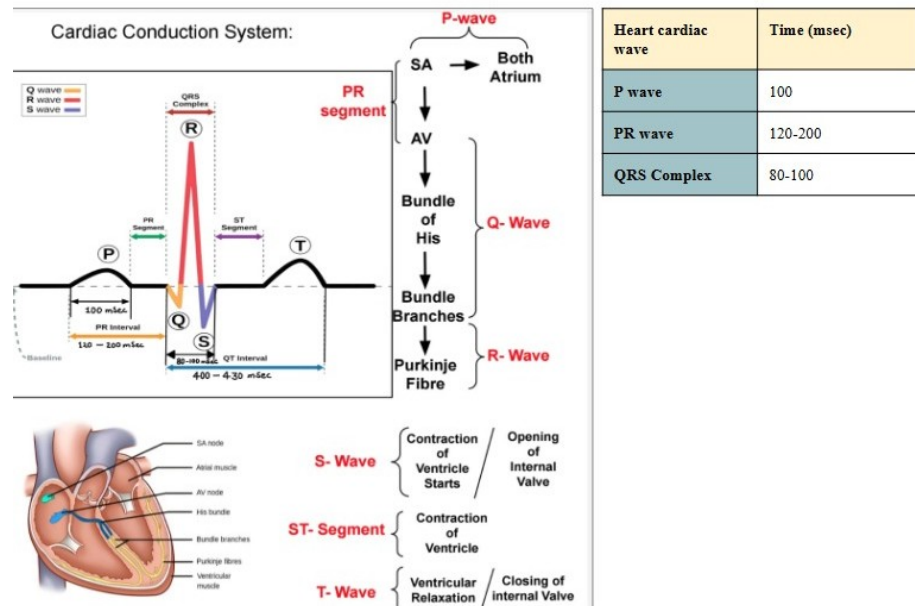


Figure 1.3: Typical EGM waveform characteristics

The QRS complex corresponds to ventricular depolarization and

represents the main contraction phase of the ventricles. The Q-wave indicates the initial depolarization phase, the R-wave represents the major ventricular activation peak, and the S-wave corresponds to the final ventricular depolarization stage. The T-wave represents ventricular repolarization or relaxation.

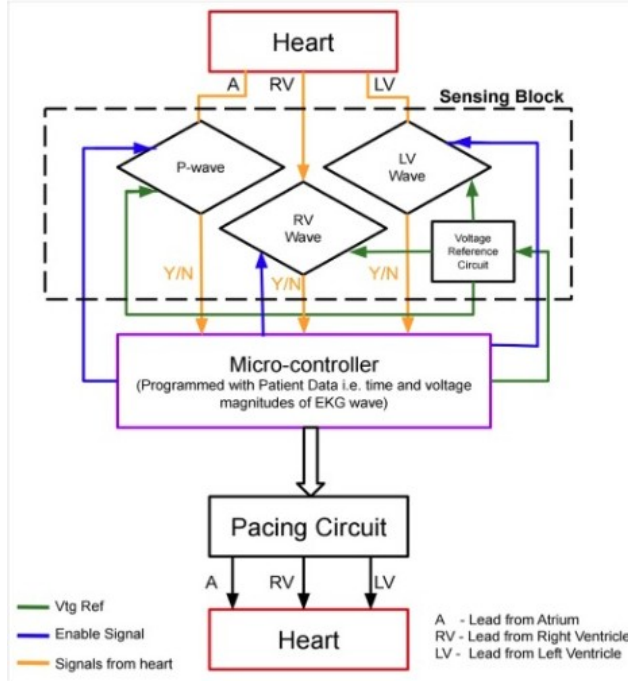


Figure 1.4: Cardiac conduction system

The timing characteristics of these cardiac waves are important for detecting abnormalities in heart rhythm and conduction behavior. Implantable pacemakers continuously monitor these signals and store critical EGM events for diagnosis, arrhythmia detection, and telemetry analysis.

1.2.4 Memory Requirement in Pacemakers

Memory subsystems play an important role in modern implantable pacemakers by supporting temporary data storage, event logging, signal buffering, and device configuration management. Unlike conventional computing systems that require large-capacity memories for general-purpose processing, pacemaker systems require

compact, highly reliable, and ultra-low-power memory architectures optimized for long-term biomedical operation.

In implantable pacemakers, the sensing block continuously monitors cardiac activity and acquires electrogram (EGM) signals from different chambers of the heart. These sensed signals are first processed by the controller interface, which performs signal analysis, rhythm evaluation, and pacing decision operations. Based on the detected cardiac activity, the controller generates event information such as abnormal heart rhythms, pacing events, timing information, and diagnostic records. This event-related data is then transferred to the memory subsystem for temporary buffering and storage.

The stored data can later be accessed by the telemetry unit and transmitted to external medical monitoring equipment for diagnostic analysis and device programming. In addition to event storage, the memory subsystem may also store pacing parameters, configuration settings, device status information, and operational history required for reliable long-term device operation.

As modern pacemakers increasingly support continuous monitoring and multi-channel sensing, the amount of generated physiological data has increased significantly. Therefore, the memory architecture must provide: reliable data retention, low-latency access, low standby leakage power, and stable operation under low supply voltages.

Among the available semiconductor memory technologies, Static Random Access Memory (SRAM) is widely preferred for implantable biomedical applications due to its fast access time, low operational latency, and simple interface integration. SRAM stores data using bistable latch structures and does not require periodic refresh operations for data retention during active operation. This characteristic makes SRAM highly suitable for low-power biomedical systems where energy efficiency is extremely important.

In contrast, Dynamic Random Access Memory (DRAM) stores data in capacitive elements and requires continuous refresh cycles to maintain stored information. These periodic refresh operations increase power consumption and introduce additional control complexity, making DRAM less suitable for ultra-low-power implantable

devices. Furthermore, DRAM sensing circuitry and refresh controllers increase system overhead and may reduce long-term energy efficiency.

Non-volatile memories such as Flash memory and EEPROM provide permanent data storage but generally exhibit slower write speeds, higher write energy consumption, and limited write endurance. Since pacemakers may require frequent event updates and continuous data buffering, such memories are less efficient for real-time temporary signal storage applications.

SRAM also offers better compatibility with standard CMOS technology and enables easier integration with digital controllers and communication interfaces such as SPI. In addition, SRAM provides high reliability, low access delay, and simpler peripheral circuitry compared to several alternative memory technologies. These advantages make SRAM a suitable choice for low-power implantable pacemaker systems requiring reliable and efficient short-term physiological data storage.

Therefore, the development of an energy-efficient SRAM-based memory subsystem capable of supporting event-driven data buffering and low-power communication has become an important research area in implantable biomedical electronics..

1.3 Motivation for Low-Power Memory Design

The rapid advancement of implantable biomedical devices has significantly increased the demand for compact, reliable, and energy-efficient electronic systems capable of operating continuously for extended durations. Among these devices, implantable cardiac pacemakers require highly optimized hardware architectures due to their life-critical functionality and stringent power constraints. Since pacemakers are powered using miniature implanted batteries, minimizing power consumption is one of the most important design objectives in modern biomedical electronics. Frequent battery replacement is undesirable because it requires surgical intervention,

increases healthcare costs, and introduces additional risks to patients. Therefore, every subsystem within the pacemaker must be carefully designed to maximize operational lifetime while maintaining reliable performance.

Modern pacemakers perform much more than simple cardiac pacing operations. In addition to generating stimulation pulses, they continuously monitor intracardiac electrical activity, detect abnormal heart rhythms, record diagnostic information, and communicate with external medical devices through telemetry interfaces. These functionalities generate a significant amount of physiological and event-related data that must be temporarily buffered and processed within the device. As the complexity of monitoring algorithms and multi-channel sensing capabilities continues to increase, the requirement for efficient low-power memory architectures has become increasingly important.

Conventional pacemaker systems generally utilize limited internal memory resources and rely on event-driven storage mechanisms. However, modern diagnostic and monitoring requirements demand improved memory support for buffering electrogram (EGM) signals, storing pacing history, maintaining configuration parameters, and recording abnormal cardiac events. External memory solutions can provide additional storage capacity, but they often introduce increased interface overhead, higher communication latency, larger routing complexity, and additional power consumption. These limitations are particularly critical in implantable biomedical systems where energy efficiency and reliability are essential.

To address these challenges, low-power SRAM-based memory systems integrated with compact serial communication interfaces have emerged as a promising solution for biomedical applications. Static Random Access Memory (SRAM) offers several advantages such as fast access time, low operational latency, high reliability, and compatibility with CMOS technology. Furthermore, SRAM does not require periodic refresh operations unlike DRAM, thereby reducing power consumption and simplifying control circuitry. These characteristics make SRAM highly suitable for temporary buffering

and real-time storage applications in implantable devices.

The use of Serial Peripheral Interface (SPI) communication further improves system efficiency by reducing pin count, minimizing routing complexity, and simplifying memory interfacing. SPI-based memory architectures are widely used in embedded systems due to their simple hardware implementation and low communication overhead. Integrating SPI-compatible SRAM systems within biomedical devices can therefore provide scalable and energy-efficient data storage solutions while maintaining compact chip area and reliable operation.

Another important motivation for this work is the increasing emphasis on complete system-level integration in biomedical System-on-Chip (SoC) designs. Modern implantable devices require memory architectures that can be efficiently integrated with digital controllers, sensing blocks, telemetry units, and power management circuitry. In addition, these systems must operate reliably under low-voltage conditions while satisfying stringent timing, power, and area constraints.

Based on these requirements, this thesis focuses on the design and implementation of a low-power SPI-enabled SRAM architecture suitable for implantable pacemaker applications. The proposed work aims to develop a scalable memory subsystem capable of supporting reliable event-driven data storage while minimizing power consumption, communication overhead, and implementation complexity.

1.4 Existing Memory Architectures and Limitations

Existing implantable pacemaker systems primarily focus on cardiac sensing, pacing control, telemetry communication, and analog front-end circuitry. Although modern devices support basic event logging and physiological data storage, the development of efficient large-scale memory architectures for biomedical applications has received comparatively less attention. Most conventional pacemaker systems utilize limited internal memory resources that are mainly op-

timized for storing event-driven information rather than continuous multi-channel signal buffering.

With the increasing demand for continuous cardiac monitoring and advanced diagnostic capabilities, modern pacemakers require improved memory support for storing electrogram (EGM) data, pacing history, configuration parameters, and abnormal event records. However, integrating larger memory subsystems within implantable devices introduces several challenges related to power consumption, reliability, area overhead, and interface complexity.

Several existing embedded memory architectures rely on external memory communication interfaces for additional storage support. Although such approaches improve storage capacity, they may increase communication latency, routing complexity, and overall system power consumption. These limitations become critical in implantable biomedical systems where ultra-low-power operation and long-term reliability are essential requirements.

SPI-based SRAM systems have been widely explored for embedded and IoT applications because of their simple interface structure and reduced pin count. However, many of these architectures are primarily optimized for consumer electronics and low-cost embedded systems rather than implantable biomedical environments. Biomedical applications require memory systems that provide stable low-voltage operation, reliable data retention, low leakage power, and compact physical implementation.

In addition, conventional memory architectures often prioritize high operating speed and storage density over energy efficiency and reliability. Such design approaches may not be suitable for implantable devices that operate continuously using limited battery energy for several years. Therefore, there exists a need for a scalable and low-power SRAM-based memory architecture specifically optimized for implantable pacemaker applications.

1.5 Problem Statement

Modern implantable pacemakers increasingly require efficient memory subsystems to support event-driven data storage, electrogram signal buffering, and continuous physiological monitoring. However, existing memory architectures used in embedded systems are not specifically optimized for the stringent requirements of implantable biomedical applications. Conventional external memory solutions often introduce higher communication overhead, increased latency, larger routing complexity, and additional power consumption, which can significantly reduce device lifetime.

Furthermore, implantable pacemakers operate under strict constraints related to power consumption, reliability, chip area, and long-term stable operation. The memory subsystem must therefore provide efficient data storage while minimizing leakage power and interface complexity. Existing SPI-based SRAM architectures are primarily targeted toward IoT and consumer applications and may not adequately address the ultra-low-power and reliability requirements of implantable biomedical systems.

Therefore, there is a need for a low-power and scalable SRAM-based memory architecture capable of supporting reliable event buffering and efficient communication for implantable pacemaker applications. The proposed work aims to address these challenges through the design and implementation of an SPI-enabled SRAM memory subsystem optimized for biomedical operation.

1.6 Objectives of the Thesis

The primary objective of this thesis is to design and implement a low-power SPI-enabled SRAM memory architecture suitable for implantable pacemaker applications. The specific objectives of this work are as follows:

- To design a low-power SRAM-based memory subsystem for implantable biomedical applications.

- To implement an SPI-compatible memory interface for efficient communication between the controller and memory unit.
- To construct a 64 KB SRAM architecture using memory macro integration and banking techniques.
- To minimize power consumption and interface complexity while maintaining reliable memory operation.
- To perform complete ASIC design implementation and physical verification of the proposed architecture using Cadence EDA tools.
- To evaluate the proposed architecture in terms of power, timing, area, and physical design performance.

1.7 Organization of the Thesis

The thesis is organized into six chapters covering the background, design methodology, implementation, and evaluation of the proposed SPI-enabled SRAM architecture for implantable pacemaker applications.

- **Chapter 1** presents the introduction and motivation for low-power memory design in implantable biomedical systems. It discusses the importance of efficient memory architectures in pacemakers, introduces SRAM and SPI communication, identifies existing limitations, and outlines the objectives and contributions of the proposed work.
- **Chapter 2** provides the theoretical background and literature review related to SRAM memory architectures, SPI communication protocols, low-power memory design techniques, and previously reported biomedical memory systems. The chapter also discusses different SRAM cell architectures and their design trade-offs.
- **Chapter 3** describes the proposed SPI-enabled SRAM architecture in detail. It presents the overall system organization,

SRAM macro integration, SPI interface design, memory banking structure, control logic, and functional operation of the proposed memory subsystem.

- **Chapter 4** explains the RTL-to-GDSII implementation flow of the proposed architecture using Cadence design tools. This chapter includes synthesis, floorplanning, power planning, placement, clock tree synthesis, routing, physical verification, and post-route validation.
- **Chapter 5** presents the simulation and implementation results of the proposed design. The performance of the SRAM subsystem is analyzed in terms of power consumption, timing, operating frequency, area utilization, and physical design metrics. The chapter also discusses post-route verification and system-level performance analysis.
- **Chapter 6** concludes the thesis by summarizing the major contributions and outcomes of the work. The chapter also outlines possible future research directions for improving low-power SRAM architectures for biomedical applications.

Chapter 2

Literature Review

2.1 SRAM Basics

Static Random Access Memory (SRAM) is one of the most widely used semiconductor memory technologies in modern digital systems due to its high speed, low latency, and compatibility with CMOS technology. Unlike Dynamic Random Access Memory (DRAM), SRAM does not require periodic refreshing because the stored data is maintained as long as the power supply remains active. SRAM is extensively employed in cache memories, embedded systems, microcontrollers, and high-performance computing architectures.

An SRAM array is generally composed of memory cells arranged in rows and columns along with peripheral circuits such as row decoders, column multiplexers, sense amplifiers, write drivers, and precharge circuitry. The complementary bit-line architecture used in SRAM improves read speed and noise immunity. During read operations, sense amplifiers detect small voltage differences between the bit lines and convert them into logic values. During write operations, write drivers force the desired data onto the bit lines.

2.1.1 SRAM Architecture

A typical SRAM architecture consists of a memory cell array together with supporting peripheral circuitry. The row decoder activates a selected word line based on the input address, while the

column multiplexer selects the appropriate bit-line pair corresponding to the accessed memory location. Precharge circuits initialize the bit lines before read operations, and sense amplifiers are used to detect small voltage differences during data retrieval.

The SRAM array operates using complementary bit lines, denoted as BL and BLB, and a word line (WL). The complementary bit-line structure improves noise tolerance and increases sensing reliability. Figure 2.1 illustrates the SRAM sub-array organization and associated peripheral circuits.

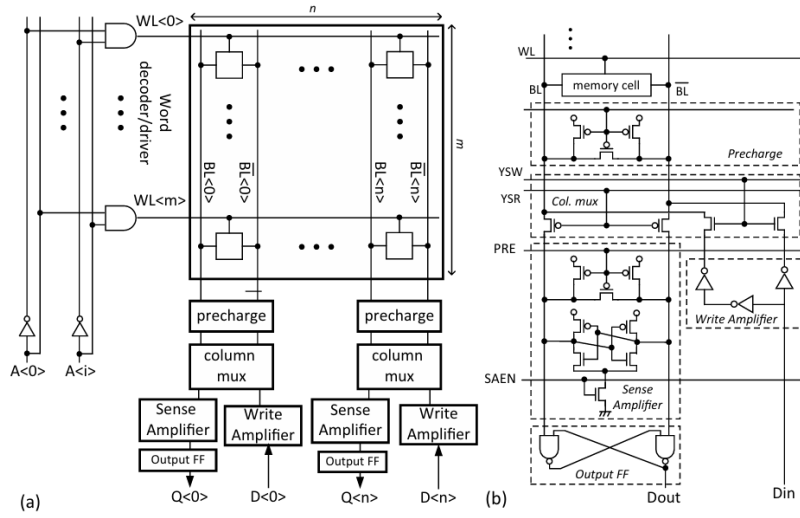


Figure 2.1: SRAM Sub-array Diagram with Peripheral Circuits

2.1.2 Conventional 6T SRAM Cell

The conventional SRAM cell consists of six transistors and is commonly referred to as the 6T SRAM cell. It is formed using two cross-coupled CMOS inverters and two access transistors. The cross-coupled inverters provide bistable operation and store one bit of information, while the access transistors connect the storage nodes to the bit lines during read and write operations.

The six transistors are categorized into:

- Pull-Up (PU) PMOS transistors,
- Pull-Down (PD) NMOS transistors,

- Pass-Gate (PG) NMOS access transistors.

The internal storage nodes are commonly represented as N_1 and N_2 , where complementary logic values are stored. If N_1 stores logic '1', then N_2 stores logic '0', and vice versa. The access transistors are controlled using the word line signal.

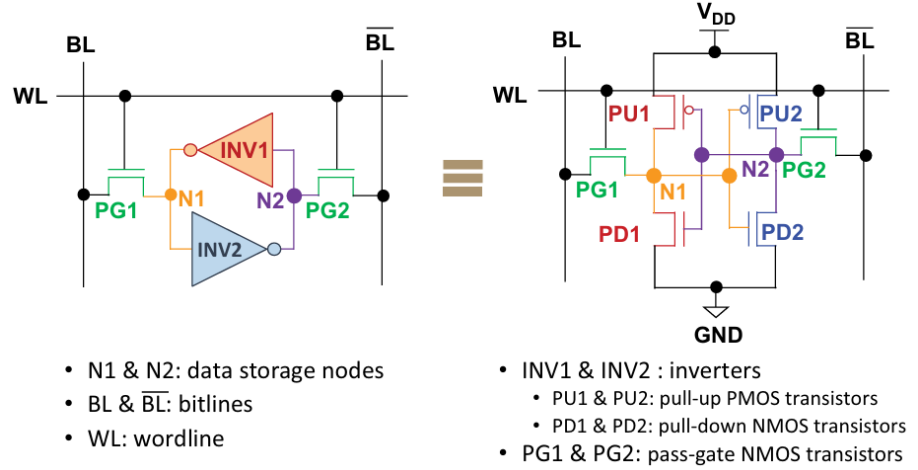


Figure 2.2: Conventional 6T SRAM Cell

2.1.3 SRAM Hold Operation

During hold operation, the SRAM cell preserves the stored data without performing either read or write operations. In this mode, the word line is maintained at logic low, thereby turning OFF the access transistors and isolating the storage nodes from the bit lines. The positive feedback provided by the cross-coupled inverters maintains the stored logic state continuously as long as power is supplied.

The hold operation is essential for low-power standby conditions where memory retention is required without active access. The stability of the SRAM cell during this mode is generally characterized using Hold Static Noise Margin (HSNM).

2.1.4 SRAM Read Operation

In a read operation, both bit lines are initially precharged to the supply voltage V_{DD} . Once precharging is completed, the word line is activated, enabling the access transistors. Depending on the

stored data, one of the bit lines begins to discharge slightly through the pull-down transistor of the cell. This small differential voltage between BL and BLB is then amplified using a sense amplifier to generate the output data.

For example, if node N_1 stores logic '0', the corresponding bit line discharges slightly while the complementary bit line remains near V_{DD} . The sense amplifier detects this voltage difference and converts it into a digital output. Since the read operation disturbs the storage node slightly, SRAM cells must be carefully designed to ensure sufficient read stability.

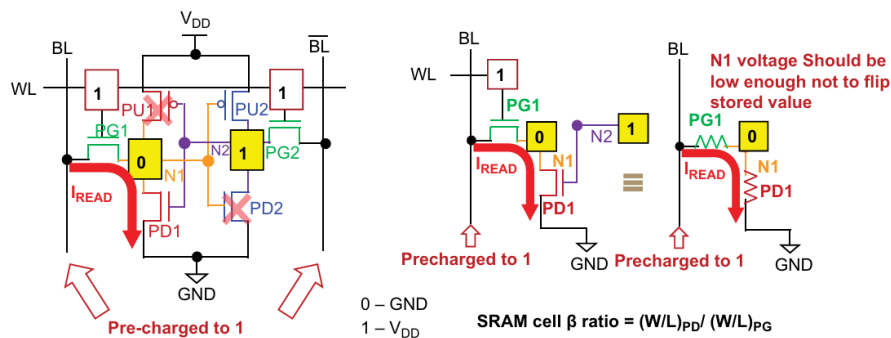


Figure 2.3: SRAM Read Operation

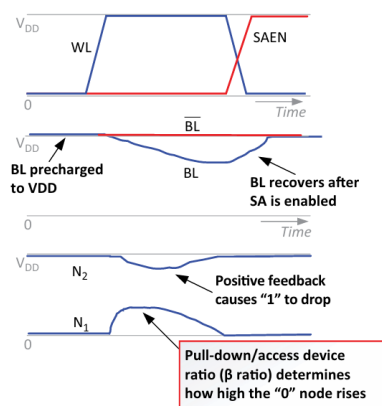


Figure 2.4: SRAM Read Timing Diagram

2.1.5 SRAM Write Operation

During the write operation, the desired data and its complement are applied to the bit-line pair. The word line is then activated, enabling the access transistors and allowing the bit-line voltages to overwrite the previously stored data in the cell.

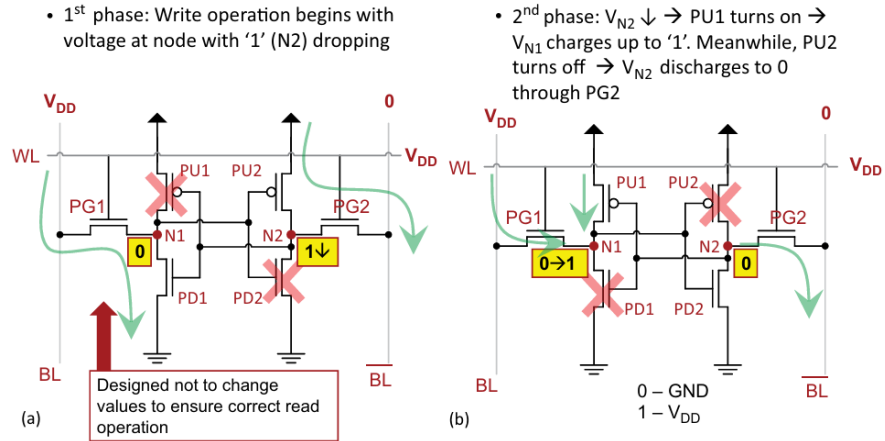


Figure 2.5: SRAM Write Operation

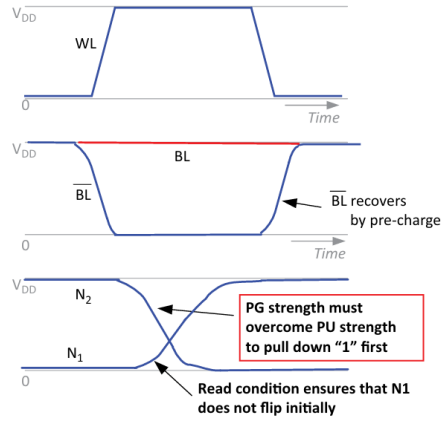


Figure 2.6: SRAM Write Timing Diagram

To successfully write data, the access transistors must be strong enough to overpower the feedback mechanism of the cross-coupled inverters. The write operation generally occurs in two stages:

1. The stored '1' is first discharged to '0'.
2. The opposite node subsequently transitions from '0' to '1'.

2.1.6 Static Noise Margin (SNM)

Static Noise Margin (SNM) is an important parameter used to evaluate the stability and reliability of SRAM cells. It represents the maximum noise voltage that can be tolerated by the SRAM cell without causing an unintended change in the stored data. SNM is typically evaluated using the butterfly curve generated from the voltage transfer characteristics (VTCs) of the cross-coupled inverters.

The major types of SNM are:

- Hold Static Noise Margin (HSNM),
- Read Static Noise Margin (RSNM),
- Write Static Noise Margin (WSNM).

During hold operation, the storage nodes remain isolated from the bit lines, resulting in maximum stability. In read operation, however, the storage node storing logic ‘0’ experiences a slight voltage rise due to the voltage divider effect between the pull-down and access transistors. Consequently, the read SNM is generally lower than the hold SNM.

Similarly, write SNM characterizes the ease with which the stored data can be overwritten during write operations. A proper balance between read stability and write ability is necessary for reliable SRAM design.

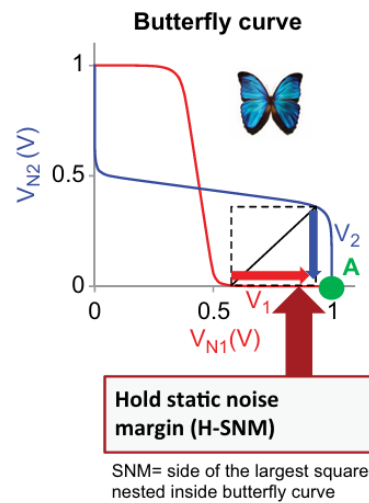


Figure 2.7: Butterfly Curve for SRAM HSNM

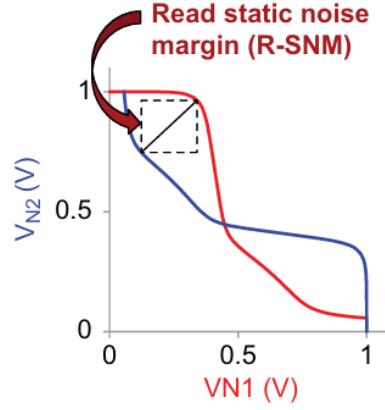


Figure 2.8: Butterfly Curve for SRAM RSNM

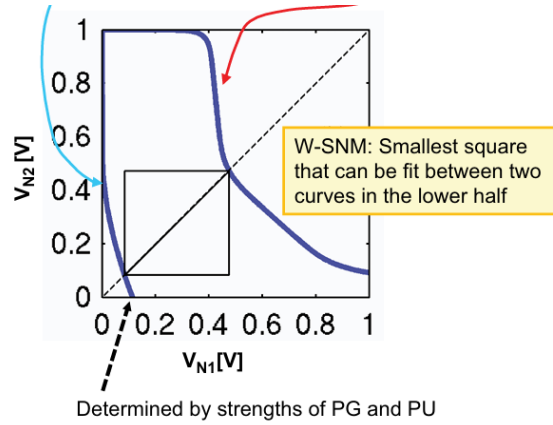


Figure 2.9: Butterfly Curve for SRAM WSNM

2.1.7 Challenges in SRAM Design

As CMOS technology scales into deep submicron regions, SRAM design faces several challenges including reduced noise margins, increased leakage currents, process variations, and lower supply voltages. Lower supply voltages reduce power consumption but also decrease the read and write stability of SRAM cells. Additionally, process variations can cause threshold voltage mismatches, leading to failures during memory operations.

Another major challenge is balancing read stability and write ability. Increasing the strength of pull-down transistors improves read stability, whereas stronger access transistors are preferred for better write performance. Therefore, SRAM design involves careful

transistor sizing and optimization to achieve reliable operation under varying conditions.

2.1.8 Selection of SPI Interface for Biomedical Memory Systems

Implantable pacemaker systems continuously acquire and buffer electrogram (EGM) signals generated from multiple sensing channels. Since these biomedical devices operate using compact batteries under strict power and area constraints, the memory interface used for external SRAM communication must provide low-power operation, reduced routing complexity, and reliable data transfer.

Conventional parallel memory interfaces provide high-speed communication but require a large number of interconnection lines for address, data, and control signals. As memory size increases, parallel interfaces significantly increase routing complexity, chip area, pin count, and switching power consumption. These limitations make parallel communication less suitable for compact implantable biomedical systems where minimizing interconnect overhead and power dissipation is critical.

To overcome these limitations, serial communication protocols are commonly preferred in embedded and low-power systems. Serial interfaces reduce the number of interconnections required between the controller and memory subsystem, thereby simplifying routing and lowering hardware complexity. Common serial communication protocols include Universal Asynchronous Receiver Transmitter (UART), Inter-Integrated Circuit (I²C), and Serial Peripheral Interface (SPI).

UART is an asynchronous communication protocol that provides simple point-to-point communication using separate transmit and receive lines. Although UART requires minimal hardware, it generally supports lower communication speed and is not well suited for high-speed memory interfacing applications.

I²C is a synchronous protocol that supports multiple master and slave devices using only two communication lines. However, I²C introduces additional communication overhead due to address arbitration, acknowledgment cycles, and half-duplex operation, resulting

in lower data transfer efficiency for memory-oriented applications.

In contrast, SPI provides high-speed synchronous full-duplex communication with comparatively lower protocol overhead. SPI communication does not require address arbitration or acknowledgment cycles, thereby reducing latency and improving memory access efficiency. The protocol also provides simpler timing control and efficient hardware implementation suitable for SRAM interfacing applications.

Table 2.1: Comparison of Communication Protocols

Parameter	Parallel	UART	I ² C	SPI
Communication Type	Parallel	Asynchronous	Synchronous	Synchronous
Duplex Mode	Full Duplex	Full Duplex	Half Duplex	Full Duplex
Communication Speed	High	Low	Moderate	High
Pin Count	Very High	Low	Very Low	Low
Routing Complexity	High	Low	Low	Low
Protocol Overhead	Low	Moderate	High	Low
Hardware Complexity	High	Simple	Moderate	Simple
Memory Interfacing	Good	Limited	Moderate	Excellent
Power Efficiency	Low	Moderate	Moderate	High
Suitability for Implantable Systems	Limited	Moderate	Moderate	High

For implantable biomedical systems, reliable and low-latency memory access is important for continuous EGM signal buffering and storage. Compared to UART, I²C, and conventional parallel interfaces, SPI offers reduced pin count, lower routing complexity, higher communication speed, and improved compatibility with external SRAM architectures. Therefore, SPI is selected in this work for interfacing the proposed 64 KB SRAM architecture with the pacemaker controller.

For implantable biomedical systems, reliable and low-latency memory access is important for continuous EGM signal buffering and storage. Compared to UART, I²C, and conventional parallel interfaces, SPI offers reduced pin count, lower routing complexity, higher communication speed, and improved compatibility with external SRAM architectures. Therefore, SPI is selected in this work for interfacing the proposed 64 KB SRAM architecture with the pacemaker controller.

2.2 SPI Communication Protocol

Serial Peripheral Interface (SPI) is a synchronous serial communication protocol widely used for data exchange between digital devices such as microcontrollers, sensors, memory units, and embedded peripherals. SPI provides high-speed communication with simple hardware implementation and is commonly used in System-on-Chip (SoC) and embedded system applications.

SPI follows a master-slave communication architecture in which the master device controls clock generation and communication timing, while the slave devices respond to commands from the master. Data transmission in SPI is synchronized using a serial clock generated by the master device. Since SPI is a full-duplex protocol, data transmission and reception can occur simultaneously between the communicating devices. The SPI interface generally consists of four primary signals:

- Serial Clock (SCLK)
- Master Output Slave Input (MOSI)
- Master Input Slave Output (MISO)
- Chip Select (CS) or Slave Select (SS)

The SCLK signal is generated by the master device and is used for synchronization during data transfer. MOSI carries data from the master to the slave, while MISO transfers data from the slave to the master. The CS signal is used to activate a specific slave device during communication. Compared to parallel communication interfaces, SPI requires fewer interconnection lines and provides simpler hardware implementation with lower routing complexity. These characteristics make SPI suitable for compact embedded systems and low-power biomedical applications.

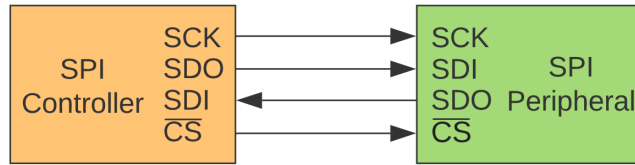


Figure 2.10: Basic SPI communication interface between master and slave devices.

2.2.1 SPI Architecture

A typical SPI communication system consists of one master device connected to one or more slave devices. The master initiates communication by selecting the target slave using the chip select signal and generating the clock pulses required for synchronization.

Data transmission occurs through shift registers present in both master and slave devices. During each clock cycle, one bit of data is shifted from the master to the slave through SDO, while another bit is simultaneously shifted from the slave to the master through SDI. Therefore, SPI communication supports full-duplex data transfer.

In multi-slave configurations, multiple slave devices share the same SCK, SDI, and SDO lines, while separate chip select lines are used to activate individual slave devices.

2.2.2 SPI Modes

SPI communication supports four operating modes based on two timing parameters known as Clock Polarity (CPOL) and Clock Phase (CPHA). These parameters determine the clock idle state and the clock edge used for data sampling.

Clock polarity defines whether the clock remains LOW or HIGH during idle conditions:

- CPOL = 0 : Clock remains LOW during idle state.
- CPOL = 1 : Clock remains HIGH during idle state.

Clock phase determines the clock edge at which data is sampled:

- CPHA = 0 : Data is sampled on the first clock edge.

- $CPHA = 1$: Data is sampled on the second clock edge.

The combination of CPOL and CPHA results in four SPI operating modes, as shown in Table 2.2.

Table 2.2: SPI Communication Modes

SPI Mode	CPOL	CPHA
Mode 0	0	0
Mode 1	0	1
Mode 2	1	0
Mode 3	1	1

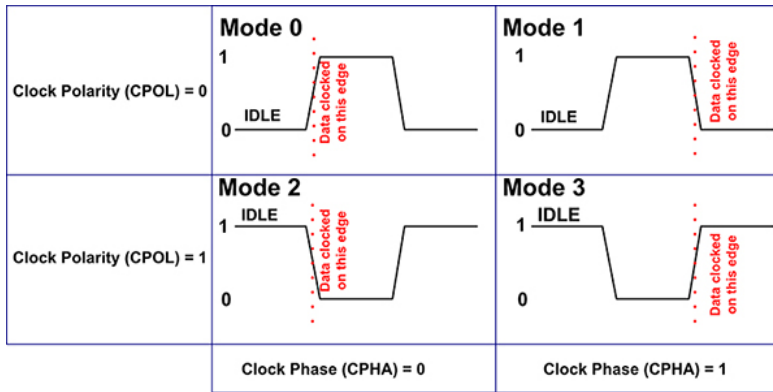


Figure 2.11: Timing diagrams of different SPI modes.

2.2.3 SPI Timing Diagram

The SPI timing diagram illustrates the relationship between clock transitions, data transfer, and chip select activation during communication. Communication begins when the master activates the chip select signal by driving it LOW. Once the slave device is selected, clock pulses are generated through the SCLK signal.

During each clock cycle, data bits are shifted serially through MOSI and MISO lines. Depending on the SPI mode, data is sampled either on the rising edge or falling edge of the clock signal. Proper synchronization between clock edges and data transitions is essential for reliable communication.

At the end of communication, the chip select signal is deactivated by driving it HIGH, thereby terminating the data transfer session.

2.2.4 Advantages of SPI Communication

SPI communication offers several advantages that make it suitable for embedded systems and memory interfacing applications:

- Simple hardware implementation
- High-speed synchronous communication
- Full-duplex data transfer capability
- Reduced pin count compared to parallel interfaces
- Lower routing complexity
- Easy integration with digital controllers and memory systems

Due to these advantages, SPI is widely used in embedded memory architectures, biomedical electronics, sensor interfacing, and low-power System-on-Chip applications.

2.3 Literature Review

Implantable pacemakers have evolved significantly from simple fixed-rate stimulators to highly programmable embedded biomedical systems. Richard S. Sanders and Michael T. Lee [1] explained that modern pacemakers incorporate microprocessors, telemetry, sensors, embedded memory, and lithium-based power sources to improve reliability, longevity, and functionality. Their work highlighted that advancements in hybrid integration, low-power electronics, and memory technologies enabled the miniaturization and sophistication of present-day pacemakers. However, the focus of existing systems remained mainly on pacing functionality, sensing, and device programmability rather than large-scale digital memory architecture.

Larry J. Stotts [2] discussed the role of CMOS integrated circuits in implantable biomedical devices and emphasized the importance of ultra-low-power circuit techniques for extending battery lifetime. The study presented weak inversion MOS operation and low-voltage circuit design methodologies suitable for long-term implantable systems.

It also highlighted the importance of minimizing leakage current and optimizing power consumption in implantable electronics. Although the work addressed low-power circuit design extensively, it primarily focused on analog biomedical circuitry rather than efficient memory buffering architectures for biomedical signal storage.

Sigfredo E. González Díaz [3] proposed a cardiac pacemaker prototype using the MSP430F1611 ultra-low-power microcontroller. The research demonstrated programmable pacing operations, embedded software control, electrogram acquisition, and low-power embedded implementation techniques. The system successfully achieved microampere-level current consumption, proving the effectiveness of embedded low-power architectures for implantable devices. However, the design mainly concentrated on controller implementation and pacing algorithms, with limited emphasis on scalable external memory architectures for continuous electrogram (EGM) signal buffering.

Santosh Chede and Kishore Kulat [4] designed a processor-based implantable pacemaker using an MSP430 ultra-low-power processor. The work implemented pacing functionalities such as pulse generation, refractory period management, and VOO pacing mode while evaluating software-related energy consumption and battery longevity. The authors showed that software optimization and embedded low-power methodologies can significantly reduce energy consumption in implantable pacemakers. Nevertheless, the design focused mainly on pacing control and processor energy optimization, while memory organization and multi-channel biomedical signal storage received limited attention.

The SPL-2 [5] implantable pacemaker ASIC specifications developed by Shree Pacetronix introduced an integrated biomedical ASIC architecture containing pacing blocks, sensing blocks, telemetry circuits, battery monitoring units, and EGM storage support. The specification included programmable pacing channels, sensing amplifiers, analog filtering, telemetry communication, and memory interfacing features. Although the architecture supported EGM signal acquisition and storage, it primarily emphasized analog front-end design and controller interfacing rather than efficient large-scale

memory buffering mechanisms optimized for multiple sensing channels.

The SPL-2 [6] requirements document further described detailed electrical and functional requirements for implantable pacemaker ASICs, including sensing circuits, pacing pulse generators, watchdog systems, telemetry communication, and EGM storage circuits. The document highlighted important implantable device constraints such as ultra-low power consumption, EMI/ESD protection, leakage current minimization, and high reliability. While the design supports intracardiac electrogram storage, it does not provide an optimized memory architecture for continuous multi-channel EGM buffering with reduced communication overhead.

Wong et al. [7] proposed a leakage current cancellation technique for switched-capacitor circuits intended for low-power implantable medical devices. The study introduced an active feedback mechanism to minimize leakage currents and thereby reduce power dissipation in battery-operated biomedical systems. This work is highly relevant for implantable pacemaker applications because leakage reduction directly contributes to longer battery life and improved reliability in continuously operating biomedical electronics.

Prasant Kumar Swain et al.[8] designed a compact SPI-based SRAM using 0.6 μm CMOS technology for low-power IoT applications. The SRAM used the Serial Peripheral Interface (SPI) protocol for read, write, and control operations, enabling compact design and reduced power consumption. The architecture demonstrated efficient serial communication-based memory access and low-power SRAM operation. However, the proposed SRAM system was intended mainly for IoT sensor applications and was not optimized for implantable biomedical systems requiring continuous multi-channel EGM signal buffering, ultra-high reliability, and strict power constraints.

Robert Balas et al. [9] proposed the CV32RT RISC-V architecture with fast interrupt handling and efficient context switching mechanisms for real-time applications. The work demonstrated the importance of minimizing latency and improving deterministic behavior in embedded systems. Such real-time processing concepts are

important for implantable pacemakers where timely acquisition and storage of cardiac events are critical.

2.4 Existing SRAM Architectures

The conventional 6T SRAM cell is widely used due to its compact structure and high integration density. However, as CMOS technology scales toward lower supply voltages, the 6T SRAM cell faces challenges such as reduced read stability, poor write ability, increased leakage current, and higher sensitivity to process variations. During read operation, the internal storage nodes may get disturbed because the same access transistors are used for both read and write operations.

To overcome these limitations, advanced SRAM architectures such as 8T, 9T, and 10T SRAM cells have been developed. These architectures introduce additional transistors to separate read and write paths, thereby improving read stability and reducing disturbance during memory access. Separate read ports also help improve Static Noise Margin (SNM) under low-voltage operation.

Some advanced SRAM cells additionally employ leakage reduction techniques, feedback control mechanisms, and transistor stacking methods to minimize standby power consumption. Although these architectures improve reliability and low-power operation, they generally increase area overhead and circuit complexity compared to conventional 6T SRAM cells.

Therefore, selecting an appropriate SRAM architecture involves balancing stability, power consumption, area, and performance requirements depending on the target application. In this work, a conventional 6T SRAM architecture is utilized due to its simpler design, compact area, easier integration, and suitability for implementing the proposed SPI-based memory system.

2.4.1 Research Gap

Existing implantable pacemaker systems mainly focus on pacing control, sensing circuitry, telemetry communication, and analog

front-end design. Most current systems utilize event-based storage methods with limited on-chip memory support, while efficient large-scale digital memory architectures for continuous biomedical signal buffering are generally not emphasized. Multi-channel electrogram (EGM) signals generated from different sensing channels require reliable temporary buffering and storage, but current systems are not optimized for efficient multi-channel EGM data management.

In many existing pacemaker architectures, external memory interfacing is performed using serial communication protocols such as SPI. Although SPI-based memory systems provide compact implementation and reduced pin count, they introduce additional latency, communication overhead, and power consumption compared to direct on-chip memory access. Furthermore, most SPI-based SRAM architectures reported in literature are primarily targeted toward IoT and embedded sensor applications rather than implantable biomedical systems operating under strict ultra-low-power and reliability constraints.

From the literature survey, it is observed that there is a lack of a low-power external SRAM architecture specifically optimized for implantable pacemaker systems. Existing designs do not efficiently support buffering of EGM signals from multiple sensing channels while satisfying important implantable device requirements such as low power consumption, limited chip area, reduced interface overhead, and high reliability.

Therefore, the proposed work focuses on the design of a low-power SPI-based SRAM system for implantable pacemaker applications. The proposed architecture interfaces a 64 KB SRAM memory with the pacemaker controller to support efficient event-driven multi-channel EGM signal buffering while minimizing power consumption and communication overhead.

Chapter 3

Proposed SPI Enabled SRAM Architecture

3.1 Proposed 64 KB SRAM Architecture

The proposed memory subsystem consists of a 64 KB SRAM architecture constructed using eight 8192×8 Single Port SRAM (SPRAM) macros. The architecture is designed to support low-power operation, scalable memory organization, and efficient memory access suitable for implantable biomedical applications.

Instead of implementing a large monolithic memory block, the complete memory is organized using smaller SRAM macros connected through address decoding and output multiplexing circuitry. This banking-based organization improves modularity, simplifies physical implementation, and reduces switching activity during memory access operations.

The overall organization of the proposed 64 KB SPRAM architecture is shown in Figure 3.1. The architecture consists of:

- Eight 8192×8 SRAM macros
- Address decode and control logic
- 8-to-1 output data multiplexer
- Shared address and data buses

The architecture receives a 16-bit address input $A[15:0]$, an 8-bit input data bus $I[7:0]$, and control signals CE , WEB , OEB , and CSB . The address bus is divided into two parts:

- $A[15:13]$: Bank selection address
- $A[12:0]$: Local address inside SRAM macro

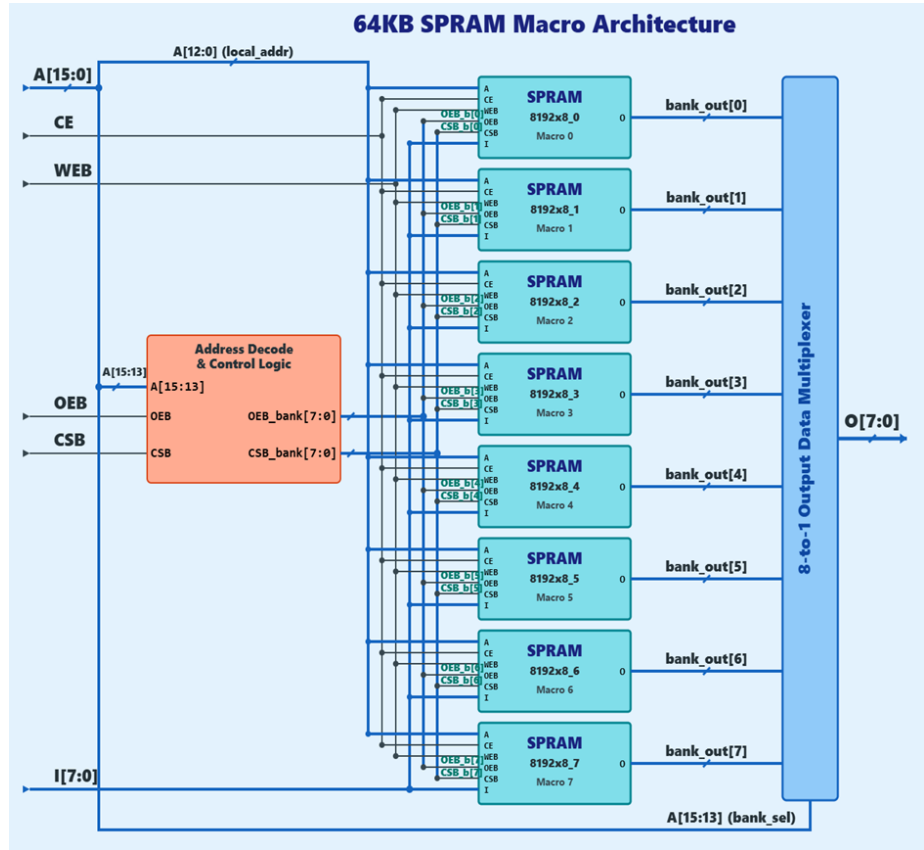


Figure 3.1: Proposed 64 KB SPRAM architecture using eight 8192×8 SRAM macros.

The higher-order address bits $A[15:13]$ are connected to the address decode and control logic block. This block selects one of the eight SRAM macros depending on the applied address. Only the selected SRAM bank becomes active during read or write operation, while the remaining memory banks remain disabled. This selective bank activation reduces unnecessary switching activity and improves power efficiency.

The lower-order address bits A[12:0] are connected simultaneously to all SRAM macros and are used to select the memory location inside the selected SRAM bank. Since each SRAM macro contains 8192 memory locations, 13 address bits are required for internal memory addressing.

The major features of the SRAM macro are summarized below:

- Memory Type : Single Port SRAM
- Capacity : 8192×8 bits (8 KB)
- Technology : TSL18S100 (180 nm)
- Synchronous read and write operation
- Tri-state output bus
- Clock-based memory operation
- Low leakage memory architecture
- Width = 526.84 μm , Height = 699.51 μm
- Macro Area = 0.368 mm^2
- Maximum Frequency = 263 MHz

The outputs from all SRAM banks are connected to an 8-to-1 output multiplexer. Depending on the selected bank address A[15:13], the corresponding SRAM output is routed to the final output bus O[7:0].

3.1.1 SRAM Read Operation

During read operation, the write enable signal WEB remains HIGH to indicate read mode operation. The chip select signal CSB is asserted LOW to activate the memory system, while the output enable signal OEB is also asserted LOW to enable output data transfer.

The higher-order address bits A[15:13] select the required SRAM bank through the address decode and control logic block. Simultaneously, the lower-order address bits A[12:0] select the required memory location inside the selected SRAM macro.

At the active clock edge, the address and control signals are internally latched by the SRAM macro. After the specified access time delay, the stored data appears at the corresponding bank output. The 8-to-1 output multiplexer forwards the selected bank output to the final output bus O[7:0].

Only the selected SRAM macro drives the output bus during read operation, while all remaining SRAM banks remain disabled in tri-state condition. This prevents bus contention and reduces unnecessary power dissipation. The synchronous read timing waveform of the SRAM macro is shown in Figure 3.2.

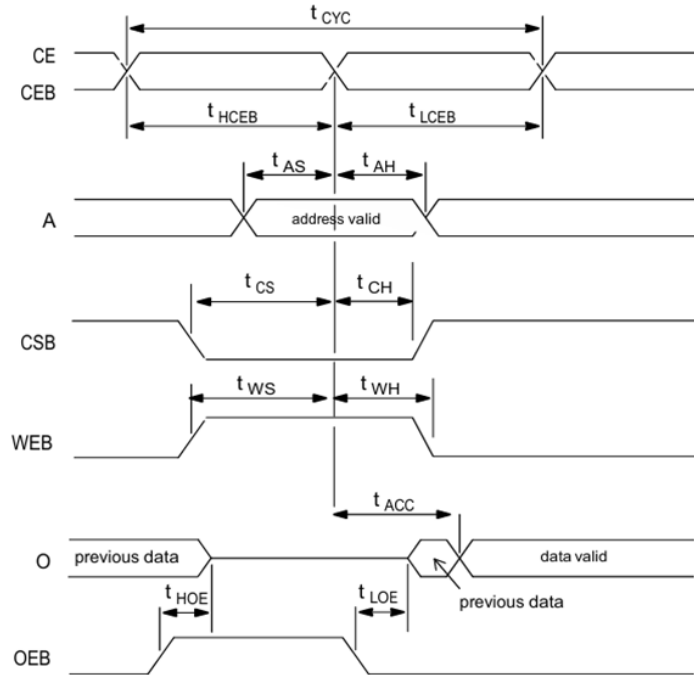


Figure 3.2: Read cycle timing diagram of the SRAM macro.

3.1.2 SRAM Write Operation

During write operation, the selected SRAM macro stores input data into the specified memory location. To perform write operation, the write enable signal WEB and chip select signal CSB are asserted LOW while the desired input data is applied to the input data bus I[7:0].

The higher-order address bits A[15:13] activate the required SRAM bank through the address decode and control logic block, while the lower-order address bits A[12:0] select the target memory location inside the selected SRAM macro.

At the active clock edge, the address, control signals, and input data are internally latched by the SRAM macro, and the input data is written into the selected memory location. During write operation, the output enable signal OEB generally remains HIGH to disable output data transfer and avoid bus conflicts.

Only the selected SRAM bank performs the write operation, while all remaining memory banks remain disabled. This selective bank activation minimizes unnecessary switching activity and improves overall memory power efficiency. The write timing waveform of the SRAM macro is shown in Figure 3.3.

The proposed banking-based SRAM organization provides improved modularity, simplified memory expansion, reduced switching activity, lower routing complexity, and efficient memory access suitable for low-power implantable biomedical systems.

3.1.3 SRAM Hold Operation

During hold operation, the SRAM macro retains the previously stored data without performing any read or write activity. In this mode, the chip select signal CSB remains HIGH, thereby disabling the memory bank and isolating it from external operations. The write enable signal WEB remains HIGH, while no switching activity occurs on the data bus. Since only the internal cross-coupled storage nodes maintain the stored data, the hold mode minimizes dynamic power consumption and supports low-power standby operation suitable for

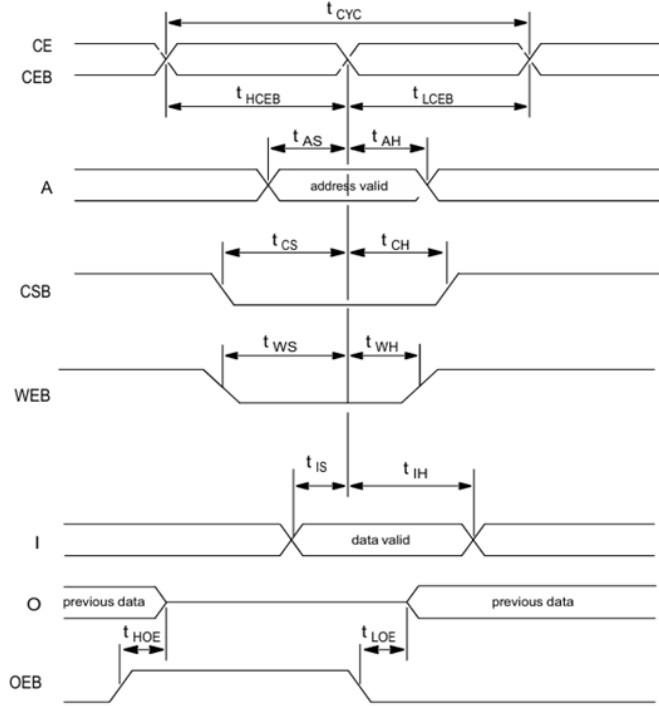


Figure 3.3: Write cycle timing diagram of the SRAM macro.

implantable biomedical applications.

3.2 SPI Interface Design

The proposed memory architecture employs a Serial Peripheral Interface (SPI) for communication between the controller and the external SRAM subsystem. SPI communication is selected because of its simple hardware implementation, synchronous operation, reduced routing complexity, and lower pin requirement compared to parallel communication interfaces. The SPI interface enables efficient serial communication between the controller and SRAM memory while minimizing hardware overhead and interface complexity.

The SPI interface mainly consists of serial communication logic, synchronous counters, decoder circuitry, SRAM control logic, read/write registers, and SRAM memory blocks. The SPI-based memory interface architecture used in the proposed system is shown in Figure 3.5.

The SPI interface receives serial input data through the *sdo* line

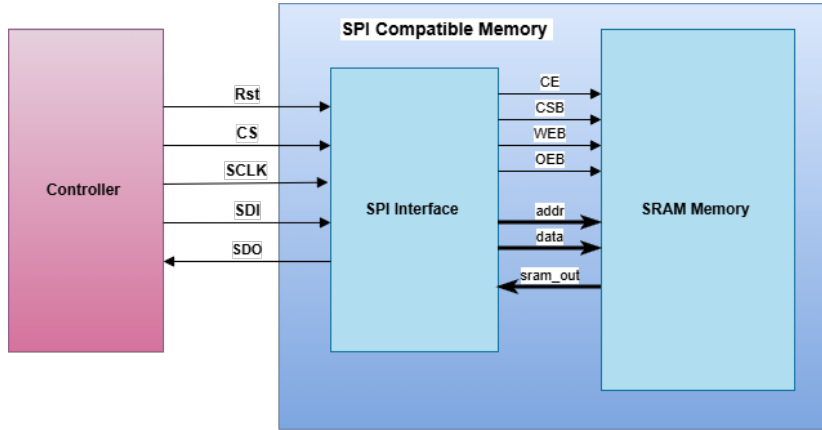


Figure 3.4: SPI interface connected with SRAM memory

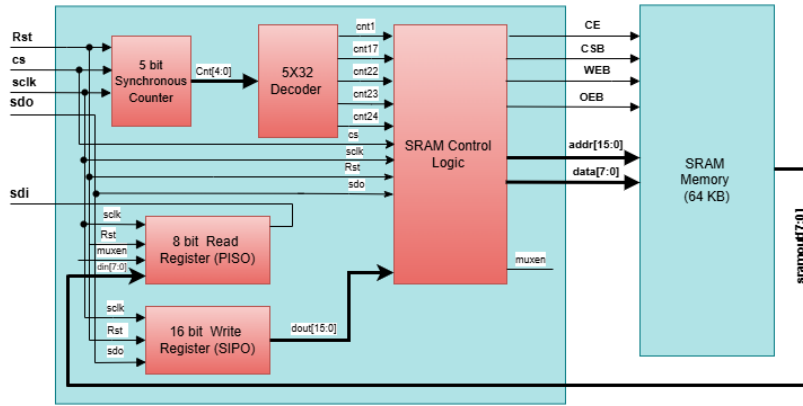


Figure 3.5: SPI interface internal architecture

along with synchronization signals such as *sclk*, *cs*, and *Rst*. The synchronous counter increments with every clock cycle and generates counter outputs required for sequencing SPI communication and memory operations. The counter outputs are applied to the 4×16 decoder, which activates different control signals during various stages of SPI communication.

The SRAM control logic block acts as the central controller of the interface. It receives inputs from the decoder, SPI communication signals, and read/write registers to generate memory control signals such as Clock Enable signal (CE), chip select signal CSB, WEB, *addr*[15:0], *data*[7:0] and *muxen* signal. The SPI interface contains two major registers are 8-bit Write Register (SIPO) and 8-bit Read Register (PISO). The Serial-In Parallel-Out (SIPO) register receives serial input data from the SPI interface and converts it into parallel

data before transferring it to the SRAM memory during write operation. Similarly, the Parallel-In Serial-Out (PISO) register converts SRAM output data into serial format for transmission back through the SPI interface during read operation.

The SRAM memory block stores the received data and performs read/write operation based on the control signals generated by the SRAM control logic block. The synchronous operation of the complete architecture ensures reliable data transfer between the SPI controller and SRAM subsystem.

The proposed SPI architecture operates in SPI Mode-3 configuration, where:

- Clock Polarity (CPOL) = 1
- Clock Phase (CPHA) = 1

In SPI Mode-3 operation, the serial clock remains HIGH during idle condition, and data sampling occurs on the rising edge of the clock signal. This mode provides reliable synchronization between serial data transfer and clock transitions.

3.2.1 SPI Finite State Machine (FSM) Operation

The operation of the SPI controller is managed using a Finite State Machine (FSM) which controls serial communication, address transfer, data transfer, and memory access sequencing. The FSM ensures proper synchronization between SPI communication and SRAM read/write operations.

The FSM used in the proposed SPI memory architecture is shown in Figure 3.8.

Initially, the controller remains in the *Idle* state waiting for SPI communication to begin. Once the chip select signal becomes active, the FSM moves to the *Read/Write* state where the operation type is identified.

For write operation, the FSM first enters the *W_Addr* state to receive the target memory address serially from the SPI interface. After address reception, the controller moves to the *W_Data* state where input data is serially received and stored into the write register.

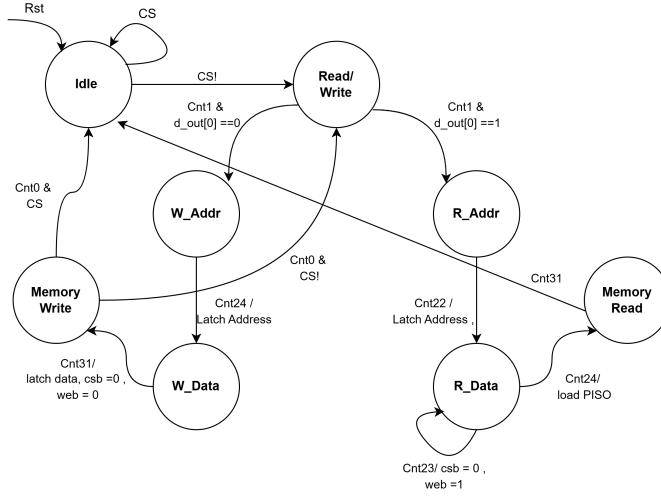


Figure 3.6: Finite State Machine (FSM) of the SPI memory controller

The FSM then enters the *Memory Write* state, where the SRAM control logic generates the required control signals to store data into the selected SRAM location.

For read operation, the FSM enters the *R_Addr* state to receive the memory address from the SPI interface. The controller then moves to the *Memory Read* state where the selected SRAM location is accessed. The retrieved data is loaded into the read register, and the FSM enters the *R_Data* state to serially transmit the output data back through the SPI interface.

The FSM-based operation provides synchronized memory access, simplified control sequencing, and reliable serial communication between the SPI controller and SRAM subsystem.

3.2.2 SPI Timing Operation

SPI communication is fully synchronized using the serial clock generated by the SPI master. The chip select signal activates the communication session, while serial data transfer occurs through the SPI communication lines.

During write operation, serial input data is shifted into the SIPO register at every clock cycle. After complete address and

data reception, the SRAM control logic generates the required write control signals for storing data into the selected SRAM memory location.

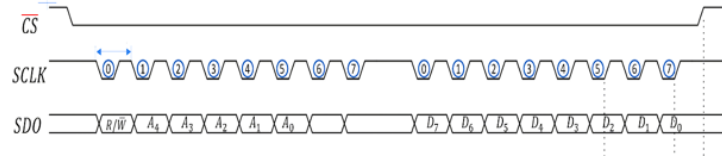


Figure 3.7: Write Timing Diagram

During read operation, the selected SRAM memory location is accessed, and the output data is loaded into the PISO register. The PISO register then serially transmits the output data back through the SPI interface using clock synchronization.

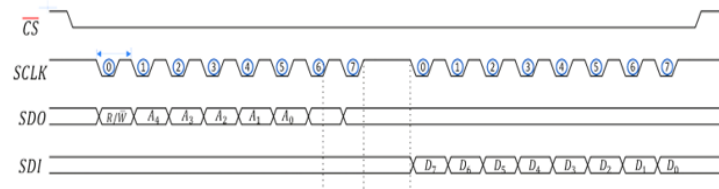


Figure 3.8: Read Timing Diagram

Proper synchronization between clock edges and serial data transfer ensures reliable communication and accurate memory access operation.

3.3 Integration of SPI with SRAM

The proposed architecture integrates the SPI communication interface with the SRAM subsystem to provide efficient serial memory access suitable for low-power embedded biomedical applications. The SPI controller acts as an interface between the external controller and SRAM memory system.

During write operation, serial input data received through the SPI interface is converted into parallel format using the SIPO write register. The SRAM control logic generates the required address, write enable, and control signals for storing the received data into the selected SRAM memory location.

During read operation, the SRAM subsystem retrieves data from the selected memory location and transfers it to the PISO read register. The read register converts the parallel SRAM output into serial format and transmits it back through the SPI communication interface.

The SRAM control logic coordinates communication between the SPI interface and SRAM subsystem by controlling address generation, read/write sequencing, register operation, and memory access timing. The integrated architecture ensures synchronized data transfer between serial communication logic and parallel memory organization.

The integrated SPI-SRAM architecture provides several advantages including:

- Reduced pin count
- Simplified routing complexity
- Efficient serial memory access
- Reduced hardware overhead
- Scalable memory organization
- Low-power operation

3.4 Overall System Operation

The overall operation of the proposed SPI-enabled SRAM architecture involves coordinated communication between the controller, SPI interface, control logic, and SRAM subsystem. The architecture is designed to support efficient serial memory access with reliable data transfer and low-power operation suitable for implantable biomedical applications.

The complete system operation begins when the external controller initiates SPI communication by activating the chip select signal. Once the SPI communication session starts, serial clock

pulses generated by the SPI master synchronize the transfer of address and data information through the SPI interface.

Depending on the received command, the system performs either read operation or write operation. During write operation, serial input data received through the SPI interface is shifted into the Serial-In Parallel-Out (SIPO) register. Simultaneously, the received address is decoded to identify the required SRAM bank and local memory location.

The SRAM control logic generates the necessary control signals for activating the selected SRAM bank and performing write operation. The received parallel data is then stored into the specified SRAM memory location.

During read operation, the controller first transmits the target memory address through the SPI interface. The address decode logic selects the required SRAM bank, while the local address bits select the corresponding memory location inside the SRAM macro.

The SRAM subsystem retrieves the stored data from the selected memory location and transfers it to the Parallel-In Serial-Out (PISO) register. The PISO register serializes the output data and transmits it back through the SPI communication interface.

The proposed architecture supports synchronized memory access, efficient serial communication, and scalable memory organization while minimizing switching activity and communication overhead. The use of memory banking and SPI communication enables reliable event-driven data buffering suitable for implantable biomedical systems.

Chapter 4

RTL to GDSII Implementation Flow

4.1 ASIC Design Methodology for SPI-Enabled SRAM

In the proposed work, the basic 8192×8 SRAM macro was already available as a pre-designed and verified memory block. To integrate eight SRAM macros along with SPI communication logic, address decoding circuitry, bank selection logic, multiplexers, counters, FSM-based control logic, and peripheral digital blocks to realize a complete 64 KB SPI-enabled SRAM architecture.

Implementing the complete system directly using full-custom CMOS transistor-level design would significantly increase design complexity and verification effort. Since the architecture contains large digital control logic and multiple interconnected SRAM macros, manual transistor-level implementation would make timing closure, clock distribution, routing optimization, and large-scale layout integration extremely difficult. In addition, full-custom integration of multiple memory macros may introduce large numbers of DRC, LVS, connectivity, and routing violations, thereby increasing overall design time and physical verification complexity.

The RTL-to-GDSII design methodology provides a more efficient approach for implementing such large digital systems. Using

RTL-based design, the complete architecture can first be modeled behaviorally using Verilog HDL and then automatically synthesized, optimized, placed, and routed using standard ASIC design tools. The methodology also simplifies integration of pre-designed SRAM macros with standard-cell-based digital logic while enabling automated timing optimization, clock tree synthesis, congestion reduction, and physical verification.

Therefore, the RTL-to-GDSII flow is adopted in this work to achieve reliable large-scale SRAM integration, simplified physical implementation, improved timing closure, and manufacturable layout generation for the proposed SPI-enabled 64 KB SRAM architecture.

4.2 RTL to GDSII Implementation Flow

The RTL to GDSII flow represents the complete ASIC physical design methodology used to transform a behavioral hardware description into a manufacturable integrated circuit layout. The flow begins with Register Transfer Level (RTL) design and functional verification, followed by logic synthesis, physical implementation, timing verification, and final tapeout generation in GDSII format.

In this work, the complete RTL to GDSII implementation of the proposed SPI-enabled SRAM architecture is carried out using Cadence EDA tools in 180 nm CMOS technology. The flow ensures that the design satisfies functional correctness, timing constraints, power requirements, and physical manufacturability conditions before fabrication.

The RTL to GDSII implementation flow followed in this work is illustrated in Figure 4.1.

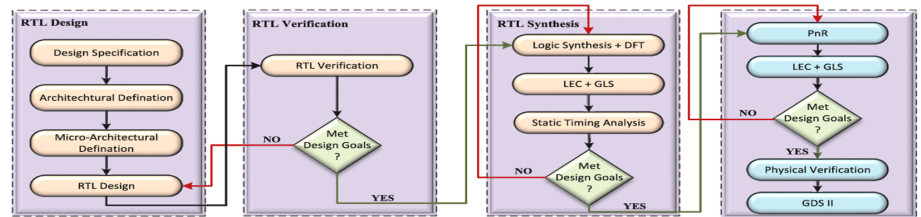


Figure 4.1: Complete RTL to GDSII implementation flow.

The overall design flow consists of four major stages:

1. RTL Design
2. RTL Verification
3. RTL Synthesis
4. Physical Design (PnR to GDSII)

Each stage contains multiple verification and optimization steps to ensure that the final chip satisfies the required design specifications. If any stage fails to meet the design goals, the flow iterates back to the previous stage for necessary modifications and optimizations.

4.2.1 RTL Design

RTL design is the first stage of ASIC implementation where the proposed architecture is described using synthesizable Verilog HDL. The RTL defines data flow, control logic, memory operations, and communication protocols of the system.

As shown in Figure 4.1, the RTL design stage consists of the following steps:

1. Design Specification
2. Architectural Definition
3. Micro-Architectural Definition
4. RTL Design

4.2.2 Design Specification

The design specification defines the functional and performance requirements of the proposed system including memory organization, SPI communication, read/write operation, clock frequency, power constraints, and IO requirements.

For the proposed work, the specifications include SPI-based communication with SRAM memory blocks, serial data transfer, and memory access control operations.

4.2.3 Architectural Definition

The architectural definition stage determines the high-level organization of the system. The architecture consists of SPI controller, SRAM banks, decoder, FSM controller, read/write logic, and SIPO/PISO registers. The interaction between these modules is finalized during this stage.

4.2.4 Micro-Architectural Definition

The micro-architectural stage defines the internal implementation of each module including FSM transitions, control signals, datapath operations, timing synchronization, and data transfer sequence.

4.2.4.1 RTL Design

After finalizing the micro-architecture, synthesizable Verilog RTL code is developed for all modules. Each functional block is coded individually and later integrated into the top-level design.

The RTL implementation includes SPI protocol implementation, SRAM read/write logic, Serial-to-parallel conversion and viceversa. The RTL code forms the input for functional verification and synthesis stages.

4.3 RTL Verification

RTL Verification is performed after completion of RTL design to validate the functional correctness of the proposed architecture before proceeding to synthesis and physical implementation.

If the design fails to satisfy the required functional goals during verification, the design is modified at the RTL stage and verified again before proceeding further in the implementation flow.

The RTL verification stage mainly focuses on validating:

- Functional correctness of the design
- FSM state transitions

- Data transfer operations
- Timing synchronization

Successful RTL verification significantly reduces implementation errors during later stages of ASIC design flow.

4.4 Advanced RTL Verification Checks

In digital ASIC design flow, advanced RTL verification checks such as Lint Checking, CDC/RDC analysis, and X-Propagation verification are performed to improve design reliability and detect errors at early stages.

4.4.1 Lint Checking

Lint checking is a static RTL verification technique used to detect coding errors, synthesis issues, and violations of coding guidelines without simulation.

The major objectives of lint checking are:

- Detect syntax and coding issues
- Identify unused or uninitialized signals
- Detect inferred latches and combinational loops
- Verify FSM and reset coding styles
- Detect width mismatches and multiple drivers

Lint checking improves RTL quality by detecting coding errors, synthesis mismatches, inferred latches, and reset-related issues at an early stage.

4.4.2 CDC and RDC Verification

Clock Domain Crossing (CDC) and Reset Domain Crossing (RDC) verification are performed in designs containing multiple clocks and reset domains.

CDC verification checks signal transfers between different clock domains, while RDC verification analyzes reset signal crossings between reset domains.

The objectives of CDC/RDC verification are:

- Detect metastability issues
- Verify synchronization circuits
- Validate safe clock and reset crossings
- Detect unsafe asynchronous transfers

Common synchronization methods include Two-flop synchronizers, Asynchronous FIFOs, Handshake synchronizer, Gray-code counters.

CDC/RDC verification prevents metastability issues and improves reliability in multi-clock and multi-reset designs.

4.4.3 X-Propagation Verification

X-Propagation verification analyzes unknown logic states caused by uninitialized registers, incomplete resets, or floating signals. It helps prevent simulation mismatches and improves FSM reliability.

4.5 Logic Synthesis

Logic synthesis converts the RTL design into an optimized gate-level netlist using technology-specific standard cell libraries. The synthesis flow reads RTL files, timing constraints, and libraries to generate a timing, power, and area optimized implementation.

Logic synthesis requires several input files and design constraints for successful implementation. The major inputs used during synthesis are RTL Verilog source files, Constraint files (SDC), TCL synthesis scripts, Standard cell libraries, IO pad libraries.

The RTL source files contain the behavioral description of the complete architecture. Constraint files define clock definitions, input

and output delays, clock uncertainty, false paths, timing exceptions, load and transition constraints.

Technology library files provide timing, area, and power information of the standard cells used during technology mapping. The synthesis flow also includes IO pad instantiation before gate-level implementation.

The generated gate-level Verilog netlist represents the synthesized hardware implementation using technology-specific standard cells. The generated SDC files preserve timing constraints for later physical design stages.

The synthesis flow also generates Standard Delay Format (SDF) files containing timing delay information required for gate-level timing simulation.

4.5.1 Synthesis Reports

Synthesis reports provide timing, area, power, and constraint information of the synthesized design. Timing reports contain critical paths and setup/hold slack. Area reports provide cell utilization statistics, while power reports estimate dynamic and leakage power consumption. These reports are analyzed before physical implementation.

4.6 Logic Equivalence Checking (LEC)

Logic Equivalence Checking (LEC) is performed to verify that the synthesized gate-level netlist is functionally equivalent to the original RTL design. During synthesis, optimization and technology mapping modify the internal implementation of the circuit. Therefore, LEC is required to ensure that these optimizations do not change the intended functionality of the design.

In this work, post-synthesis LEC is performed using Cadence Conformal LEC tool. The RTL design is considered as the golden design, while the synthesized gate-level netlist is treated as the revised design.

The Conformal LEC tool compares:

- Primary inputs and outputs
- Combinational logic and Sequential elements
- State transitions and Internal connectivity

Successful LEC verification confirms that the synthesized implementation is logically correct and functionally identical to the RTL design before proceeding to physical design stages.

4.7 Post Synthesis Gate Level Simulation (GLS)

Gate Level Simulation (GLS) verifies the functionality of the synthesized gate-level netlist using standard cell timing information.

GLS is performed in three stages:

- Zero Delay Simulation
- Unit Delay Simulation
- SDF Simulation

Zero delay simulation verifies logical functionality without delays. Unit delay simulation introduces equal delay to all gates to identify timing-related issues. SDF simulation uses actual back-annotated delays for accurate post-layout timing verification.

GLS helps detect timing violations, glitches, synchronization issues, and RTL-to-netlist mismatches before fabrication.

4.8 Floorplanning

Floorplanning defines the physical structure of the chip including die size, core area, IO placement, and macro locations before placement and routing. Proper floorplanning improves timing, routability, congestion, and power distribution.

The major objectives of floorplanning are:

- Define chip dimensions and core area

- Place IO pads and power pads
- Plan macro locations
- Minimize routing congestion
- Improve timing performance
- Support efficient power distribution

4.8.1 Macro Planning

Macro planning determines the placement of SRAM macros to minimize wirelength, congestion, and timing violations. Proper spacing and orientation are maintained for routing resources and power distribution.

Since SRAM macros occupy a large portion of chip area, macro planning strongly influences placement quality, clock tree synthesis, routing complexity, and overall chip performance.

4.8.2 MMMC Configuration

Modern ASIC implementation requires verification under multiple operating conditions. Multi-Mode Multi-Corner (MMMC) analysis is configured to verify timing under different process, voltage, and temperature conditions using worst-case and best-case timing libraries.

Thus, floorplanning establishes the initial physical framework of the ASIC design and significantly impacts the overall implementation quality, timing closure, power integrity, and routing efficiency of the final chip.

4.9 Power Planning

Power planning establishes a reliable power distribution network for standard cells, macros, and IO cells while minimizing IR-drop and electromigration effects.

The major objectives of power planning are:

- Provide stable power and ground connections

- Reduce IR-drop and electromigration
- Support high current flow
- Improve power integrity

4.9.1 Power Distribution Network

The power distribution network consists of Power and ground pads, Core power rings, Macro power rings and Power stripes.

Core rings distribute power across the chip, while macro rings provide stable supply to SRAM macros. Power stripes connect the rings to internal standard cells and memory blocks. Dedicated routing layers are selected for power rings and stripes to improve current carrying capability and reduce voltage drop.

4.10 Placement

Placement is an important stage in physical design flow in which all standard cells are positioned inside the predefined floorplan region after completion of floorplanning and power planning. The primary objective of placement is to arrange the standard cells in a manner that minimizes routing congestion, reduces wirelength, and satisfies timing constraints.

The quality of placement strongly affects:

- Timing closure and Routing congestion
- Power consumption
- Signal integrity and Clock tree synthesis
- Overall chip area utilization

Improper placement may result in excessive routing congestion, large interconnect delays, timing violations, and difficulty during routing stages. Therefore, placement acts as a critical optimization stage in the ASIC implementation flow.

The placed database generated after placement is used as the input for Clock Tree Synthesis (CTS) and routing stages.

4.11 Clock Tree Synthesis (CTS)

Clock Tree Synthesis (CTS) constructs the clock distribution network to deliver the clock signal uniformly across all sequential elements.

The major objectives of CTS are:

- Minimize clock skew
- Control clock latency
- Balance clock arrival time
- Improve timing closure

During CTS, clock buffers and inverters are inserted to distribute the clock efficiently throughout the design. After CTS, the clock changes from an ideal clock to a propagated clock with actual routing delays.

4.12 Routing

Routing creates physical metal interconnections between standard cells, SRAM macros, IO pads, and clock networks while satisfying timing and design rule constraints.

Routing affects:

- Timing performance
- Signal integrity
- Routing congestion
- Manufacturability

Routing is generally divided into:

- Global Routing
- Detailed Routing

Global routing estimates routing paths and congestion, while detailed routing generates exact metal and via connections.

Post-route optimization is performed to fix setup/hold violations, antenna violations, and signal integrity issues before signoff.

4.13 Post-Route Logic Equivalence Checking

Post-Route Logic Equivalence Checking (LEC) is performed after placement and routing stages to verify that the final routed netlist remains functionally equivalent to the synthesized netlist or RTL design. During physical implementation, several optimizations are performed such as buffering, clock tree insertion, routing optimization, antenna fixing, and ECO modifications.

Therefore, post-route LEC is necessary to ensure that these physical modifications do not alter the logical functionality of the design. Without post-route LEC, undetected functional mismatches may propagate into fabrication, resulting in chip failure.

Successful completion of post-route LEC ensures that the final routed design is functionally correct and ready for final signoff stages such as DRC, LVS, and GDSII generation.

4.14 Post-Route Gate Level Simulation (GLS)

Post-route GLS verifies the functionality of the routed gate-level netlist using actual routing and clock delays extracted after physical implementation.

The simulation is performed using:

- Zero Delay Simulation
- Unit Delay Simulation
- SDF Simulation

SDF simulation uses back-annotated routing delays to validate timing-correct operation under realistic post-layout conditions.

Post-route GLS helps identify setup/hold violations, glitches, race conditions, and clock synchronization problems before tapeout.

4.15 Signoff Verification

Signoff verification is the final stage of ASIC physical design flow performed before tapeout and GDSII generation. It ensures that the routed design satisfies timing, connectivity, and manufacturability requirements before fabrication.

The major objectives of signoff verification are:

- Verify timing closure
- Validate routing connectivity
- Check physical design correctness
- Ensure DRC-clean layout

4.15.1 Physical Signoff

Physical signoff includes RC extraction, connectivity verification and DRC verification

RC extraction generates parasitic information required for accurate timing analysis, while DRC and connectivity checks ensure layout correctness and manufacturability.

4.15.2 Timing Signoff

Timing signoff is performed using Static Timing Analysis (STA) with post-route netlist, SDC constraints, SPEF parasitic files, and timing libraries.

The analysis verifies:

- Setup and Hold timing
- Clock timing and signal integrity effects

Timing analysis is performed across multiple process, voltage, and temperature corners to ensure reliable chip operation. The generated GDSII file is used for tapeout and fabrication.

4.16 Physical Verification

Physical verification is performed after routing to ensure that the final layout is manufacturable and electrically correct before tapeout. The major verification stages include:

4.16.1 Design Rule Check (DRC)

Design Rule Check (DRC) verifies whether the layout satisfies the foundry manufacturing rules. The major DRC checks include:

- Minimum metal width and spacing
- Via enclosure rules
- Minimum area violations
- Routing geometry correctness

In this work, DRC verification was performed using Calibre integrated with Cadence Virtuoso.

4.16.2 Antenna Checks

Antenna checks identify charge accumulation on long metal wires during fabrication which may damage transistor gate oxide layers. These checks help prevent:

- Gate oxide damage
- Antenna rule violations
- Fabrication reliability issues

4.16.3 Layout Versus Schematic (LVS)

Layout Versus Schematic (LVS) verification checks whether the final layout matches the intended circuit netlist.

LVS verifies:

- Device and net connectivity
- Pin matching
- Power and ground connections
- Missing or extra devices

In this work, LVS verification was performed using Calibre nmLVS integrated with Cadence Virtuoso. After successful DRC, LVS, and antenna verification, the design is considered ready for fabrication.

Chapter 5

Results and Performance Analysis

This chapter presents the functional verification, synthesis, physical implementation, and performance analysis results of the proposed SPI-enabled 64 KB SRAM architecture implemented using 180 nm CMOS technology. The results obtained at different stages of the RTL-to-GDSII flow are discussed and analyzed in detail.

5.1 64KB SRAM Results

5.1.1 SRAM Macro Verification

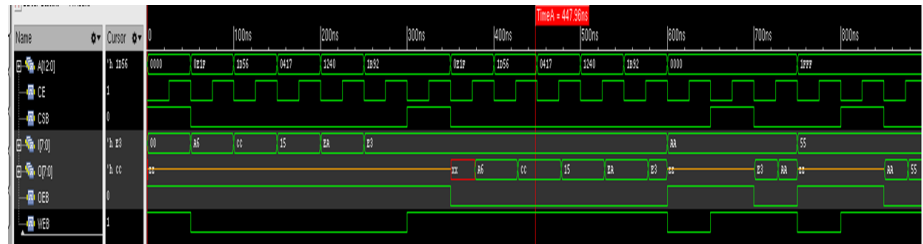


Figure 5.1: 8192x8 (8KB) Macro Simulation

The simulation results confirm the correct read and write operation of the 8192x8 SRAM macro. The waveform and console outputs verify proper address decoding, data storage, and data retrieval functionality. The results demonstrate stable synchronous

memory operation suitable for integration into larger memory architectures.

```

==== RANDOM WRITE PHASE ====
WRITE: Addr = 3615  Data = 0xa6
WRITE: Addr = 7510  Data = 0xcc
WRITE: Addr = 1047  Data = 0x15
WRITE: Addr = 4672  Data = 0xea
WRITE: Addr = 7058  Data = 0xe3

==== RANDOM READ & VERIFY PHASE ====
PASS: Addr 3615  Data = 0xa6
PASS: Addr 7510  Data = 0xcc
PASS: Addr 1047  Data = 0x15
PASS: Addr 4672  Data = 0xea
PASS: Addr 7058  Data = 0xe3

==== CORNER ADDRESS CHECK ====
PASS: Corner Addr 0 OK
PASS: Corner Addr 8191 OK

==== TEST COMPLETE ====

Simulation complete via $finish(1) at time
./tb_SPRAM_8192x8.v:128      $finish;
xcellium>

```

Figure 5.2: 8192x8 (8KB) Macro Simulation Console Output

5.1.2 64 KB SRAM Verification

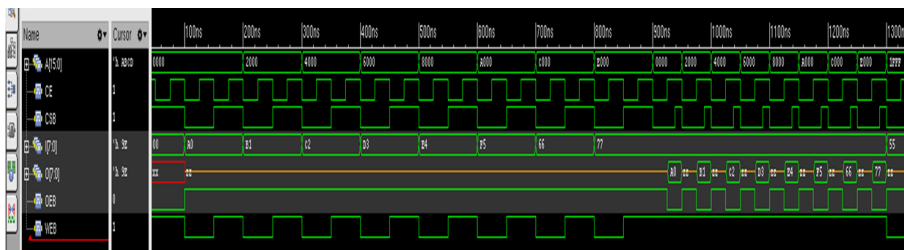


Figure 5.3: 64KB SRAM Simulation

The simulation results validate the successful integration of eight SRAM macros to form the 64 KB memory architecture. Correct bank selection, address partitioning, and data transfer operations are observed in the waveform results. The console output confirms reliable multi-bank memory access and overall functional correctness of the architecture.

```

=== STARTING 64KB (8192x8 banks) FUNCTIONAL TEST ===

--- Verifying bank decoding ---
PASS : Addr 0x0000 | Data 0xa0
PASS : Addr 0x2000 | Data 0xb1
PASS : Addr 0x4000 | Data 0xc2
PASS : Addr 0x6000 | Data 0xd3
PASS : Addr 0x8000 | Data 0xe4
PASS : Addr 0xa000 | Data 0xf5
PASS : Addr 0xc000 | Data 0x66
PASS : Addr 0xe000 | Data 0x77
PASS : Addr 0x1fff | Data 0x55
PASS : Addr 0x0000 | Data 0xa0
PASS : Addr 0xffff | Data 0xaa
PASS : Addr 0x1234 | Data 0x3c
PASS : Addr 0xabcd | Data 0x9e

=== FUNCTIONAL TEST COMPLETE ===

Simulation complete via $finish(1) at time 2138 NS + 0
./tb_SPRAM_64KB_8192x8.v:161      $finish;
xcelium> I

```

Figure 5.4: 64KB SRAM Simulation Console Output

5.1.3 Synthesis Results

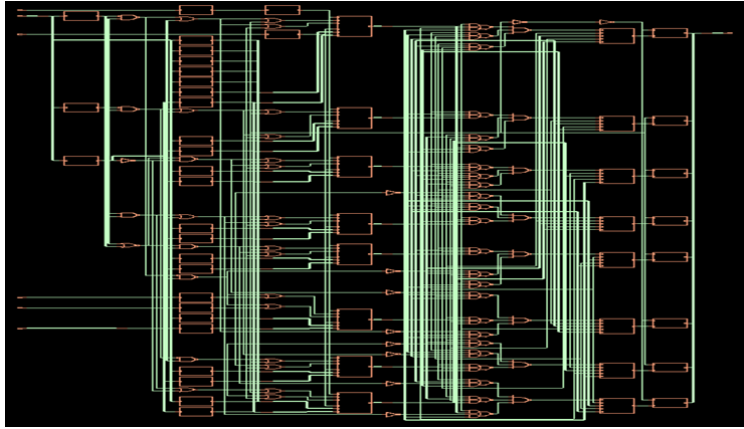


Figure 5.5: 64KB SRAM Schematic

The synthesized schematic confirms successful RTL-to-gate level conversion of the proposed 64 KB SRAM architecture. The generated netlist correctly represents the interconnection between SRAM macros, decoder logic, and memory control circuitry. The synthesis results demonstrate proper hardware realization of the proposed design.

```

=====
Generated by:      Genus(TM) Synthesis Solution 20.11-s111_1
Generated on:      Apr 13 2026  01:36:25 am
Module:           SPRAM_64KB_from_8192x8
Operating conditions:  ts118cio150_max (worst_case_tree)
Wireload mode:    enclosed
Area mode:        timing library
=====

```

Instance	Module	Cell Count	Cell Area	Net Area	Total Area	Wireload
SPRAM_64KB_from_8192x8		124	284318.122	163.521	284481.643	540000 (S)

(S) = wireload was automatically selected

Figure 5.6: 64KB SRAM Area Report

The area report shows that the majority of the chip area is occupied by the SRAM macros, while the standard cell logic contributes comparatively smaller area. The obtained area utilization demonstrates efficient banking-based memory integration suitable for large memory implementation in 180 nm technology.

```

Instance: /SPRAM_64KB_from_8192x8
Power Unit: W
PDB Frames: /stim#0/frame#0

```

Category	Leakage	Internal	Switching	Total	Row%
memory	2.24042e-04	1.35642e-03	3.44909e-06	1.58391e-03	40.24%
register	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
latch	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
logic	1.38421e-08	3.89351e-06	1.99535e-05	2.38609e-05	0.61%
bbox	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
clock	0.00000e+00	0.00000e+00	1.85703e-04	1.85703e-04	4.72%
pad	1.42928e-05	1.18860e-03	9.39652e-04	2.14254e-03	54.43%
pm	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
Subtotal	2.38349e-04	2.54891e-03	1.14876e-03	3.93602e-03	100.00%
Percentage	6.06%	64.76%	29.19%	100.00%	100.00%

Figure 5.7: 64KB SRAM Power Report

The power analysis indicates that the proposed SRAM architecture achieves low overall power consumption suitable for biomedical applications. The results demonstrate that the banking architecture and controlled memory access help reduce unnecessary switching activity and power dissipation.

5.1.4 Synthesis Summary

The synthesis results show that the proposed design successfully meets the target operating frequency of 20 MHz with significant timing margin. The architecture achieves low power consumption and moderate area utilization while supporting reliable SRAM operation. These results confirm the suitability of the design for low-power implantable applications.

Table 5.1: Performance Summary of Proposed 64KB SRAM Architecture

Parameter	Value
Technology	SCL 180 nm
Maximum Frequency	104.6 MHz
Clock Frequency	20 MHz
Standard Cell Area	0.284 mm ²
Macros Area (8)	2.949 mm ²
Total Area	3.233 mm ²
Total Power	3.936 mW

5.1.5 Post Synthesis LEC Results

Compare Results:		PASS
Number of EQ compare points:	16	
Number of NON-EQ compare points:	0	
Number of Aborted compare points:	0	
Number of Uncompared compare points :	0	

Figure 5.8: LEC Compare Results

Conformal Version	20.20-s200
Root Module	SPRAM_64KB_from_8192x8
CPU Time	17.0 sec
Elapsed Time	143.0 sec
Memory	127.95 MB
Power Grid Comparison	N/A
Power Intent Compare	N/A
Supply Power Consistency	N/A
Retention Power Consistency	N/A
Compare Power Crossing	N/A
LEC Comparison	PASS
Flatten Comparison	N/A
Hierarchical Comparison	PASS
Hierarchical Comparison Modules	1
Logfile Lines	571
Logfile Name	rtl2intermediate.lec.log
Logfile Modified Time	Sat Feb 28 00:20:18 2026
Implementation Information	fv/SPRAM_64KB_from_8192x8

Figure 5.9: LEC Comparison

The LEC results confirm that the synthesized gate-level netlist is functionally equivalent to the original RTL design. No logical mismatches were observed during comparison, ensuring correctness of the synthesis process and preserving intended design functionality.

5.1.6 Post Synthesis GLS Results

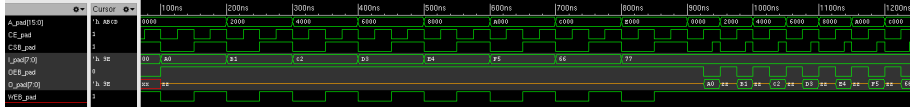


Figure 5.10: Zero Delay Gate Level Simulation

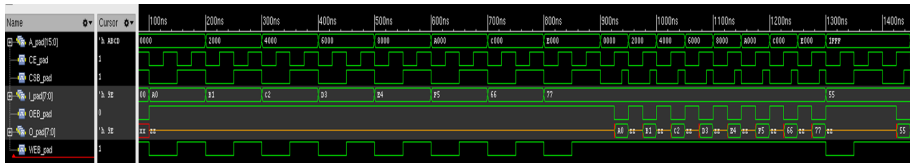


Figure 5.11: Unit Delay Gate Level Simulation

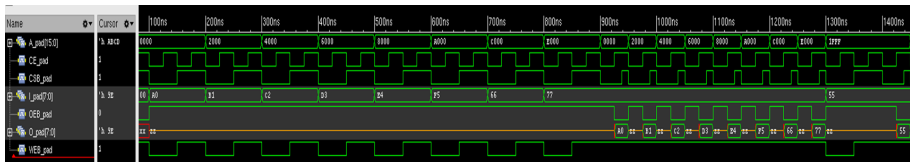


Figure 5.12: SDF Gate Level Simulation

The gate-level simulation results verify correct functional behavior of the synthesized design under zero delay, unit delay, and SDF delay conditions. The results confirm that the synthesized netlist operates correctly even after inclusion of gate and interconnect delays.

5.1.7 Floor Planning and Macro Planning Results

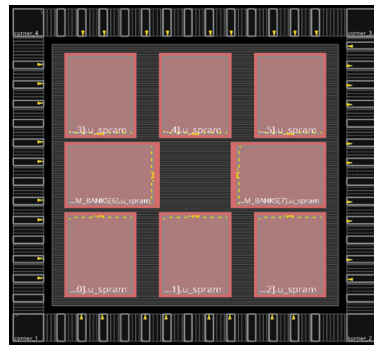


Figure 5.13: After FloorPlannig & Macro Placement

- Die Size : 3 mm $\times$ 3 mm
- Number of Macros : 8
- Core Area : 5.57 mm²
- Aspect Ratio : 1
- Core Utilization : 57.8%
- IO Pad Area : 2.75 mm²

The floorplanning results show successful placement of eight SRAM macros with balanced utilization and proper routing space between macros. Adequate routing channels and aspect ratio selection help improve routability and timing performance during later physical design stages.

5.1.8 Power Planning and Placement Results

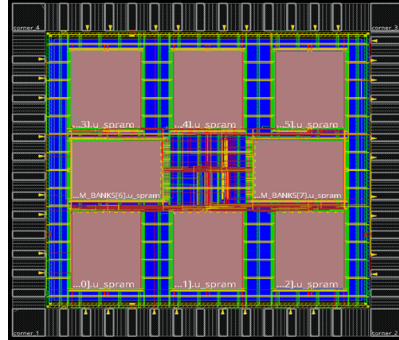


Figure 5.14: After Power planning & Placement

- Core & Macro Rings : M4(H), M3(V)
- Routing Channel Between Macros : 173 μ m

The placement results demonstrate successful standard cell placement around the SRAM macros while maintaining proper routing resources. Core rings and macro power rings ensure reliable power distribution across the chip and help minimize IR drop issues during operation.

5.1.9 CTS and Routing Results

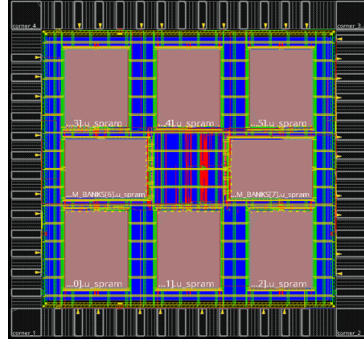


Figure 5.15: After CTS & Routing

- Max Frequency: 43.6 MHz
- Routing Congestion : 0
- Clock Skew : 0.2 ns
- DRV Violations : 0

The CTS and routing results indicate successful clock distribution with low clock skew and zero routing congestion. The absence of DRV violations confirms that the design satisfies timing and routing constraints after physical implementation.

5.1.10 Post Route LEC

```
Conformal Version |20.20-s200
Root Module      |SPRAM_64KB_from_8192x8
CPU Time         |1.0 sec
Elapsed Time     |71.0 sec
Memory          |83.63 MB
Power Grid Comparison |N/A
Power Intent Compare |N/A
Supply Power Consistency |N/A
Retention Power Consistency |N/A
Compare Power Crossing |N/A
LEC Comparison    |PASS
Flatten Comparison |PASS
Hierarchical Comparison |N/A
Hierarchical Comparison Modules |N/A
Logfile Lines     |447
Logfile Name      |intermediate2final.lec.log
```

Figure 5.16: Post Route LEC Comparison

The post-route LEC results confirm that the routed netlist remains functionally equivalent to the synthesized design. This ensures that no functional modifications were introduced during placement and routing stages.

5.1.11 Post Route GLS

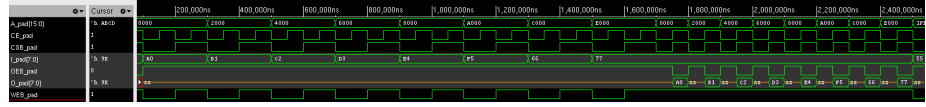


Figure 5.17: Zero Delay Simulation

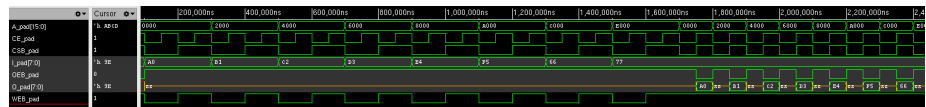


Figure 5.18: Unit Delay Simulation

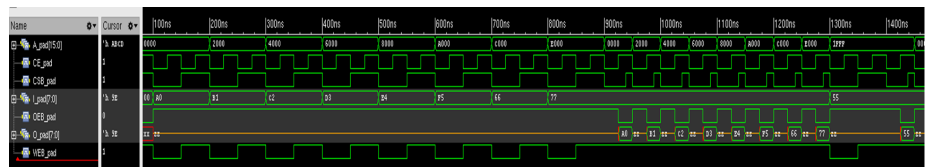


Figure 5.19: Post Route GLS SDF Simulation

The post-route GLS results validate correct functionality of the final routed design under realistic timing conditions. The simulations confirm successful operation after considering routing parasitics and extracted interconnect delays.

5.1.12 Signoff & Verification

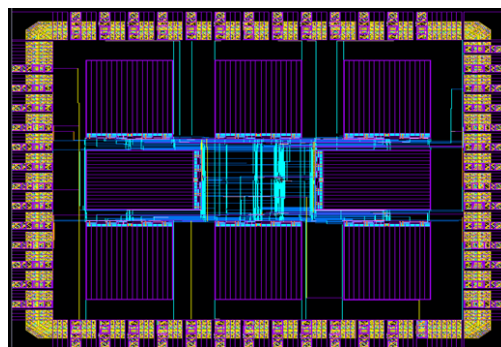


Figure 5.20: Signoff

The signoff results indicate successful completion of timing, routing, and physical verification stages. The design satisfies all major implementation constraints required for final tapeout readiness.

Check / Cell	Results
Check M5.C.1	1
Check DENSITY_PRINT_FILES	0

Figure 5.21: DRC Report

The physical verification results confirm that the final layout is free from design rule violations and matches the intended circuit netlist. Capacitance violations can be ignored. Successful DRC, LVS, and antenna verification demonstrate the correctness and manufacturability of the proposed layout.

The final implementation demonstrates successful realization of the proposed SPI-enabled SRAM architecture in 180 nm CMOS technology. The achieved frequency, power, and area results confirm the feasibility of the architecture for low-power biomedical memory applications.

5.2 SPI Enabled SRAM Results

5.2.1 SPI Enabled SRAM Verification

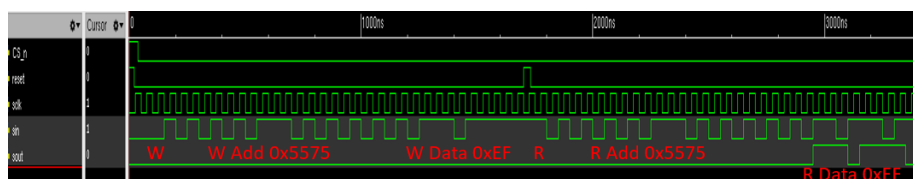


Figure 5.22: SPI Enabled Memory Simulation

The SPI simulation results confirm correct serial communication between the SPI controller and SRAM memory system. Proper read/write sequencing, serial data transfer, and FSM operation are successfully verified through the waveform results.

5.2.2 Synthesis Results

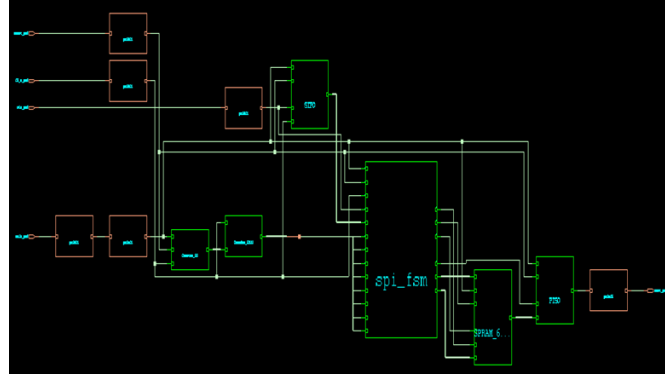


Figure 5.23: SPI SRAM Schematic

The synthesized SPI SRAM schematic confirms successful integration of SPI controller logic with the SRAM memory architecture. The generated hardware implementation correctly realizes serial communication and memory control functionality.

Instance	Module	Cell Count	Cell Area	Net Area	Total Area	Wireload
SPRAM_64KB_from_8192x8_SPI		274	250155.810	231.527	250387.338	280000 (S)
u_SPRAM_64KB_from_8192x8	SPRAM_64KB_from_8192x8	87	236308.430	66.225	236374.655	280000 (S)
u_counter	Counter_32	11	479.830	1.449	481.279	4000 (S)
u_decoder	Decoder_5X32	16	232.050	2.504	234.554	4000 (S)
u_piso	PISO	16	727.590	1.817	729.407	4000 (S)
u_sipo	SIPO	17	1066.300	0.000	1066.300	4000 (S)
uu1	spi_fsm	121	3568.970	38.525	3607.495	4000 (S)

(S) = wireload was automatically selected

Figure 5.24: SPI SRAM Area Report

The area report indicates that the additional SPI control logic introduces minimal area overhead compared to the SRAM macro area. This demonstrates efficient implementation of the SPI-enabled memory interface.

Instance: /SPRAM_64KB_from_8192x8_SPI					
Power Unit: W					
PDB Frames: /stim#0/frame#0					
Category	Leakage	Internal	Switching	Total	Row%
memory	2.22217e-04	8.41550e-04	3.35587e-06	1.06712e-03	52.47%
register	5.71596e-08	1.01739e-04	1.71886e-05	1.18984e-04	5.85%
latch	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
logic	2.46187e-08	9.37997e-06	2.25617e-05	3.19662e-05	1.57%
bbox	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
clock	7.54920e-10	8.03360e-07	1.87402e-04	1.88206e-04	9.25%
pad	2.00618e-06	4.47268e-04	1.78288e-04	6.27562e-04	30.86%
pm	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
Subtotal	2.24306e-04	1.40074e-03	4.08796e-04	2.03384e-03	100.00%
Percentage	11.03%	68.87%	20.10%	100.00%	100.00%

Figure 5.25: SPI SRAM Power Report

The power analysis shows that the SPI-enabled SRAM architecture maintains low power operation suitable for implantable systems. Controlled serial communication and efficient memory access contribute to reduced overall power consumption.

5.2.3 Synthesis Summary

Table 5.2: Performance Summary of Proposed 64KB SPI-Enabled SRAM Architecture

Parameter	Value
Technology	SCL 180 nm
Maximum Frequency	81.8 MHz
Clock Frequency	20 MHz
Standard Cell Area	0.25 mm ²
Macros Area (8)	2.949 mm ²
Total Area	3.2 mm ²
Total Power	2 mW

The synthesis results demonstrate that the SPI-enabled SRAM architecture successfully satisfies the target operating frequency with low area and power consumption. The results confirm efficient integration of communication and memory subsystems.

5.2.4 Post Synthesis LEC Results

```
Conformal Version | 20.20-s200
Root Module      | SPRAM_64KB_from_8192x8_SPI
CPU Time         | 130.0 sec
Elapsed Time     | 303.0 sec
Memory           | 149.66 MB
Power Grid Comparison | N/A
Power Intent Compare | N/A
Supply Power Consistency | N/A
Retention Power Consistency | N/A
Compare Power Crossing | N/A
LEC Comparison    | PASS
Flatten Comparison | N/A
Hierarchical Comparison | PASS
Hierarchical Comparison Modules | 5
Logfile Lines     | 1049
Logfile Name      | rtl2intermediate.lec.log
```

Figure 5.26: LEC Comparison

The LEC comparison confirms functional equivalence between the RTL and synthesized SPI SRAM design. This validates correctness of the synthesis process and preservation of intended SPI functionality.

5.2.5 Post Synthesis GLS Results

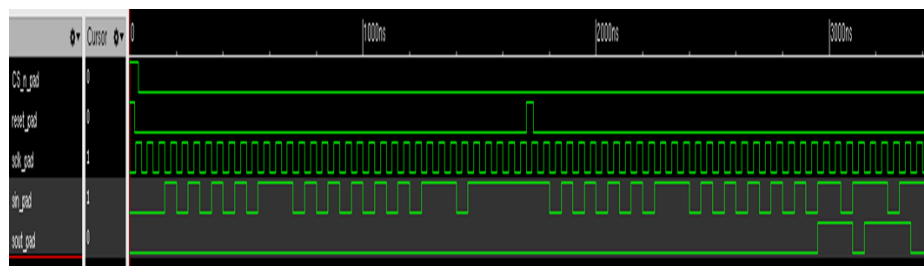


Figure 5.27: Gate Level Simulation

The gate-level simulation results verify correct operation of the SPI-enabled SRAM architecture after synthesis. The design functions correctly under timing-aware simulation conditions.

5.2.6 Physical Design Results

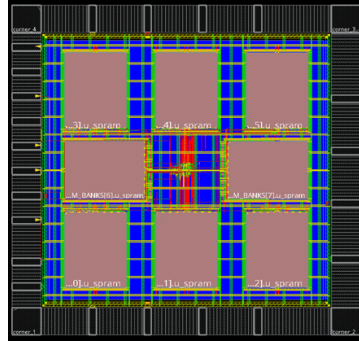


Figure 5.28: After Routing Stage

- Core and Macro Rings : M4(H), M3(V)
- Maximum Frequency : 48.95 MHz
- Routing Channel Between Macros : 173 μm
- Clock Skew : 0.2 ns
- IO Pad Area : 2.75 mm^2
- Routing Congestion : 0

The CTS and routing results show successful physical implementation of the SPI-enabled SRAM design with low clock skew and zero routing congestion. Proper routing resource allocation ensures reliable signal propagation across the chip.

5.2.7 Post Route GLS Results

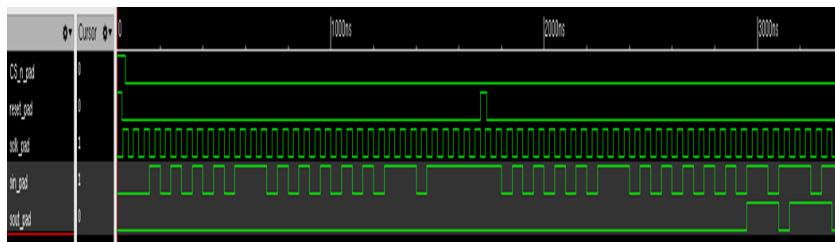


Figure 5.29: Gate Level Simulation

The post-route simulation results confirm correct operation of the final routed SPI-enabled SRAM architecture after inclusion of routing delays and parasitic effects.

5.2.8 Post Route LEC Results

```
Conformal Version | 20.20-s200
Root Module       | SPRAM_64KB_from_8192x8_SPI
CPU Time          | 1.0 sec
Elapsed Time      | 20.0 sec
Memory           | 89.91 MB
Power Grid Comparison | N/A
Power Intent Compare | N/A
Supply Power Consistency | N/A
Retention Power Consistency | N/A
Compare Power Crossing | N/A
LEC Comparison    | PASS
Flatten Comparison | PASS
Hierarchical Comparison | N/A
Hierarchical Comparison Modules | N/A
Logfile Lines     | 433
Logfile Name      | intermediate2final.lec.log
```

Figure 5.30: LEC Comparison

The post-route LEC results verify that the routed SPI SRAM netlist remains functionally equivalent to the synthesized design without introducing logical mismatches.

5.2.9 Signoff & Verification

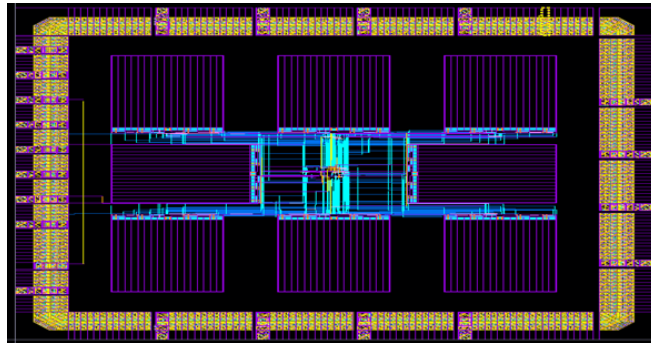


Figure 5.31: Signoff

Filter: Show All		SPRAM_64KB_from_8192x8_SPI, 2 Results	
Check / Cell /		Results	
✗ Check ML.C.1		1	2
✗ Check M5.C.1		1	
✓ Check DENSITY_PRINT_FILES		0	

Figure 5.32: DRC Report

The signoff verification confirms that the SPI-enabled SRAM architecture successfully satisfies physical verification and implementation requirements required for fabrication readiness.

Table 5.3: Comparison of 64 KB SRAM Architecture With and Without SPI Interface

Parameter	64KB SRAM	SPI-Enabled 64KB SRAM
Technology	SCL 180 nm	SCL 180 nm
Maximum Frequency	43.6 MHz	48.95 MHz
Target Clock Frequency	20 MHz	20 MHz
Total Area	3.233 mm ²	3.2 mm ²
Total Power	3.936 mW	2 mW

The comparison results show that both architectures successfully satisfy the target operating frequency of 20 MHz in SCL 180 nm CMOS technology at 1.8V supply voltage. The SPI-enabled SRAM architecture achieves lower total power consumption of 2 mW compared to 3.936 mW for the standalone SRAM architecture, while maintaining nearly similar chip area. The SPI-enabled memory provides efficient serial communication, reduced interface complexity, and improved suitability for low-power implantable biomedical applications.

5.3 Final Results Summary

The complete RTL-to-GDSII implementation of the proposed SPI-enabled 64 KB SRAM architecture was successfully carried out using Cadence EDA tools in SCL 180 nm CMOS technology. Functional verification, synthesis, physical implementation, and post-route verification confirmed the correctness and reliability of the proposed design.

Post-route implementation results show that the proposed SPI-enabled SRAM architecture achieved a maximum operating frequency of 48.95 MHz while meeting the target operating frequency of 20 MHz. The total chip area obtained was approximately 3.2 mm², including SRAM macro area and standard cell area. The overall power consumption of the complete architecture was approximately 2 mW.

The final results demonstrate that the proposed architecture provides an efficient, scalable, and low-power external memory solution suitable for implantable biomedical applications requiring reliable EGM signal buffering and SPI-based memory interfacing.

Chapter 6

Conclusion and Future Scope

6.1 Summary of Work

This thesis presented the design and implementation of a low-power SPI-enabled SRAM architecture for implantable pacemaker applications. The work focused on developing an efficient external memory system for buffering and storing electrogram (EGM) signals in implantable biomedical devices.

An 8192×8 single-port SRAM macro was implemented as the basic memory block. Eight SRAM macros were integrated using banking architecture to construct a 64 KB memory system. Address partitioning and decoder-based bank selection were used for efficient memory access.

An SPI Mode-3 communication interface was integrated with the SRAM architecture using FSM-based control logic, counters, and SIPO/PISO registers to support serial read and write operations.

The complete architecture was modeled using Verilog HDL and verified through functional simulation. The design was implemented using the Cadence RTL-to-GDSII flow in SCL 180 nm CMOS technology, including synthesis, floorplanning, power planning, placement, CTS, routing, and physical verification.

6.2 Conclusion

The proposed SPI-enabled 64 KB SRAM architecture for implantable pacemaker applications was successfully designed and implemented using SCL 180 nm CMOS technology. The architecture demonstrated efficient integration of SPI communication with banking-based SRAM organization suitable for low-power biomedical memory applications.

The final implementation achieved a maximum operating frequency of 48.95 MHz with a target operating frequency of 20 MHz. The total chip area obtained was approximately 3.2 mm² with a total power consumption of nearly 2 mW. The design achieved low routing congestion, clock skew of 0.2 ns, and zero DRV violations during physical implementation.

Functional verification, gate-level simulation, and post-route verification confirmed correct SRAM and SPI operation. The final layout successfully passed DRC, LVS, and antenna verification checks, demonstrating the correctness and manufacturability of the proposed architecture.

6.3 Limitations

Although the proposed architecture successfully demonstrates SPI-enabled SRAM integration for biomedical applications, certain limitations still exist.

- The implemented architecture uses conventional SRAM macros without advanced low-leakage SRAM optimization techniques.
- The work focuses mainly on digital implementation and does not include transistor-level SRAM analysis such as SNM characterization.
- The design is implemented in 180 nm CMOS technology, resulting in larger silicon area compared to advanced technology nodes.

6.4 Future Scope

The proposed work can be extended further to improve power efficiency, scalability, and performance.

- Advanced SRAM architectures such as 8T or 10T SRAM cells can be used to improve stability and reduce leakage power.
- The design can be migrated to smaller technology nodes to reduce area and power consumption.
- High-speed interfaces such as Quad SPI (QSPI) can be implemented for faster memory access.
- Additional low-power techniques such as clock gating and power gating can be incorporated.
- Error correction and fault-tolerant techniques can be added to improve reliability for biomedical applications.
- The architecture can be extended for larger memory capacity and multi-channel biomedical signal buffering systems.

Bibliography

- [1] L. J. Stotts, “Introduction to implantable biomedical ic design”, *IEEE Circuits and Devices Magazine*, vol. 5, no. 1, pp. 12–19, 1989.
- [2] R. S. Sanders and M. T. Lee, “Implantable pacemakers”, *Proceedings of the IEEE*, vol. 84, no. 3, pp. 480–486, 1996.
- [3] S. E. González Díaz, “Methodology, design, and implementation of a cardiac pacemaker prototype using a commercial low power microcontroller”, Master’s thesis, University of Puerto Rico, Mayagüez Campus, 2006.
- [4] S. Chede and K. Kulat, “Design overview of processor based implantable pacemaker”, *Journal of Computers*, vol. 3, no. 8, pp. 49–57, 2008.
- [5] Shree Pacetronix Ltd., “Specifications of implantable pacemaker asic (spl2)”, 2022.
- [6] Vikas, “Pacetronix: Spl-2 requirements”, March 2022.
- [7] L. H. S. C. Wong and A. Walker, “Leakage current cancellation technique for low power switched-capacitor circuits”, Oct 2003.
- [8] P. K. Swain, S. Sarangi, and S. Rout, “A compact spi-based sram in 0.6 μ m cmos for low-power iot applications”, in *2023 1st International Conference on Circuits, Power and Intelligent Systems (CCPIS)*, pp. 1–6, IEEE, 2023.
- [9] R. Balas, A. Ottaviano, and L. Benini, “Cv32rt: Enabling fast interrupt and context switching for risc-v microcontrollers”,

IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 32, no. 6, pp. 1032–1044, 2024.

- [10] S. S. Pai and A. D. S, “Performance improvement of 9t sram cell by efficient memory design”, in *2025 2nd International Conference on Trends in Engineering Systems and Technologies (ICTEST)*, pp. 1–4, IEEE, 2025.
- [11] A. B. Yandrapati, H. K. Yakkanti, S. Nilagiri, and P. Domathoti, “Design and implementation of risc-v based pacemaker”, *Proceedings of the 8th International Conference on Communication and Electronics Systems (ICCES)*, pp. 123–127, 2023.
- [12] E. Abbasian and M. Gholipour, “Improved read/write assist mechanism for 10-transistor static random access memory cell”, *International Journal of Circuit Theory and Applications*, vol. 50, no. 10, pp. 3642–3660, 2022.
- [13] S. Yu, *Semiconductor Memory Devices and Circuits*. Boca Raton, Florida: CRC Press, 1st ed., 2022.
- [14] K. Cho, J. Park, T. W. Oh, and S.-O. Jung, “One-sided schmitt-trigger-based 9t sram cell for near-threshold operation”, *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 5, pp. 1551–1561, 2020.
- [15] N. H. E. Weste and D. M. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*. Pearson, 5th ed., 2020.
- [16] S. Pal, S. Bose, W.-H. Ki, and A. Islam, “A highly stable reliable sram cell design for low power applications”, *Microelectronics Reliability*, vol. 105, p. 113503, 2020.
- [17] S. Pal, S. Bose, W.-H. Ki, and A. Islam, “A highly stable reliable sram cell design for low power applications”, *Microelectronics Reliability*, vol. 105, p. 113503, 2020.
- [18] K. Shin, W. Choi, and J. Park, “Half-select free and bit-line sharing 9t sram for reliable supply voltage scaling”, *IEEE Trans-*

actions on Circuits and Systems I: Regular Papers, vol. 64, no. 8, pp. 2036–2047, 2017.

- [19] A. W. C. Chow and A. E. Buxton, eds., *Implantable Cardiac Pacemakers and Defibrillators: All You Wanted to Know*. Oxford, UK: Blackwell Publishing Ltd, 2006.
- [20] A. B. Yandrapati, H. K. Yakkanti, S. Nilagiri, and P. Domathoti, “Design and implementation of risc-v based pacemaker”, *Proceedings of the 8th International Conference on Communication and Electronics Systems (ICCES)*, pp. 123–127, 2023.