Methodology and Validation of Power Management Flows

A Thesis report submitted in partial fulfillment of the requirement for the Award of the Degree of

MASTER OF TECHNOLOGY

in VLSI DESIGN

Submitted By

Vibha Sharma

602362041

Under Supervision of

Dr. Hardeep Singh

(Associate Professor)

Keshava B Murthy

(Manager, Intel Corporation)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

THAPAR INSTITIUTE OF ENGINEERING AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY), PATIALA, PUNJAB

JUNE 2025

DECLARATION

I, **Vibha Sharma** hereby declare that the work presented in this thesis entitled **" Methodology and Validation of Power Management Flows"** in partial fulfilment of the requirement for the award of degree of **Master of Technology (VLSI Design**) submitted at **Department of Electronics and Communication Engineering**, Thapar Institute of Engineering & Technology (Deemed to be University), Patiala is an authentic record of work carried out under supervision of Industry Mentor **Mr. Keshava B Murthy (Manager, CVE CSVE BDC, Intel Technology India Pvt Ltd)** and **Dr. Hardeep Singh (Associate Professor, Department of Electronics and Communication Engineering)** matter presented in this has not been submitted either in part or full to any other university or institute for the award of any other degree.

Vibha Sharma

Date: 28 June, 2025

Vibha Sharma
(602362041)

B. S Keshava Murthy

Keshava B Murthy
(Manager, CVE CSVE BDC, Intel)

Dr. Hardeep Singh
(Associate Professor, DECE, T.I.E.T, Patiala)

INTERNSHIP CERTIFICATE

Regd. Office:
Intel Technology India Private Limited
23-56P, Outer Ring Road,
Devarabeesanahalli, Varthur Hobli website: www.intel.in
Bellandur Post
Bangalore 560 103, India
CIN-U85110KA1997PTC021606

Tel: +91-80-2605 3000
Fax: +91-80-2605 6190

To Whomsoever It May Concern

WWID: 12258839

Employee Name: Vibha Sharma

Internship Dates: 24/06/2024 to 23/06/2025

The letter is to confirm the mentioned above has undergone internship at Intel Technology India Private Limited.

We wish you all the best for your future assignments.

Yours Sincerely

Gowre Saseedaran

Date: June 27, 2025

Place: **Bangalore**

ACKNOWLEDGEMENT

This thesis would not have been possible without the guidance and the help of several individuals who in one way or another contributed and extended their valuable assistance in the preparation and completion of this study.

First and foremost, my sincere gratitude to Dr. Hardeep Singh for his continuous support, motivation, immense knowledge. His guidance has helped me in all the time of research and writing of the thesis.

.

Keshava B Murthy Manager of my team CVE CSVE BDC, Intel Pvt Ltd for his patience and steadfast encouragement, his constructive criticisms at different stages of my work were thought-provoking and he helped me focus on my ideas. I am deeply grateful to him for all discussions that helped me in understanding the technical details of my work.

Last but not the least, my family and dear friends. None of this would have been possible without their love. And the one above all of us, the omnipresent God for giving me the strengthto plod on, thank you so much everyone.

ABSTRACT

The quick rise of mobile devices, high-performance computers, and data centres has made it necessary to use advanced power management methods to cut down on energy use without hurting performance. Dynamic Voltage and Frequency Scaling (DVFS) is one of the best ways to control how much power processors use. It changes the voltage and frequency based on how much work they have to do. This report talks about the ideas, uses, and power controller architectures that make DVFS work, focusing on its importance in today's power management strategies.

The report also talks about different power management states, such as C-states, G-states, and P-states, and how they work together to make the system use less power when it's idle and when it's working. The study talks about two types of power management: idle power management, which tries to use less power when the system is not doing much, and active power management, which tries to use power more efficiently when the system is working hard. The method includes looking at current power management models and getting a full picture of the technologies involved. The report also talks about the problems and limits of DVFS, such as latency, lower performance when scaling, and the difficulties of implementing it in multi-core and heterogeneous systems. The report gives a complete picture of how dynamic voltage and frequency adjustments can save a lot of energy and improve performance by looking at real-world uses like mobile devices, embedded systems, and data centres. The conclusion talks about possible future directions for research on power management, especially in relation to more advanced processors and systems that care about energy.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation: Description

SoC: System on Chip

IP: Intellectual Property

SUT: System under test

BIOS: Basic input output system

NOC: Network on Chip

MBIST: Memory Built-In Self-Test

RAM: Radom Access Memory

RVP: Reference Validation Platform

ROM: Read only Memory

PI: Primary Input

PO: Primary Output

TAP: Test Access Port

JTAG: Joint Test Action Group

FSM: Finite State Machine

RTL: Register Transfer Language

PPW: Performance per Watt

CHAPTER 1
INTRODUCTION

1.1 Introduction to the area of work

The semiconductor industry has made optimizing power and performance a top design goal, especially since the demand for more efficient digital devices keeps growing. This is especially true for System on Chips (SoCs), which are the brains of modern electronic devices like smartphones, tablets, and high-performance computers. In the past, faster clock speeds were the main way to improve performance, but this method has hit a wall because it uses more power. When a lot of power is used, problems like heat dissipation happen that can degrade quality of chip and created heat problem, which stops performance from getting better. Because of this, making sure that both performance and power use are as good as possible has become a big problem. Power management techniques in SoCs are important for making them use less energy without slowing them down. The goal of these techniques is to lower both dynamic and static power use. Dynamic Voltage and Frequency Scaling (DVFS), for example, lets you change the voltage and frequency in real time to match the workload. This saves power when the workload is less demanding. But managing power is a difficult job that needs careful adjustment of many things, such as supply voltage, operating frequency, and leakage current. As silicon transistors get smaller, it gets even harder to manage power. As the size of transistors gets smaller, the amount of static power they use goes up a lot because of leakage current. This phenomenon makes it harder to optimize power, especially for SoCs made with 130nm technologies or less. As the industry moves toward smaller process nodes, it is becoming more important to find ways to reduce leakage power without hurting performance. The goal of this report is to look into different ways to manage power and see how well they work to lower power use in SoCs when they are both idle and active. We focus on techniques like DVFS, idle power management, and power gating because they are very important for making modern SoC designs more energy efficient [1].

1.2 Motivation of the work

The main reason for doing this research is the urgent need to make sure that semiconductor processors can meet both high performance and low power consumption requirements. As the electronics industry makes more advanced SoC designs, it is becoming more and more important to not only make devices that are faster and more powerful, but also to manage the power they use in an efficient way. As we rely more and more on mobile devices, wearables, and other portable electronics, processors that use less energy are important for extending battery life and reducing the impact on the environment.

In this situation, post-silicon validation is an important step in making sure that the power management techniques built into SoCs work and are reliable. After making the device, it's important to make sure that the power management flows, such as dynamic voltage frequency scaling, idle power modes, and power gating mechanisms, work as they should in real life. This process of validation includes testing the hardware, firmware, and software to make sure that all of the parts work properly under different workloads and environmental conditions. Also, working with experts in the field and doing stress tests and stability runs are important for finding possible problems with hardware design, software performance, and thermal management. The goal is to tweak the power management system so that it works as well as possible and uses as little energy as possible, without causing problems like overheating, instability, or using too much power [1].

1.3 Problem statement

1) Ensuring Functional Accuracy on Real Hardware : During Fabrication process or in pre silicon stage, a bug can be introduced in silicon. So it is crucial to validate that power management flows and reset operations are functioning correctly as designed on actual silicon after fabrication process. This validation is necessary to guarantee system stability and performance in practical use.

2) Identification and Debugging of System -Level Bugs: Post-silicon validation frequently reveals bugs spanning hardware, integrated firmware, and operating systems. Capturing and diagnosing these issues involves initializing the system with up-to-date OS and integrated firmware (IFWI) and executing various power states (Sleep, Hibernate and Shutdown) and reset flows (warm and cold resets) under stress and reliability tests.

3) Coordinated Issue Resolution and Re-Validation: Resolving the identified issues necessitates collaboration with firmware, hardware, and software experts to debug the failure logic and release fixes. After implementing the fixes, the failure flows must be re-validated to ensure the issues are resolved. Utilizing tools like Logic Analyzers and Oscilloscopes for precise debugging and result capture, this iterative process aims to enhance the stability and performance of semiconductor Processor in real world environments.

CHAPTER 2

LITERATURE REVIEW

Efficient power management in System on Chips (SoCs) has emerged as a vital area of research due to the increasing demand for high-performance, low-power digital systems. Numerous strategies have been explored to reduce both dynamic and static power dissipation, aiming to strike a balance between performance and energy efficiency.

Dynamic power consumption in SoCs arises primarily due to switching activities of transistors. As clock frequencies increase, dynamic power consumption rises exponentially. Several approaches have been developed to address this issue:

2.1 Dynamic Voltage and Frequency Scaling (DVFS)

Dynamic power constitutes a substantial portion of the total energy consumption in contemporary SoC designs. It primarily results from the switching activity of transistors during logical transitions. Every change in logic state from '0' to '1' (or vice versa) charges or discharges a node's capacitance, drawing energy from the power supply. This form of power usage can be modeled using the following equation:

$$P_{dynamic} = \alpha \cdot CL \cdot V_{dd}^2 \cdot f$$

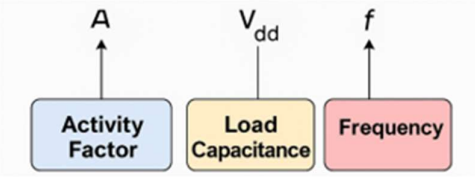

Figure 2.1: Dynamic Power Consumption in CMOS Logic [1]

Here, α is the activity factor, CL denotes the load capacitance, Vdd is the supply voltage, and f is the clock frequency. As illustrated in Figure 2.1, the relationship shows that dynamic power increases linearly with switching frequency and quadratically with voltage, making it a significant design consideration, especially with increasing integration and performance demands. To address dynamic power issues, a suite of techniques has been developed. Dynamic Voltage and Frequency Scaling (DVFS) is one of the most commonly used approaches. It takes the supply voltage and operating frequency in real time, on the basis of performance requirements and conserving energy when full processing capacity is not required.

Clock gating is another important method that stops components from switching on and off stage when they don't need to by turning off the clock signal. More advanced solutions, like adaptive body biasing and multi-voltage domains, let you optimize power and performance in specific areas of the chip by changing the operating conditions for each functional unit. These strategies are particularly beneficial for mobile devices, where reducing heat and extending battery life are primary concerns. Good dynamic power management also makes sure that thermal limits and performance goals are met at the same time [1].

Dynamic Voltage and Frequency Scaling (DVFS) allow real-time enabling and disabling of both voltage and frequency as per the requirement of workload activity, By decreasing the voltage when there is no activity can decrease power consumptions significantly due to the square-law relationship with Vdd. This ability to adapt to changing conditions lets processors keep running efficiently in a wide range of applications without significantly lowering performance [1].

Clock Gating is a technique used to decrease the switching power by stopping the clock signal to unused sections of a circuit. Since clock signals are a primary source of toggling activity in digital systems and at higher frequency it become prominent as switching increases, it prevent energy consumptions when circuit is not in use. Working Principle: A gating signal determines whether a particular block of logic requires a clock. If the logic is inactive, the gating mechanism suppresses the clock signal, thereby minimizing switching events and conserving power. This technique is straightforward to implement and highly effective, making it a staple in low-power design strategies. Circuit diagram of clock gating using flip flop is shown in figure 2.2.

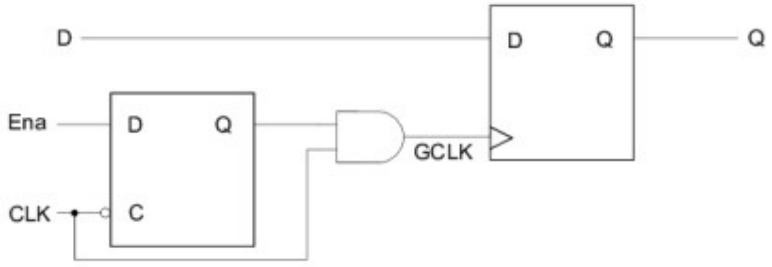

Figure 2.2: Clock Gating Using D-Flipflop [1]

2.2. Static Power Management Techniques

As device geometries continue to shrink, **static power**, predominantly caused by leakage currents, has become an increasingly significant contributor to total power dissipation. Addressing static power is essential, particularly in advanced technology nodes.

Power Gating combats leakage power by completely disconnecting the power supply from inactive blocks using sleep transistors. These high-threshold voltage transistors are inserted between logic circuits and either ground or power rails. When a circuit block is not in use, the sleep transistors are turned off, thereby eliminating leakage paths and placing the block in a low-power state. This approach is highly effective in idle states and contributes significantly to overall power savings, particularly in mobile and always-on devices. Block diagram of power gating shown in figure 2.3.

Body Biasing is another static power reduction technique that involves modulating the threshold voltage of transistors by applying a bias to their body terminals. By increasing the threshold voltage, leakage currents can be curtailed during idle states. The bias can be dynamically adjusted, providing a trade-off between performance and leakage power, depending on the operational context [2].

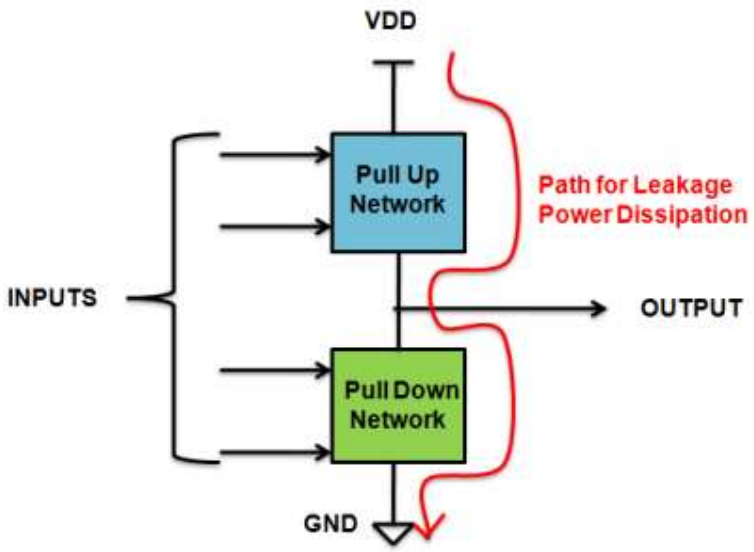

Figure 2.3: Power Gating [2]

2.3 Challenges in Power Management

Power management in today's SoCs needs to find a balance between performance and energy efficiency that is hard to find. Techniques like DVFS and clock gating lower power usage, but if they aren't set up correctly, they can make the system less responsive. As transistors get smaller, leakage power has become a major problem, even when the device is not in use. Even though power gating and body biasing help, they are still hard to use effectively. As SoCs get more complicated with multiple cores and different types of components, it gets even harder to manage power across these blocks. Each unit may need different amounts of power and performance. Mobile and embedded systems need fast, flexible power control, which means they need systems that can respond in real time with as little overhead as possible. Higher power density also causes thermal problems, so thermal-aware power strategies are needed to keep things from getting too hot and causing reliability problems. For power management to work, the hardware and software layers need to work together very closely. When things don't match up, they can work poorly or waste power. Finally, it is hard to check power-aware designs because there are so many power states and transitions. This means that strong validation methods are needed.

CHAPTER 3

POWER MANAGEMENT

This chapter discusses the proposed techniques for managing power consumption in System on Chips (SoCs). The focus is on combining existing strategies with novel approaches to optimize both idle and active power consumption while maintaining system performance. The proposed techniques leverage dynamic adjustments and architectural modifications to achieve power efficiency.

3.1 System-Wide Overview of Power management

Hardware from the system, devices, system software, and drivers must all work together to provide power management. The industry specifications, which were discussed in the previous section, address necessary hardware support. This issue deals with software support, specifically what drivers need to do to manage power appropriately for their devices and to comply with operating system requirements.

In addition, the power manager offers an interface for drivers that includes necessary driver entry points, power management minor IRPs, and support routines. Drivers respond by setting their devices to the proper device power state when the power management requests a change to the system power state. Furthermore, drivers have the ability to identify idle devices and place inactive devices into sleep mode. Device power capabilities are reported, device status is established and reported, and device power is changed by bus-specific procedures. The type of gadget and its hardware capabilities determine the precise method and timing of power changes [2].

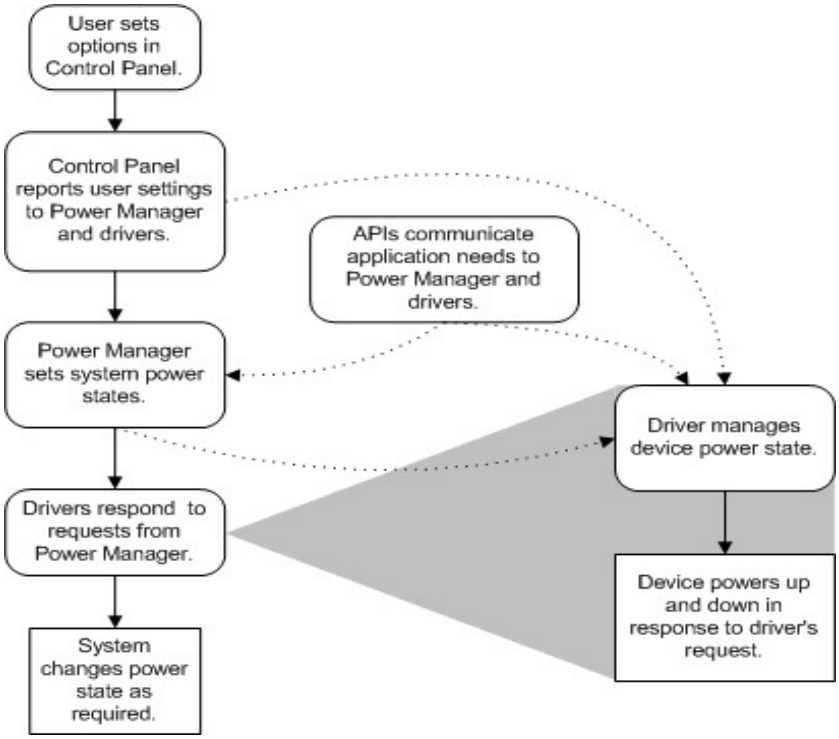

Fig 3.1 overview of power management [3]

3.2 Dynamic Voltage and Frequency Scaling (DVFS)

Dynamic Voltage and Frequency Scaling (DVFS) is a way to manage power by changing the voltage and frequency of a processor based on the needs of the workload. It makes sure that the system uses less energy without slowing it down.

DVFS is an important way to deal with the trade-off between power and performance in modern processors. DVFS helps lower power use during times of low activity and boost performance during times of high activity by changing the voltage and frequency on the fly [3].

3.2.1 Principles of DVFS

The Power calculated for DVFS depends on voltage, frequency and capacitor . Power consumption in CMOS circuits can be expressed as:

$P = C \cdot V2 \cdot f$

where P is power, C is capacitance, V is voltage, and f is frequency. By lowering V and f, significant power savings can be achieved. However, this will reduce the operating speed, so there is a trade-off between them [3].

Applications of DVFS

DVFS is widely used in the various domains, including:

• Mobile Devices: We can increase the operating battery life by recducing the power consumption during the idle time and when the system is not in used in or in sleep state.

• Data Centers: We can increase server efficiency and reduce operational costs in terms of electricity.

• Embedded Systems: To manage power in systems with limited energy resources [3].

3.2.2 Power Controller Architecture

The Power Controller Architecture is used to control the scaling of voltage and frequency across system IPs and its block . This makes sure that energy is used efficiently while performance stays at its best. It uses different block for its operation, as shown in figure 3.2 :

• Voltage Regulation (VR): This block gives controlled voltage levels to the system based on inputs from the DVFS controller.

• Clocks: Clock synchronizes operations between the architecture to maintain performance.

• RAM and FSM: They are responsible for storing data like states and data during state transitions in response to workload changes.

• DVFS Controller: Executes the scaling policies and communicates with other modules to optimize performance and power [3].

Advantages of DVFS

There are many benefits of using DVFS :

• **Lowering Power Consumption**: DVFS cut down the power used by changing voltage and frequency used.

• **Longer Battery Life**: DVFS extends the time that devices can be used which is important for potable or mobile devices.

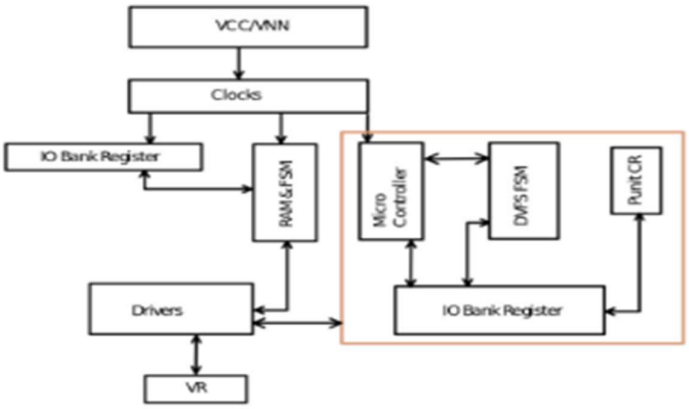

Figure 3.2: Power Controller Architecture [4]

• **Thermal Management**: DVFS control heat generation, enhancing device longevity and reliability by reducing power consumptions.

Limitations of DVFS

DVFS also have some limitations:

• **Latency:** The performance of device degraded due to increase of switching time between frequencies.

• **Complexity**: To implement DVFS, lot of hardware block needed that will eventually increases complexity.

• **Reduced Performance**: Lowering voltage and frequency can reduce computational performance as speed slows down [4].

3.2.3 Adaptive Control Mechanism

The adaptive control mechanism keeps an eye on the workload in real time and changes the voltage and frequency as needed. This system is different from traditional DVFS because it looks at more than just the current workload. It also looks at the workload history and predictive trends, which makes transitions smoother and saves more energy [4].

Multi-Level Scaling

Multi-level DVFS breaks up workloads into smaller pieces, which lets you change voltage and frequency more accurately. By making scaling more precise, the system can find a better balance

between saving power and meeting performance needs [4].

3.3 Idle Power Management Enhancements

Power management in modern electronic systems can be divided into two main approaches: **Idle Power Management** and **Active Power Management**.

These classifications aim at different operational states of a processor or system to save energy while keeping performance and responsiveness high [5].

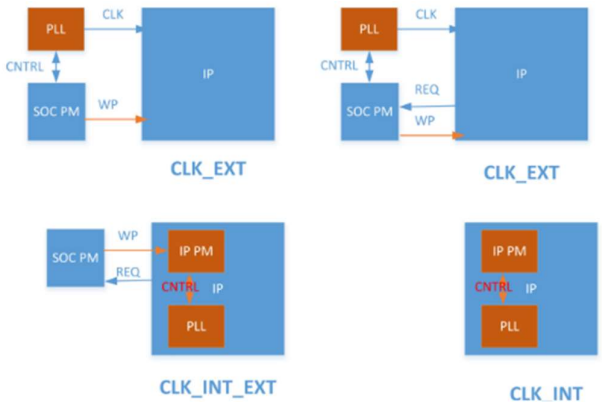

Figure 3.3 : Clock configurations [5]

Idle power management focuses on minimizing energy consumption during periods when the processor or system is not actively executing tasks. This is achieved by employing strategies that reduce power usage without hindering the system's ability to resume operations efficiently. The clk configuration of idle PM shown in fig 3.4 [5].

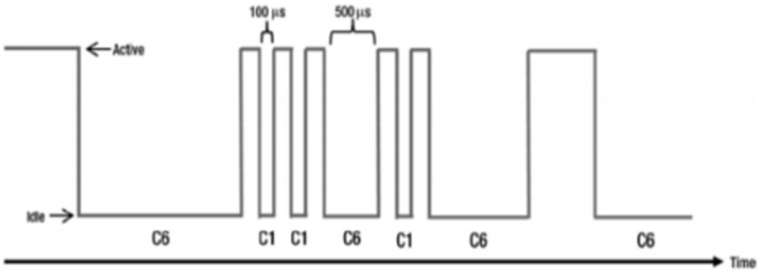

Figure 3.4: Clock configurations for Idle PM [5]

3.3.1 Role of system, Core and Package C States

During analysis of power data, various states of system like system, core and package states were checked. These states tell us for how much percentage of time, system / core / package is in active working state or inactive state which gives the idea about power consumption of that system / core / package.

System states:

System means complete computing device upon which SoC, all remaining fundamental peripherals like memory, display, external USB connections, etc., comes and sits [6].

S0 = full active mode.

S1/S2 = sleep mode.

S3 = suspended to RAM, only the system memory is retained.

S4 = suspended to disk, means disk containing the hibernate file isgetting back.

S5 = system is completely shut down.

Note: Higher state number means less power consumption and longer exit latency. These states are similar to that of power options in our desktop system,like-

S3 = sleep

S4 = Hibernate

S5 = Shut down

• **C-States**:. Utilizing deeper idle states to reduce both leakage and dynamic power by powering down inactive components such as caches and internal clocks [6].

C0 = Normal code execution state, means all our cores were working and P-unit thinks that we are doing some work.

C1 = core clocks local clock gating, means PLL is ON but the line between PLL and core is OFF.

C1E = same as above but transition to Vmin (operating voltage is less, like 0.7V or any low value to support the idle state).

C3 = core cache memories L1 & L2 were flushed. And interface between core and last level cache

(LLC) is turn off for that core.

C6 – C10 = lower core states. Cores were completely turn off, we can say power gated.

Package states:

Package means it includes IPs in silicon like Last Level Cache (LLC), System Agent (SA), Memory Controller (MC), Graphics core, etc. [6].

PC0 = All cores: big, atom and graphic are active.

PC2 = All cores are OFF. Ring is scoopable.

PC3 = Main memory is clock gated.

PC6= System agent PLLs are clock gated.

PC7 = IP is partially power gated.

PC8 = IP is completely power gated.

PC9 = Internal power rails are OFF

PC10 = Internal power rails, input power rails and reference clock are off.

Applications

Widely used in devices with frequent idle periods, such as smartphones, laptops, and data center servers.

Particularly effective in standby scenarios, where systems experience extended idle durations.

Challenges

Latency: Transitioning between idle and active states introduces delays, which can impact real-time operations.

Prediction Accuracy: Correctly estimating idle periods is crucial to ensure optimal power savings without affecting performance [6].

3.4 Active Power Management

Active power management addresses power optimization during periods of active operation. This is used for dynamically adjusting system parameters to balance energy efficiency and performance. There are many voltage regulators which are used to provide power while the system is running which is shown in fig 3.5.

Techniques

Dynamic Voltage and Frequency Scaling (DVFS)**:** Scaling the voltage and frequency on the basis of instructions activity to optimize dynamic power consumption.

Adaptive Voltage Scaling (AVS): Decreasing the operating voltage to the lowest feasible level that still maintains operation of device.

Dynamic Power Partitioning**:** Regulating the powers between components on the basis of their activity, so that power can be saved from low activity units [4].

Applications

Important for workloads that need different amounts of processing power, like gaming, multimedia processing, and high-performance computing.
Mainly used in energy-constrained devices to extend battery life without significantly degrading performance [4].

Challenges

Performance Degradation : On decreasing frequency and voltage can increase execution time of instruction that will affect latency-sensitive applications.
Workload Prediction: Accurately predicting workload variations is essential for effective power.

It's very important to lower power use when the device is idle, especially for apps that spend a lot of time idle. The suggested methods include making power gating and sleep mode transition better [4].

Enhanced Power Gating

By adding low-leakage transistors and decreasing the time it takes to switch from active to idle states makes power gating better. These changes lower leakage currents without adding much latency when waking up components that are not in use [4].

Dynamic Sleep Modes

Dynamic sleep modes allow for multiple levels of idle states, ranging from light sleep to deep hibernation. This method automatically selects the best sleep state based on the how long the device is expected to be idle, which will save power and reduce the wake-up time [4].

3.5 COMPARISON OF C-STATES, G-STATES, AND P-STATES

Power states in processors are divided into different types of states to manage power consumption efficiently under different workload activity. C-states, G-states, and P-states have distinct purposes and uses, each utilizing a different aspect of power and performance.

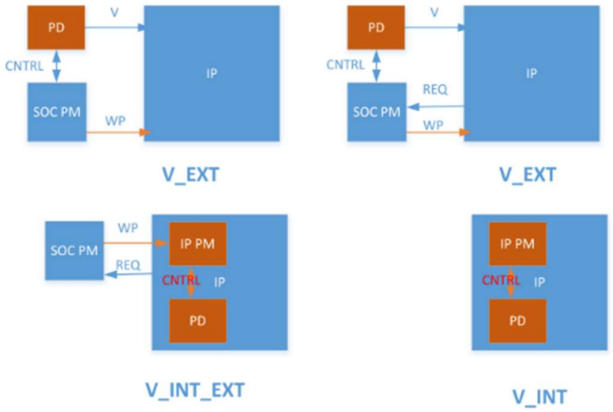

Figure 3.5: Voltage regulation: Int,Ext [6]

C-States (Idle States)

C-states represent processor idle states, where portions of the processor are powered down or put into a low-power mode to save energy during inactivity. These states allow the processor to conserve power when the system is waiting for tasks.

Purpose: Reduce power consumption during idle periods.

Mechanism: Gradually shuts down parts of the processor, such as internal clocks and caches.

Examples:

• C0: Fully active state.

• C1: Halt state with minimal power savings.

• C3: Powering down caches but keeping core context.

• C6: Deep power-down mode with greater latency to wake.

G-States (Global States)

G-states describe the global power and operational states of the entire system as defined by the Advanced Configuration and Power Interface (ACPI). They define how the overall system behaves during power transitions.

Purpose: Manage the entire system's power behavior (e.g., sleep, hibernate).

Mechanism: Controls power delivery to the processor, memory, and peripherals.

Examples:

• G0 (Working): System is fully operational.

• G1 (Sleeping): Low-power state; system can resume quickly.

• G2 (Soft Off): System powered down but can restart.

• G3 (Mechanical Off): System completely shut down

P-States (Performance States)

A power state shows the system's or a particular device's level of power consumption and, consequently, the amount of computational activity. The system's overall power condition is established by the power manager. Device drivers are responsible for configuring each device's power state. System power states and device power states are the two discrete power state sets that are defined by the ACPI specification. The names of the system power states are Sx, where x is a state number ranging from 0 to 5. The names of the device power states are Dx, where x is a state number ranging from 0 to 3. Higher numbered states utilize less power, and the relationship between the two is inverse. States D0 and S0 are the most functional and powerful P states represent processor performance levels. They allow dynamic scaling of voltage and frequency to optimize performance and power consumption during active workloads. Purpose: Optimize energy efficiency under different workload conditions [6].

Mechanism: Adjusts voltage and clock frequency dynamically.

Examples:

• P0: Maximum performance state (highest frequency and power).

• P1-Pn: Successive performance states with reduced voltage and frequency.

Comparison of C-States, G-States, P-States in Tabular Format

Table 3.1: Comparison of C-States, G-States, and P-States

Feature	P-States	C-States(idle)	G- States (Global)
Purpose	Optimize performance and energy efficiency during workloads	Minimize power during idle period	Manage entire system power
Focus	Processor voltage and frequency scaling	Processor cores and caches	Entire system (processor, memory)
State Example	P0, P1, P2, ..., Pn	C0, C1, C3, C6	G0, G1, G2, G3
Activation	Triggered dynamically based on workload	Triggered during idle	Triggered via system state transition
Power Consumption	Higher in P0, lower in Pn	Lowest in deeper states	Varies with system state
Latency	Minimal latency between P-states	Higher latency for deeper idle states	Transition time depends on global state
Scope	Core and processor	Core-level	System-level

3.6 Hybrid Techniques for Static and Dynamic Power Management

Combining techniques for managing both static and dynamic power can yield more comprehensive energy savings. The hybrid approach integrates DVFS and power gating to address power consumption holistically.

Synergistic DVFS and Power Gating

The system can cut down on both dynamic and static power at the same time by using DVFS and power gating together. DVFS lowers the operating frequency and voltage when a component's workload is expected to drop a lot, and then power gating turns off components that aren't being used.

Body Biasing Integration

To further reduce leakage power, body biasing is integrated into the power gating process. Reverse body biasing increases the threshold voltage of transistors in idle components, further reducing leakage currents.

Hardware and Software Co-Design

To work best, the suggested methods need hardware and software to work closely together. Voltage regulators and clock generators are examples of hardware parts that need to be designed to allow for fine-grained control. Software algorithms need to be optimized for quick decision-making.

Hardware Modifications

The hardware changes consist low-power voltage regulators, efficient clock distribution networks, and enhanced power domains to support accurate power control. These changes makes the hardware to respond quickly and efficiently to software directives.

Software Algorithms

The software layer employs machine learning algorithms for workload prediction and decision-making. These algorithms analyse runtime data to select the optimal combination of DVFS settings, power gating states, and sleep modes.

CHAPTER 4

SX FLOWS & RESET STATES & THERMAL POWER MANAGEMENT

4.1 Introduction

Sx flows such as S3, S4, S5 and modern standby is the low power mode. The sleeping states are S1, S2, S3, and S4. When a system is in one of these states, it seems to be off and is not carrying out any computations. As mentioned for each power state below in the System hardware context sections, a sleeping system, in contrast to a shutdown system (S5), maintains memory state, either in RAM or on disk. Rebooting the operating system is not necessary to get the computer back up and running.

Reset flow refers to the sequence of steps taken to verify and validate the behavior of a system or device when it undergoes a reset. Reset flows are essential in ensuring that the system behaves as expected and maintains its functionality and integrity after undergoing a reset event.

4.1.1 ACPI

An industry standard called Advanced Configuration and Power Interface (ACPI) outlines how the peripheral devices, operating system, and hardware of a computer communicate about power consumption. Through ACPI, an operating system may associate drivers with peripherals and adjust to the components and settings of a computer. Additionally, it offers interfaces for power and system management, including auto configuration, status monitoring, and the ability to sleep unwanted hardware.

The ACPI specifies how to enter and exit system sleeping states for the entire machine. Additionally, it offers a universal method for waking the computer from any device.

The power states of motherboard devices, the power planes to which they are attached, and the controls for varying the power states of the devices are all described in ACPI tables. Because of this, the operating system can set devices to low-power modes based on how they use apps. The operating system will employ commands from ACPI to put processors in low-power states. In order to attain a desired balance between performance and energy saving goals as well as other environmental criteria, OSPM will move devices and processors into various performance states, specified by ACPI, while the system is in operation.

4.1.2 OSPM

All power state transitions in devices and systems are managed by the OS under OSPM. The operating system switches devices in and out of low-power states based on user settings and application usage data. It is possible to turn off unused devices. Similar to this, the OS puts the system into a low-power state by utilizing data from user preferences and programs. The OS controls hardware power state transitions through ACPI.

4.2 Power States

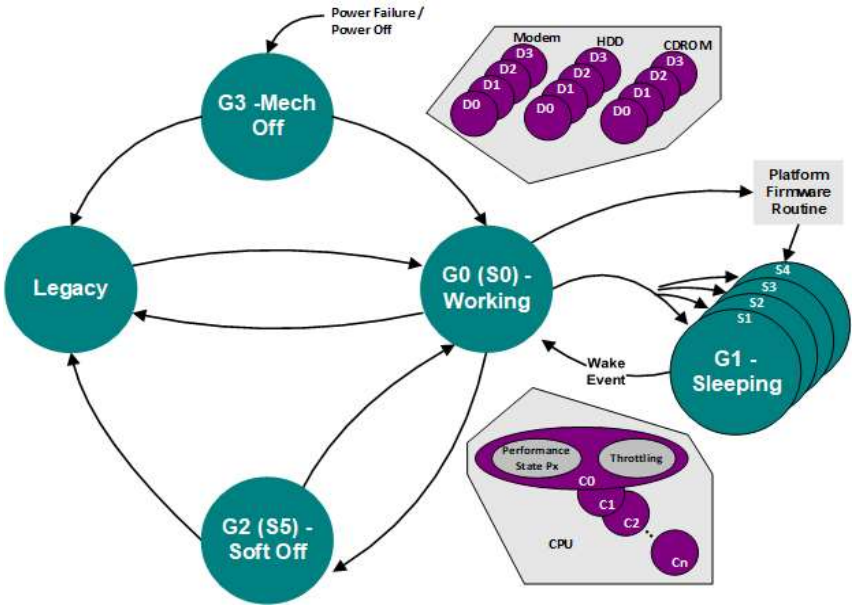

Fig 4.1: System Power states and transition [7]

Working State:

When a computer is in the Working state, it's actively engaged in tasks. User-mode application threads are bustling, ensuring that your computer operates smoothly. Even individual devices aren't idling; they might slip into low-power (Dx) states, and processors may gracefully dip into low-power (Cx) states when they're not in demand. Any device the system deems unnecessary is swiftly powered off, ready to be reawakened with minimal delay. The hallmark of this state is its functionality the entire machine is primed and ready. Yet, within this state, there are nuances. Different sub-states offer varying speeds of computation, power usage, heat generation, and noise levels. The states transitions are shown in in figure 4.1. Tuning within this spectrum becomes a delicate balance of these factors, optimizing for the desired outcome [7].

Sleeping State:

But when the computer finds itself idle or the user signals for repose by pressing the power button, the Operating System (OS) steps in, guiding the system into one of the sleeping (Sx) states. Here, user-visible computation takes a backseat. These sleeping sub-states diverge in terms of the events capable of rousing the system back to life and the time needed for this revival. For scenarios where swift awakening to any event is paramount, the system delves into sub-states that offer a partial reduction in power consumption. However, when only a user-initiated event matters, and time isn't of the essence, the OS can meticulously preserve the system context into an NVS file and usher the hardware into the deep slumber of the S4 state. Here, power consumption plunges to almost zero, yet the system retains its context, poised to awaken even after extended hibernation [7].

Less Common States: Beyond these two primary states lie others, less frequented yet essential in certain contexts. Legacy BIOS power management interfaces usher some computers into the Legacy state, with a subsequent transition to the Working state upon the arrival of an ACPI OS. In contrast, systems devoid of such legacy support, like RISC systems, skip intermediary steps, leaping straight from the Mechanical Off state to the Working state. Users typically guide their machines into the Mechanical Off state by flipping the mechanical switch or unplugging the device altogether, ensuring minimal power consumption when the computer's services are no longer needed [7].

4.2.1 Sleep States (S3)

S3 typically refers to one of the sleeping states defined by the Advanced Configuration and Power Interface (ACPI) standard. ACPI is a set of standards developed to manage power consumption and system resource usage in computers.

S3, also known as the "Standby" state or "Sleep" state, is a low-power state where the computer's state is saved to RAM (random access memory), allowing for a quick wake-up time. When the computer goes into S3 mode, it stops most of its tasks, including those that users can see, but it keeps the system's state in RAM. This lets the computer start up quickly when it needs to because it doesn't have to load the operating system and application data from the disk again.

When it comes to validation, S3 might be useful for checking how well and reliably a system's power management features work. Some validation tasks for S3 might include checking the wake-up time from the S3 state, making sure the system starts up correctly, and making sure that all of the components and peripherals work as they should after waking up from S3. Also, checking that S3 works may include making sure it works with different types of hardware and meets industry standards and government rules [8].

4.2.2 Hibernate State (S4)

Many modern operating systems, like Windows and some Linux distributions, have a power-saving mode called hibernate. When a computer goes into hibernate mode, it saves the data in its RAM (random access memory) to the hard drive or SSD (solid-state drive) and then shuts down completely. This lets the system save power while keeping the operating system, open apps, and any open documents or files in their current state.

Here's how hibernate mode typically works:

Saving State to Disk: Before entering hibernate mode, the operating system writes the contents of RAM to a special file on the hard disk or SSD. This file is called the hibernation file or hiberfil.sys on Windows systems.

Powering Down: Once the state of the system has been saved to disk, the computer powers down completely. This means that all hardware components are turned off, and the system consumes very little to no power.

Resume from Hibernate: When the user powers on the computer again, the operating system reads the saved hibernation file from disk and restores the system's state to what it was before entering hibernate mode. This includes reopening any applications and documents that were open at the time of hibernation.

Hibernate mode is particularly useful for laptops and other portable devices because it allows users to conserve battery power without losing their work. Unlike sleep mode (S3 state), which maintains power to RAM and consumes some power, hibernate mode completely powers down the system while preserving the state. This means that hibernate mode can be used for longer periods of inactivity without draining the battery.

However, it's important to note that entering and exiting hibernate mode may take longer compared to sleep mode, as the system needs to read from and write to the disk to save and restore the state. Additionally, hibernate mode may not be available on all systems or may need to be enabled in the operating system's power settings [8].

4.2.3 Shut down State (S5)

The S5 state, it is also known as the "Soft Off" state or "Power Off" state. It is the lowest power state present in the Advanced Configuration and Power Interface (ACPI) standard. The computer or device is complete off mode when it is in S5, and all system components are in a state of minimum power consumption. Here's how it works:

Closing Processes and Applications: As in shutdown process, the operating system sends signals to running processes and applications, instructing them to gracefully close. So that applications can save any unsaved work and perform necessary cleanup tasks.

Terminating System Services: Background services and system processes are terminated in an orderly manner, just like during a shutdown. This ensures that all system resources are released properly.

Saving System Settings: Certain system settings and configurations may be saved to non-volatile memory, such as the BIOS or UEFI firmware, before entering the S5 state. This allows the system to retain its settings across power cycles.

Shutting Down Hardware: Once all processes and applications have been closed, the operating system sends commands to the hardware components to power off completely. This includes shutting down the CPU, RAM, storage devices, and other peripherals.

Powering Off: Finally, the power to the entire system is cut off, and the computer enters the S5 state. In this state, virtually no power is consumed by the computer, and it remains in a state of minimal energy usage until power is restored or the system is manually powered on by the user.

The S5 state is often used when the computer is not in use for an extended period or when it needs to be completely powered off for maintenance or security reasons. Unlike sleep or hibernate modes, which maintain some level of power to the system for quick wake-up times, the S5 state is a complete shutdown, ensuring that no power is wasted when the computer is not in use [8].

4.2.4 Modern Standby State

Windows 8 and later versions of the Windows operating system added a power management feature

called Modern Standby, which is also known as "Connected Standby" or "Instant Go." It is meant to have a low-power sleep mode that lets you wake up quickly and keeps the network connection and some background tasks going even when the computer is asleep.

In Modern Standby mode:

Fast Resume: The system can quickly wakeup from sleep, similar to traditional sleep modes (S1-S3).

Network Connectivity: Network connection during standing remain active so that device can connect to network.

Background Tasks: Some background task can run in standby mode. Task include operating system updates, VoIP etc. This ensures that essential functions remain available without fully waking up the system.

Low Power Consumption: The system consumes minimal power, similar to traditional sleep modes in modern standby. However, it maintains enough functionality to keep essential tasks active.

Selective Wake: The system can wake up only to do certain things, like getting notifications or syncing data, while keeping other parts in a low-power state.

Modern Standby is great for mobile devices like laptops, tablets, and 2-in-1 devices because it lets them stay connected and responsive even when they are in a low-power sleep state. It strikes a balance between saving power and making things easier for the user, making sure that important tasks can keep going without interruption while extending battery life [8].

4.3 Warm Reset

A warm reset, soft reset, or warm reboot is a way to restart a computer system without turning off all the power to the hardware. A warm reset is different from a cold reset (or hard reset) because the system is turned off and then back on again. With a warm reset, only the operating system and other software parts are restarted, while the hardware stays on.

Initiating the Reset: The user or the software starts the warm reset process. You can do this by using the operating system's restart command, special reset buttons, or keyboard shortcuts.

Operating System Shutdown: The operating system begins the shutdown process, closing all running processes and applications in an orderly manner. This ensures that data is saved and that no data loss or corruption occurs during the reset.

Hardware State Retention: In a cold reset, the hardware is turned off completely, but in a warm reset, the hardware components stay on. This means that the CPU, memory, peripherals, and other hardware will stay the same.

BIOS or UEFI Interaction: The BIOS (Basic Input/Output System) or UEFI (Unified Extensible Firmware Interface) firmware may do some setup work as part of the reset process. This includes getting the hardware ready again and getting the system ready to start up.

Operating System Boot: Once the system has been reset and the hardware is ready, the operating system boot process begins. This involves loading the operating system kernel and initializing system

services and drivers.

System Restart: The last step is for the operating system to restart, and the user sees a new login screen or desktop environment. You can also start any programs or services that are set to run automatically when your computer starts up.

A warm reset is faster than a cold reset because the hardware doesn't have to be turned off and back on again. This means that the operating system and apps can start up faster. This can be helpful when you need to restart your computer quickly, like when you're trying to fix software problems or install system updates. In some cases, though, a cold reset may be needed to fix hardware problems or to completely reset the system [9, 10].

4.4 Cold Reset

A cold reset, which is also called a hard reset or cold boot, is when you turn off all the power to the hardware and then turn it back on again. A cold reset is different from a warm reset because it turns off the whole system and then boots it up again from scratch. In a warm reset, the operating system and software are restarted while the hardware stays on.

Shutting Down the System: The operating system starts the shutdown process, by closing all running processes and applications in an orderly manner. This ensures that data is saved, and that no data loss or corruption happens during the reset.

Hardware Power Off: Once the shutdown of OS is done, the power to the hardware components(CPU, memory (RAM), storage devices (hard drives, SSDs), and peripherals.) is completely cut off.

Hardware Initialization: When the hardware gets power back, the system goes through a process called hardware initialization. This includes the BIOS (Basic Input/Output System) or UEFI (Unified Extensible Firmware Interface) firmware setting up the hardware and running POST (Power-On Self-Test) checks to make sure everything is working right.

Boot Process: Once the hardware is completely initialized, the system starts the boot process. This includes loading the operating system kernel from storage (such as the hard drive or SSD) into memory and initializing system services and drivers.

Operating System Start-up: Once the boot process is complete, the operating system starts up, and the user is presented with a fresh login screen or desktop environment. Any startup programs or services configured to run automatically may also be initiated.

Overall, a cold reset results in a complete restart of the computer system, similar to turning off the power and then turning it back on again. It is often used to troubleshoot hardware or software issues, reset system configurations to default settings, or perform system maintenance tasks. While a cold reset may take longer than a warm reset due to the need to initialize hardware components, it can help resolve certain types of problems and ensure a clean start for the system [9, 10].

4.5 Thermal Management

Semiconductor technology forms the backbone of modern electronics, powering everything fromSmartphones and computers to advanced medical devices and automative systems. At the heart of

These technologies are semiconductor devices, primarily integrated circuits (ICs), which have theRevolutionized the way we process and manage information. The relentless pursuit of Moore's Law has led to an exponential increase in the number of transistors on a chip, enhancing performance but also indroducing significant thermal challenges.

As semiconductor devices become more powerful and compact, managing the heat they generatebecomes crucial. Excessive heat can lead to performance degradation, reduced reliability, and even catastrophic failure of electronic components. Therefore, effective thermal management is essential to ensure the optimal performance and longevity of semiconductor devices. This involves not only the design of efficient cooling solutions but also the integration of thermal considerations into the entire lifecycle of semiconductor development, from design to post-silicon validation.

4.5.1 Importance of Thernal Management

Thermal management is very important for post-silicon validation. When chips are tested, they need to work within safe temperature ranges to get accurate results and avoid damage. This needs advanced tools and strategies for managing heat to keep an eye on and control the temperature during testing. Post-silicon validation gives us important information that helps us find the best design flow for performance.

Managing heat is an important part of designing and using semiconductor devices. As semiconductor technology improves, devices get stronger and smaller, which means they generate more heat and power. There are a number of reasons why good thermal management is important:

Performance Optimization: Semiconductor devices like processors and memory chips which work best at certain temperature. When devices get too hot, to decrease the temperature they used thermal throttling. This process cools down the system but decreases the overall performance of electronic systems.

Reliability and Longevity: High temperatures can increases wear and tear on semiconductor components, leading to premature failure. Thermal management ensures that devices operate within safe temperature limits, enhancing their reliability and extending their operational lifespan.

Preventing Thermal Runaway: When devices get too hot it stuck in feedback loop named thermal runaway. This means temperatures rises, which makes more heat that can cause more failures. Thermal Management prevents such situation.

Energy Efficiency: Good thermal management can make semiconductor devices energy saviours. Device work mor efficiently by usinh less energy and make the running cost also low.

Compliance with Safety Standards: There are some safety standards set by industry and international institutes which limits the maximum heat that can be generated. Thermal management keep these standard in mind so that device as well as end user can be protected [12].

4.5.2 Impact of Temperature on Device Performance and Reliability

Temperature has a high impact on the performance and reliability of semiconductor devices. The effects of temperature can be observed in several key areas:

Electrical Characteristics: Temperature changes the kinetic energy of carriers that will eventually changes the mobility and also it can change threshold voltage. The changes in these specifications can degrade the device performance.

Signal Integrity: High temperatures can cause problems with signal integrity, like more noise and crosstalk, which can make data transmission within a device less reliable. This is especially important in digital circuits that work at high speeds.

Material Degradation: Long term exposure to high temperatures can cause damage the physical properties in semiconductor devices, like solder joints and dielectric layers. This can lead to mechanical failures and reduced device reliability.

Electromigration: Incresed temperatures can increase electromigration, in which metal atoms in interconnects migrate due to high current densities. This can lead to open circuits and device failure over time.

Thermal Stress: High variation in temperature can make thermal stress, causing mechanical deformation and failure of semiconductor components. This become prominent in applications with frequent power cycling.

4.6 Thermal Management Techniques

Effective thermal management is crucial for maintaining the performance and reliability of semiconductor devices. Various techniques are employed to manage heat generation and dissipation, each with its own advantages and limitations.

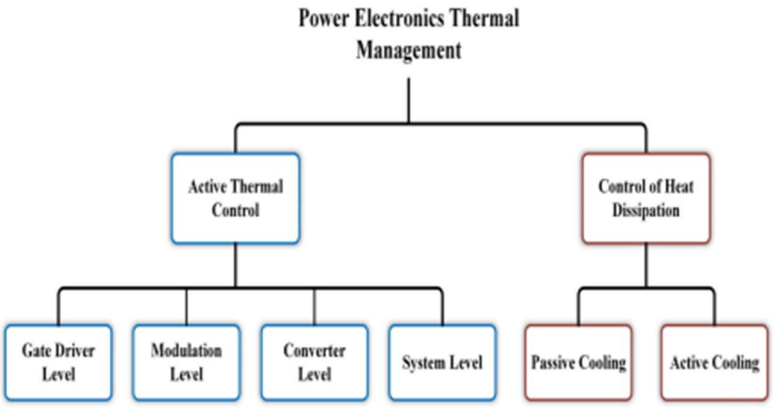

Figure 4.2 Thermal management approaches [12]

4.6.1 Passive Cooling

Passive cooling techniques rely on natural heat dissipation methods without the use of external energy sources. They are often preferred for their simplicity and reliability.

Heat sinks are metal components, typically made of aluminum or copper, that are attached to

semiconductor devices to increase the surface area for heat dissipation. They work by conducting heat away from the device and dissipating it into the surrounding air. Heat sinks are widely used due to their low cost and effectiveness in moderate heat dissipation scenarios.

Thermal Interface Materials (TIMs): TIMs are materials placed between the semiconductor device and the heat sink to improve thermal conductivity. They fill microscopic air gaps and ensure efficient heat transfer. Common TIMs include thermal pastes, pads, and adhesives.

4.6.2 Active Cooling

Active cooling techniques involve the use of external energy sources to enhance heat dissipation. They are often used in high-performance applications where passive cooling is insufficient.

1 Fans: Fans are used to increase airflow over heat sinks, enhancing convective heat transfer. They are commonly used in computers and other electronic devices to maintain optimal operating temperatures. Fans are effective but can introduce noise and consume additional power. The figure 4.3 shows the Fan using on RVP board.

2 Liquid Cooling: Liquid cooling systems use a liquid coolant to absorb heat from the semiconductor device and transfer it to a radiator, where it is dissipated into the air. Liquid cooling is highly effective for high-power applications, such as gaming PCs and data centers, but it is more complex and expensive than air cooling.

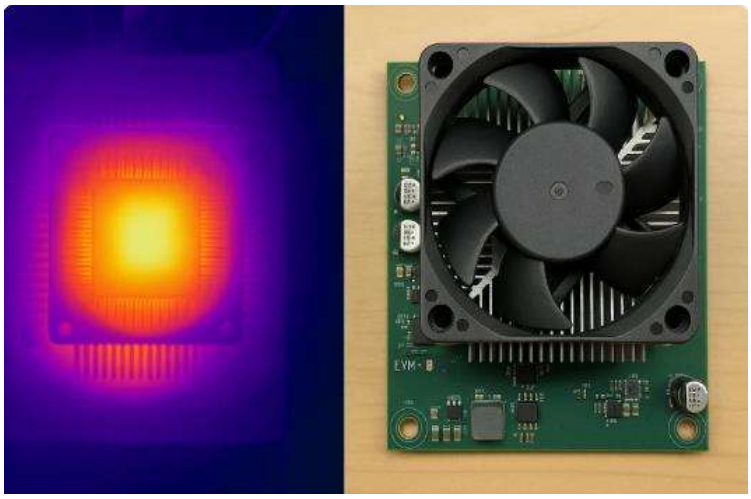

Figure 4.3 : Thermal Head and Fan [13]

4.6.3 Advanced Techniques

Advanced thermal management techniques are employed in specialized applications where traditional methods may not suffice.

Thermoelectric Cooling: Thermoelectric coolers (TECs) use the Peltier effect to create a temperature difference by applying an electric current. They can actively cool semiconductor devices by transferring heat from one side of the TEC to the other. TECs are compact and can provide precise temperature

control, but they are less efficient than other cooling methods.

Phase Change Materials (PCMs): During phase changes (like going from solid to liquid), PCMs take in and let out heat at certain temperatures. They can help keep the temperature from getting too high by taking in extra heat and letting it out slowly over time. PCMs are helpful in situations where there are short bursts of high heat, but they need to be carefully added to the thermal management system.

4.7 Effectiveness of the Different thermal Techniques

Efficiency: Passive methods are not good at getting rid of heat as liquid cooling agent and more advanced methods like thermoelectric cooling. But these methods are expensive and complex

Cost: Advance methods are good but expensive whereas passive method like heat sink are simple and cheap but not that much effective.

Complexity: Active and advanced cooling syatem are complex and hard to manage but they give better result than passive.

Noise and Power Consumption: Passive cooling is silent and consumes no additional power, while active cooling methods, especially those involving fans, can introduce noise and increase power consumption.

Application Suitability: The type of thermal management method you choose will depend on the needs of your application, such as power density, space limitations, and budget. High-performance and important applications may require the use of more advanced and expensive solutions [13].

CHAPTER 5

IMPLEMENTATION

This chapter talks about the planned way that the suggested power management methods will be put into action and tested. The method includes using simulations to test things, looking at workloads, and combining dynamic and static power management strategies. The steps are designed to ensure thorough testing and reliable results.

5.1 Framework for Power Management

The framework has three main parts: a dynamic power management module, a static power reduction module, and a system for keeping an eye on workloads. These parts work together to make the SoC use the least amount of energy possible in different situations.

5.1.1 Dynamic Power Management Module

The dynamic power management module uses an improved DVFS mechanism to change voltage and frequency based on the workload at the time. This module has a control system that can make changes based on workload history and runtime analysis.

5.1.2 Static Power Reduction Module

The static power reduction module uses power gating and reverse body biasing to cut down on leakage currents in parts that aren't being used. The module makes sure that parts of the SoC that aren't being used are powered down in a way that doesn't slow down the system.

5.1.3 Workload Monitoring System

The workload monitoring system keeps track of how much memory, processors, and peripherals are being used. Machine learning algorithms process this data to make predictions about future workload trends, which lets power management decisions be made ahead of time.

5.2 Implementation Workflow

There are several steps in the implementation workflow, such as design, simulation, testing, and validation. Each step makes sure that the suggested methods are strong and useful in the real world.

5.2.1 Design Stage

During the design phase, the SoC architecture is modeled to work with DVFS, power gating, and other suggested methods. Power domains and clock networks are made to give you precise control over performance and power.

5.2.2 Simulation and Testing

The simulation environment shows how the SoC would act in different workload situations. To test, you run synthetic benchmarks and real-world applications to see how much power they use, how fast they work, and how quickly they respond.

5.2.3 Validation and Analysis

Validation means checking the results of the simulation environment against theoretical predictions and current power management methods. The analysis is mostly about finding ways to make things better

and making sure that the suggested methods work as planned.

5.3 Evaluation Metrics

The effectiveness of the proposed power management techniques is evaluated using serval key metrics:

• Power Consumption: Total power consumed by the SoC, including both dynamic and static components.

• Performance: Execution time, throughput, and latency of the system under various workloads

• Energy Efficiency: Energy consumption per operation, calculated as power consumption over time.

• Scalability: Ability of the techniques to handle increased workload complexity and multi-core configurations.

5.4 Role of Parametric Script in SoC Design Flow

The script is tested in post-Silicon environment with real life hardware IPs and SUT with temperature-controlled environment. The Silicon tested is a revised Silicon with corrections done from previous debugs and design flaws.

The scripts can act as a link between Pre-Silicon and Post-Silicon validation imitating the parameters predicted by the assumptions and calculations done by Pre-Silicon validation. Script reads the parameters from the Pre-Silicon team and overrides the values read on a Post-Silicon SoC under controlled environment, thus duplicating a Post -Silicon SoC which mimics all the features and parameters of a Pre-Silicon environment.

With this script, pre-defined workloads can be tested in the SUT mimicking the Pre-Silicon parameters and giving output results of power consumption of Pre-Silicon parameters in a SoC under real-life environment.

5.4.1 Bare Metal Content for Pre Silicon Validation

Bare metal scripts are essential tools in post-silicon validation, used to test and validate the functionality of hardware components without the interference of an operating system. These scripts are crucial for ensuring that the hardware behaves as expected under various conditions. Purpose Bare metal scripts are designed to directly interact with hardware components, bypassing the need for an operating system. This allows to perform low-level testing and validation of silicon chips, ensuring that they meet design specifications and function correctly under various conditions.

Components of a Bare-Metal Environment

Application code: The main application logic, which directly controls hardware operations

Startup Code: Initialization routines that set up the hardware, including setting up the stack, configuring memory, and initializing peripherals.

Device Drivers: Drivers to control and communicate with hardware components like GPIO, UART, SPI, I2C, etc.

Libraries: Utility libraries that provide essential functions (e.g., mathematical operations, string handling) without relying on OS services.

Debugging and Monitoring Tools: Tools like JTAG, SWD, or serial debuggers to monitor and debug

the application directly on the hardware.

5.5 Experimental Setup

The experiments are conducted in a controlled simulation environment. The setup includes:

• Processor Model: A custom-designed SoC with support for DVFS, power gating, and body biasing.

• Workload Scenarios: Synthetic benchmarks, such as SPEC CPU, as well as real-world applications, to simulate diverse operating conditions.

• Measurement Tools: Power analyzers and performance profiling tools to measure energy consumption and system responsiveness.

5.6 OUTPUT AND ANALYSIS

5.6.1 Active and Thermal

Objective To evaluate the system's performance and thermal management capabilities by observing how the CPU frequency scales with workload intensity and how the system responds to thermal events. Background Modern processors dynamically adjust their operating frequency to balance performance and power consumption. Under heavy workloads, the CPU frequency may increase to provide more computational power. However, this can lead to increased heat generation. If the system's cooling mechanisms are insufficient to dissipate this heat, a thermal event may occur, triggering thermal throttling to protect the hardware.

Test Setup

- Hardware: A computer system with a multi-core CPU and adequate cooling solutions (e.g., fans, heat sinks).
- Software: Monitoring tools to track CPU frequency, temperature, and workload (e.g., HW Monitor, CPU-Z).
- Workload: A stress-testing application capable of pushing the CPU to its maximum frequency

Test Procedure

1. Baseline Measurement:
    - Start with the system at idle and record the baseline CPU frequency and temperature.
2. Initiate Workload:
    - Begin the stress test application to simulate a high-intensity workload.
    - Monitor and record the CPU frequency and temperature at regular intervals (e.g., every 10 seconds).
3. Observe Frequency Scaling:
    - As the workload increases, observe the CPU frequency scaling up to meet the performance demand.
    - Note the maximum frequency achieved before any thermal event occurs.
4. Thermal Event Detection:
    - Continue monitoring until the CPU temperature approaches the thermal threshold set by the manufacturer.

- Record the point at which thermal throttling is triggered, causing the CPU frequency to decrease.

5. Frequency Reduction:
    - Observe the reduction in CPU frequency as the system attempts to manage the heat.
    - Record the new stabilized frequency and temperature after throttling.
6. Recovery:
    - Once the workload is reduced or stopped, monitor the system as it cools down.
    - Record the time taken for the CPU frequency to return to baseline levels.

Results

- As observed in fig 5.1 CPU frequency should initially increase with the workload to provide maximum performance.
- Upon reaching a critical temperature, a thermal event should trigger a reduction in frequency to prevent overheating.
- The system should stabilize at a lower frequency until the temperature decreases.
- Once the system cools, the frequency should return to normal operating levels.

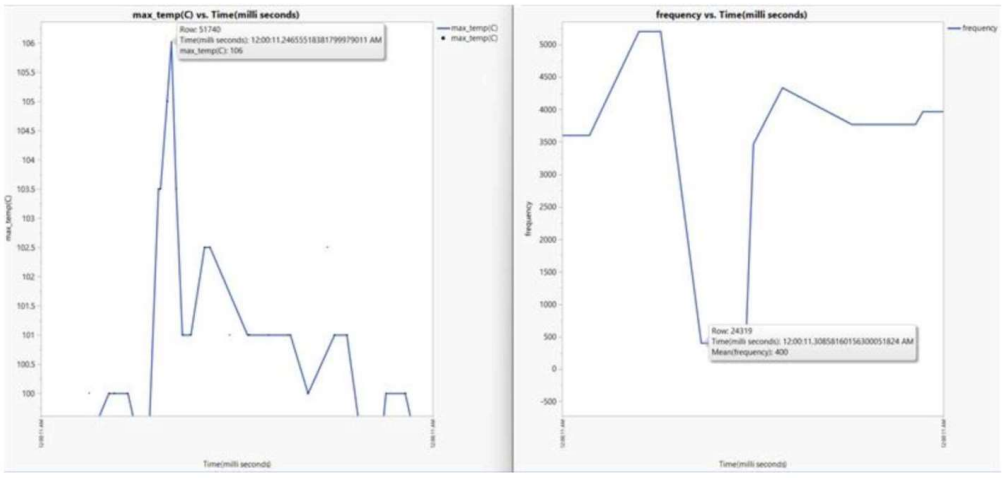

Figure 5.1 : Frequency/Temperature vs time

Conclusion This test scenario demonstrates the balance between performance and thermal management in modern CPUs. It highlights the importance of effective cooling solutions to maintain optimal performance without triggering thermal throttling.

5.6.2 Thermal Power Management and CPU Shutdown

Objective To evaluate the system's thermal power management capabilities by testing its response when the CPU reaches its maximum junction temperature (Tjmax), resulting in a shutdown and global reset.

Background The maximum junction temperature (Tjmax) is the highest temperature a CPU can safely

operate at. Exceeding this temperature can lead to hardware damage. Therefore, systems are designed to shut down or reset when Tjmax is reached to prevent overheating and ensure safety.

Test Setup

- Hardware: A computer system with a CPU capable of thermal monitoring and management.
- Software: Monitoring tools to track CPU temperature and system status (e.g., Monitor, CPU-Z).
- Cooling: Ensure that cooling solutions (e.g., fans, heat sinks) are in place but may be intentionally limited to reach Tjmax.
- Workload: A stress-testing application capable of pushing the CPU to its thermal limits .

Test Procedure

1. Baseline Measurement:
    - Start with the system at idle and record the baseline CPU temperature and frequency.
2. Set Maximum Temperature:
    - Configure the system's thermal management settings to allow the CPU to reach up to 100°C (Tjmax).
3. Initiate Workload:
    - Begin the stress test application to simulate a high-intensity workload.
    - Monitor and record the CPU temperature at regular intervals (e.g., every 10 seconds).
4. Observe Temperature Increase:
    - As the workload increases, observe the CPU temperature rising towards Tjmax.
    - Note the temperature at which the system begins to throttle, if applicable.
5. Reach Tjmax:
    - Continue monitoring until the CPU temperature reaches 100°C (Tjmax).
    - Record the exact temperature and time when Tjmax is reached.
6. CPU Shutdown and Global Reset:
    - Observe the system's response as it reaches Tjmax. The CPU should shut down to prevent damage.
    - Confirm that a global reset occurs, restarting the system automatically.
7. Post-Reset Monitoring:
    - After the reset, monitor the system to ensure it returns to normal operating conditions.
    - Record any error messages or logs generated during the reset process.

Results

- The CPU temperature should steadily increase under load until it reaches Tjmax.
- Upon reaching Tjmax, the system should automatically shut down the CPU to prevent overheating.
- A global reset should occur, restarting the system without manual intervention.
- The system should return to normal operation after the reset, with no persistent errors.

```
>>> pm_tools.thermal.sample_max_temp(100)
Displaying max temperatures consecutively, press Enter to stop
```

#	CORE0	CORE1	CORE2	CORE3	CORE4	CORE5	CORE6	CORE7	CORE8	CORE9	CORE10
1	118.0°	116.5°	117.5°	117.5°	117.5°	120.5°	120.0°	119.0°	119.0°	119.0°	119.0°
2	118.0°	116.5°	119.0°	119.0°	117.5°	119.0°	120.0°	119.0°	119.0°	119.0°	119.0°
3	118.0°	117.5°	119.0°	119.0°	119.0°	120.5°	120.0°	119.0°	119.0°	120.5°	119.0°
4	118.0°	117.5°	119.0°	119.0°	119.0°	120.5°	120.0°	120.5°	120.5°	120.5°	120.5°
5	119.0°	117.5°	119.0°	120.5°	119.0°	120.5°	120.0°	119.0°	120.5°	119.0°	120.5°
6	119.5°	117.5°	119.0°	120.5°	117.5°	120.5°	121.0°	119.0°	120.5°	119.0°	120.5°
7	118.0°	117.5°	119.0°	120.5°	119.0°	121.5°	120.0°	120.5°	120.5°	120.5°	119.0°
8	119.0°	117.5°	119.0°	120.5°	119.0°	120.5°	120.0°	120.5°	120.5°	120.5°	120.5°
9	119.0°	116.5°	119.0°	120.5°	117.5°	121.5°	120.0°	120.5°	119.0°	120.5°	120.5°
10	119.5°	119.0°	119.0°	120.5°	119.0°	123.0°	121.5°	120.5°	120.5°	120.5°	120.5°
11	119.5°	119.0°	119.0°	120.5°	119.0°	120.5°	121.5°	120.5°	120.5°	120.5°	121.5°

```
TargetEvent: PowerDomain : CPU : Off    -- 19:41:45.938647 2025-03-11
Error running sample_max_temp:Internal_Error == 0x8000000f
.       Failed to transition precondition 'PackageAwake' to state 'true' for device 'RPL_CLTAP0'.  Unable to wake up package.
>>>
TargetEvent: Reset : Started            -- 19:41:46.149085 2025-03-11
>>>
>>> unlock
could not authorize RPL_CLTAP0 to level=Red due to
        Authorization_Level_Not_Supported == 0x80140004
.       Device 'RPL_CLTAP0' (0x00003001) does not support authorization level 'Red'
ADP0 unlocked to Red
```

Figure 5.2 System behavior at Max temp

```
>>> pch.pmc.pmu.gblrst_cause_0.show
0x00000000 : pmc_rf_fusa_err (24:24) (ro/v) -- Bit PMC RF FUSA Error Global Reset is set by HW on a global reset. Only one bit is set for each global re
0x00000000 : pmc_irom_parity (23:23) (ro/v) -- Bit PMC I-ROM Parity Error Global Reset is set by HW on a global reset. Only one bit is set for each glob
0x00000000 : pmc_sram_unc_err (22:22) (ro/v) -- Bit PMC SRAM Uncorrectable Error Global Reset is set by HW on a global reset. Only one bit is set for ea
0x00000000 : cse_hec_unc_err (21:21) (ro/v) -- Bit CSE Uncorrectable HEC Error Global Reset is set by HW on a global reset. Only one bit is set for each
0x00000000 : ocwdt_icc (20:20) (ro/v) -- Bit Over-Clocking WDT Expiration In ICC Survivability Mode is set by HW on a global reset. Only one bit is set
0x00000000 : ocwdt_noicc (19:19) (ro/v) -- Bit Over-Clocking WDT Expiration In Non-ICC Survivability Mode is set by HW on a global reset. Only one bit
0x00000000 : adr_gpio (18:18) (ro/v) -- Bit ADR GPIO Reset is set by HW on a global reset. Only one bit is set for each global reset event.
0x00000000 : me_unc_err (17:17) (ro/v) -- Bit ME HW Uncorrectable Error is set by HW on a global reset. Only one bit is set for each global reset event.
0x00000000 : cpu_thrm_wdt (16:16) (ro/v) -- Bit CPU Thermal Runaway Watchdog Timer is set by HW on a global reset. Only one bit is set for each global
0x00000000 : mia_ux_err (15:15) (ro/v) -- Bit Minute IA Unexpected Error is set by HW on a global reset. Only one bit is set for each global reset event
0x00000000 : mia_uxs_err (14:14) (ro/v) -- Bit Minute IA Unexpected Shutdown Error is set by HW on a global reset. Only one bit is set for each global
0x00000000 : ish (13:13) (ro/v) -- Bit ISH requested Global Reset signal 0 is set by HW on a global reset. Only one bit is set for each global reset eve
0x00000000 : syspwr_flr (12:12) (ro/v) -- Bit SYS_PWROK Failure is set by HW on a global reset. Only one bit is set for each global reset event.
0x00000000 : pchpwr_flr (11:11) (ro/v) -- Bit PCH_PWROK Failure is set by HW on a global reset. Only one bit is set for each global reset event.
0x00000000 : pmc_fw (10:10) (ro/v) -- Bit PMC Firmware Global Reset is set by HW on a global reset. Only one bit is set for each global reset event.
0x00000000 : me_wdt (09:09) (ro/v) -- Bit ME Firmware Watchdog Timer is set by HW on a global reset. Only one bit is set for each global reset event.
0x00000000 : pmc_wdt (08:08) (ro/v) -- Bit PMC Firmware Watchdog Timer is set by HW on a global reset. Only one bit is set for each global reset event.
0x00000000 : lt_reset (07:07) (ro/v) -- Bit LTRESET# With Policy 1 is set by HW on a global reset. Only one bit is set for each global reset event.
0x00000000 : megbl (06:06) (ro/v) -- Bit ME-Initiated Global Reset is set by HW on a global reset. Only one bit is set for each global reset event.
0x00000001 : cpu_thrm (05:05) (ro/v) -- Bit CPU Thermal Trip is set by HW on a global reset. Only one bit is set for each global reset event.
0x00000000 : me_pbo (04:04) (ro/v) -- Bit ME-Initiated Power Button Override is set by HW on a global reset. Only one bit is set for each global reset
0x00000000 : pch_thrm (03:03) (ro/v) -- Bit PCH Internal Thermal Trip is set by HW on a global reset. Only one bit is set for each global reset event.
0x00000000 : pmc_unc_err (02:02) (ro/v) -- Bit PMC SUS RAM Uncorrectable Error is set by HW on a global reset. Only one bit is set for each global reset
0x00000000 : pbo (01:01) (ro/v) -- Bit power button override is set by HW on a global reset. Only one bit is set for each global reset event.
```

Figure 5.3 Global reset output

Conclusion This test scenario demonstrates the system's ability to manage thermal events by shutting down and resetting when Tjmax is reached. It highlights the importance of effective thermal management to ensure system safety and reliability.

5.6.3 Sx and Reset

After bringing up the system with latest operating system and IFWI, execute the cycles of Sx flows such as S3 (sleep), S4 (hibernate), S5 (shut down) and reset flows such as warm reset, cold reset on tool. After execution of cycles, check that the cycles passed or failed. If the cycles failed then check the logs that which type of failure like soft hang or hard hang. After diagnosing the type of failure, resolve that failure by debugging it. Logic Analyzer and Oscilloscope is used for debugging the failure by tracing the signals.

After bringing the system with the latest operating system and Intel Firmware Interface (IFWI), you need to perform several power and reset cycle tests to ensure system stability and reliability. The specific cycles to execute include various states of power management and resets, such as:

S3 (Sleep) Cycle: This cycle involves putting the system into a low-power sleep state where the system context is maintained in RAM while most other components are powered down. This test verifies the

system's ability to enter and resume from sleep mode successfully.

S4 (Hibernate) Cycle: In this cycle, the system saves the contents of RAM to the disk and then powers down completely. This test ensures that the system can successfully enter hibernate mode and later restore the previous state from the disk upon waking.

S5 (Shut Down) Cycle: This involves completely powering off the system. The test checks the system's ability to shut down properly and power back up, ensuring there are no issues during a full system restart.

Warm Reset: A warm reset is a soft reboot that restarts the system without turning off the power entirely. This test verifies that the system can restart without a full power cycle, maintaining certain states or data.

Cold Reset: This reset involves turning the power off and then back on, similar to a complete shutdown and power-up cycle. It checks the system's ability to recover from a full power-down state.

After executing these cycles, you need to verify whether they passed or failed. If any of the cycles fail, you should examine the logs to determine the nature of the failure. Failures can generally be categorized into two types:

Soft Hang: In this scenario, the system becomes unresponsive but is still powered on. This can happen due to software or minor hardware issues that do not cause the system to lose power but prevent it from operating correctly.

Hard Hang: Here, the system is completely unresponsive and often requires a power cycle to recover. This type of failure is usually more severe and can be indicative of significant hardware or critical software problems.

5.6.4 Test Result for Warm Reset

```
2025-06-20 11:12:09.258,Solar,,,,,,,,Config loop: 1 Launching Test:GSTATES,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
2025-06-20 11:12:11.231,GSTATES,2,,,,,Warm-Reset,,System is out of Reset  ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
2025-06-20 11:12:11.236,GSTATES,2,,,,,Warm-Reset,,Host Partition Reset Cause 0 = 0x2,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
2025-06-20 11:12:11.236,GSTATES,2,,,,,Warm-Reset,,Return from Warm-Reset: 11:12:11 AM  Elapsed Time:167 seconds,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
2025-06-20 11:13:11.288,GSTATES,3,,,,,Warm-Reset,,   Attempt at: 11:13:11 AM,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
2025-06-20 11:13:11.288,GSTATES,3,,,,,Warm-Reset,,XmlFile: C:\applications\solar3\3.24.334.0\SolarTemp.xml,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
2025-06-20 11:15:56.814,Solar,,,,,,,,GBLRST0 = 0x0;GBLRST1 = 0x0;HPRST0 = 0x2(CF9_ES; ),,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
2025-06-20 11:15:56.833,Solar,,,,,,,,Config loop: 1 Launching Test:GSTATES,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
2025-06-20 11:15:59.022,GSTATES,3,,,,,Warm-Reset,,System is out of Reset  ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
2025-06-20 11:15:59.027,GSTATES,3,,,,,Warm-Reset,,Host Partition Reset Cause 0 = 0x2,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
2025-06-20 11:15:59.027,GSTATES,3,,,,,Warm-Reset,,Return from Warm-Reset: 11:15:59 AM  Elapsed Time:168 seconds,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
2025-06-20 11:16:59.101,GSTATES,4,,,,,Warm-Reset,,   Attempt at: 11:16:59 AM,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
2025-06-20 11:16:59.101,GSTATES,4,,,,,Warm-Reset,,XmlFile: C:\applications\solar3\3.24.334.0\SolarTemp.xml,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
2025-06-20 11:19:39.806,Solar,,,,,,,,GBLRST0 = 0x0;GBLRST1 = 0x0;HPRST0 = 0x2(CF9_ES; ),,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
2025-06-20 11:19:39.821,Solar,,,,,,,,Config loop: 1 Launching Test:GSTATES,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
2025-06-20 11:19:41.954,GSTATES,4,,,,,Warm-Reset,,System is out of Reset  ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
2025-06-20 11:19:41.961,GS,5,,,,Warm-Reset,,XmlFile: C:\applications\solar3\3.24.334. 0 = 0x2,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
2025-06-20 11:19:41.961,GSTATES,4,,,,,Warm-Reset,,Return from Warm-Reset: 11:19:41 AM  Elapsed Time:162 seconds,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
2025-06-20 11:20:42.013,GSTATES,5,,,,,Warm-Reset,,   Attempt at: 11:20:42 AM,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
2025-06-20 11:20:42.013,GSTATES,5,,,,,Warm-Reset,,XmlFile: C:\applications\solar3\3.24.334.0\SolarTemp.xml,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
2025-06-20 11:23:30.386,Solar,,,,,,,,GBLRST0 = 0x0;GBLRST1 = 0x0;HPRST0 = 0x2(CF9_ES; ),,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
2025-06-20 11:23:30.411,Solar,,,,,,,,Config loop: 1 Launching Test:GSTATES,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
2025-06-20 11:23:32.435,GSTATES,5,,,,,Warm-Reset,,System is out of Reset  ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
2025-06-20 11:23:32.435,GSTATES,5,,,,,Warm-Reset,,H ,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
2025-06-20 11:23:32.435,GSTATES,5,,,,,Warm-Reset,,Return from Warm-Reset: 11:23:32 AM  Elapsed Time:170 seconds,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
2025-06-20 11:23:32.435,GSTATES,5,,,,,Warm-Reset,,GBLRST0 = 0x0; GBLRST1 = 0x0;   HPRST0 = 0x2; ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
2025-06-20 11:23:32.435,,,,,,,,,Test End - End Date: 20-Jun-2025 11:23:32 AM Test Summary   Warm Reset: 5     Maximum Allowed Failures:0    G-State End:
Failure Count:0    G-State End: PASS ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
2025-06-20 11:23:32.483,GSTATES,,,,,,,,End G-State test,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
Time,TestType,Iter,Affinity,Scope,Result,FailCount,Request Status Notes freq(MHz) ClkMod PC2 PC2% PC6 PC6% PC6.1 PC6.1% PC6.2 PC6.2%,PC10,PC10%,PC10.1,PC10.1
%,PC10.2,PC10.2%,PC10.3,PC10.3%,CLC2,CLC2%,CLC3,CLC3%,CLC6,CLC6%,MC6,MC6%,CC1,CC1%,CC3,CC3%,CC6,CC6%,CC7,CC7%,C0E,C0E%,GT-RC6,GT-RC6%,GT-BW,GT-BW%,GT-SR,GT-
SR%,ClipCause, Level
2025-06-20 11:23:33.591,Solar,,,,,,,,SOLAR END RESULTS,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
2025-06-20 11:23:33.591,Solar,,,,,,,,GSTATES finished with result: Pass EC: 0x0,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
2025-06-20 11:23:33.591,CONFIG,,,,,,,,Restore PM Configuration[Begin],,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
2025-06-20 11:23:33.591,CONFIG,,,,,,,,Restore PM Configuration[End],,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
```

Figure 5.4 : Warm Reset output

The test output 5.4 shows timestamps, test names, statuses, and pass/fail information.

The log starts with a configuration loop, then a warm reset is initiated. The warm reset appears to be

successful based on the message "Host Partition Reset Cause 0 0x2".

Following the reset, several GSTATE tests are carried out. These tests seem to be successful as well. The log concludes with entries showing the restoration of the PM configuration and the ending of the solar test with a pass result.

Conclusion: The log suggests that the computer system underwent a successful test.

5.6.5 Test Result of S4 (Hibernate)

```
C10.1,PC10.1%,PC10.2,PC10.2%,PC10.3,PC10.3%,CLC2,CLC2%,CLC3,CLC3%,CLC6,CLC6%,MC6,MC6%,CC1,CC1%,CC3,CC3%,CC6,CC6%,CC7,CC7%,C0E,C0E%,GT-RC6,GT-
RC6%,GT-BW,GT-BW%,GT-SR,GT-SR%,ClipCause, Level
2025-06-20 10:26:46.819,Solar,,,,,,,,Starting config loop: 1,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
2025-06-20 10:26:46.819,CONFIG,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L3
2025-06-20 10:26:46.819,Solar,,,,,,,,Config loop: 1 Launching Test:GSTATES,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
2025-06-20 10:26:46.901,,,,,,,,,Test Config:     Random[no],,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
2025-06-20 10:26:46.901,,,,,,,,,Executing cycles: 5 Failure Tolerance: 0%,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
2025-06-20 10:26:47.004,GSTATES,1,,,,,S4,,Attempt at: 10:26:47 AM,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
2025-06-20 10:28:05.330,GSTATES,1,,,,,S4,,   Return from S4: 10:28:05 AM  Elapsed Time:78 seconds  Wake Event: Type: Presume Wake Timer from
Solar,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
2025-06-20 10:29:05.331,GSTATES,2,,,,,S4,,Attempt at: 10:29:05 AM,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
2025-06-20 10:30:24.276,GSTATES,2,,,,,S4,,   Return from S4: 10:30:24 AM  Elapsed Time:79 seconds  Wake Event: Type: Presume Wake Timer from
Solar,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
2025-06-20 10:31:24.284,GSTATES,3,,,,,S4,,Attempt at: 10:31:24 AM,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
2025-06-20 10:32:42.298,GSTATES,3,,,,,S4,,   Return from S4: 10:32:42 AM  Elapsed Time:78 seconds  Wake Event: Type: Presume Wake Timer from
Solar,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
2025-06-20 10:33:42.302,GSTATES,4,,,,,S4,,Attempt at: 10:33:42 AM,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
2025-06-20 10:35:00.286,GSTATES,4,,,,,S4,,   Return from S4: 10:35:00 AM  Elapsed Time:78 seconds  Wake Event: Type: Presume Wake Timer from
Solar,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
2025-06-20 10:36:00.304,GSTATES,5,,,,,S4,,Attempt at: 10:36:00 AM,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
2025-06-20 10:37:19.307,GSTATES,5,,,,,S4,,   Return from S4: 10:37:19 AM  Elapsed Time:79 seconds  Wake Event: Type: Presume Wake Timer from
Solar,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
2025-06-20 10:37:19.307,GSTATES,5,,,,,S4,,GBLRST0 = 0x0; GBLRST1 = 0x0;   HPRST0 = 0x2; ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
2025-06-20 10:37:19.307,,,,,,,,,Test End - End Date: 20-Jun-2025 10:37:19 AM Test Summary   S4: 5     Maximum Allowed Failures:0    G-State End:
Failure Count:0    G-State End: PASS ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
2025-06-20 10:37:19.354,GSTATES,,,,,,,,End G-State test,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
2025-06-20 10:37:19.360,Solar,,,,,,,,SOLAR END RESULTS,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
2025-06-20 10:37:19.360,Solar,,,,,,,,GSTATES finished with result: Pass EC: 0x0,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
2025-06-20 10:37:19.360,CONFIG,,,,,,,,Restore PM Configuration[Begin],,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
2025-06-20 10:37:19.360,CONFIG,,,,,,,,Restore PM Configuration[End],,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,L2
```

Figure 5.5 Hibernate output

Conclusion: This Test result shows that cycles of S4 has been triggered and test is successfully passed.

5.7 REQ TREE AUTOMATION

The automation is targeted to check the requirements and sub requirements of the states like G-States.

5.7.1 Start of Automation

• The first step is to check whether the PLL is available for a particular project in the specification file provided by the architect team.

• Next, we form an equation that will help us identify the required signals from the Verdi GUI. • This equation is passed to the trace function, which provides output in the form of a list.

• This output list, denoted as "X", is used to trace the next level of signals leading to the "Y" part.

5.7.2 Second Part of Flowchart

• Now we have the output of the "X" part from the previous stage, so we can proceed to obtain the "Y" signals.

• After getting the list of "X" signals, we categorize the PLLs into two types:

– PLL Type 1: These lead directly to "Y" without any additional tracing.

– PLL Type 2: These require further levels of tracing to reach the "Y" part.

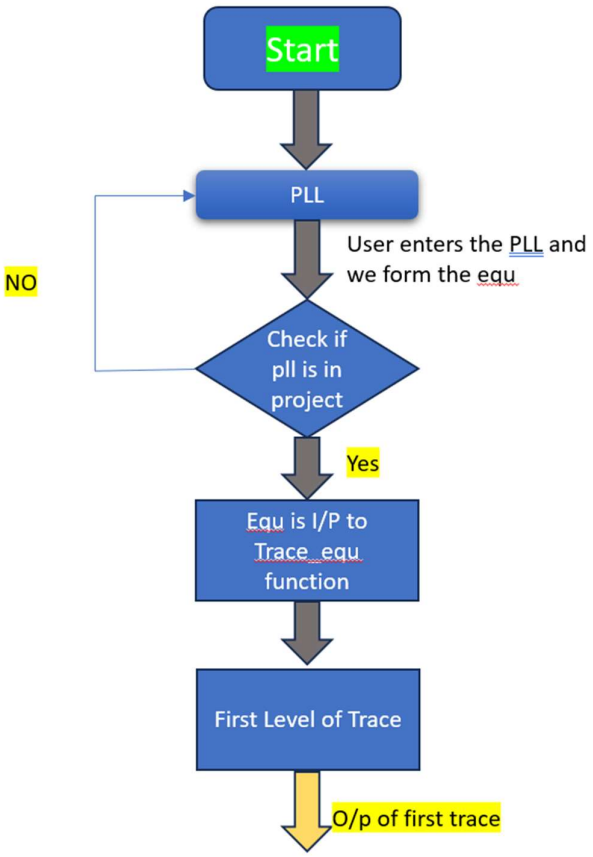

Figure 5.6 ; Flowchart start of Automation

• We check if the current list contains "X":

–If Yes: We obtain a list of the corresponding "Y" signals.

–If No: We return to the beginning of the second part for more tracing.

• Once completed, we move on to the "Z1" part.

5.7.3 Third Part of Flowchart

• The "Z1" part now takes input from the "Y" list.

• We check this list against requirement IDs to match specific conditions for automation.

• Any matches are pushed into a result table that serves as a base for the final output

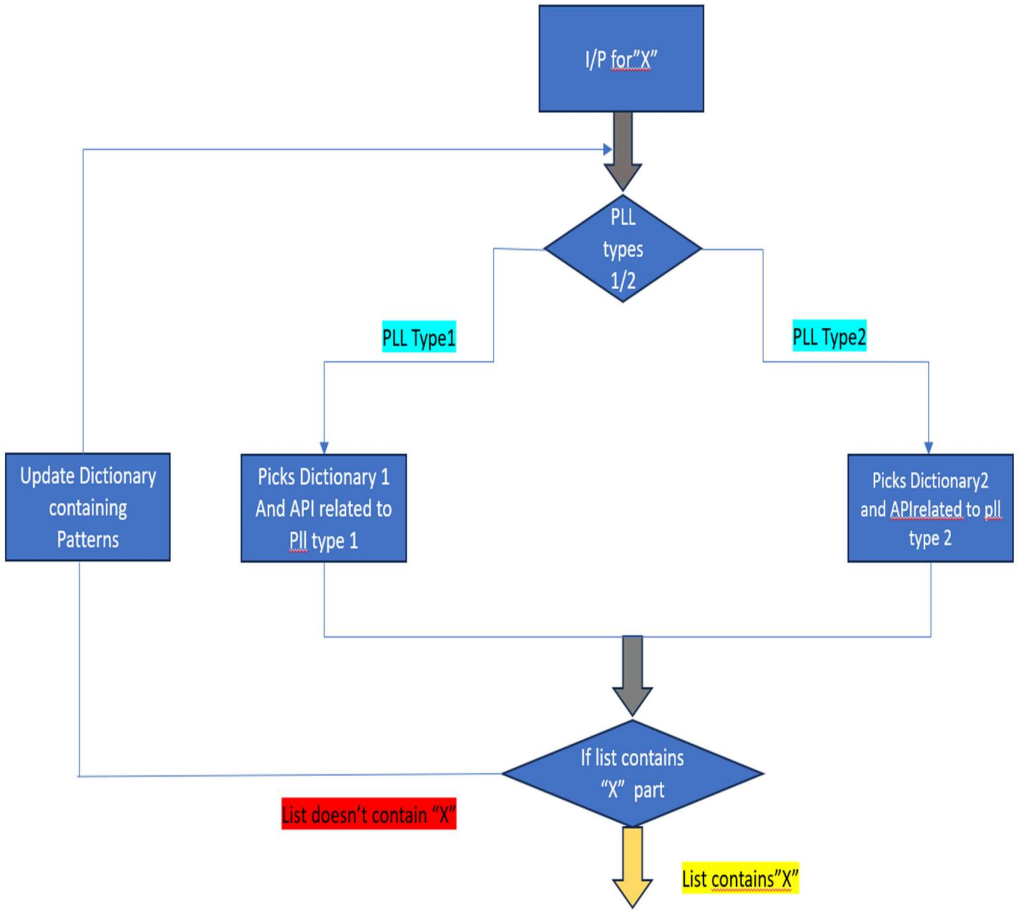

Figure 5.7: Flowchart Second Part

5.7.4 Fourth Part of Flowchart

• The result table from the previous stage is now processed to format it in a readable structure.

• We generate a CSV or report file that lists all the matched requirements and trace paths.

• This marks the end of the automation flow

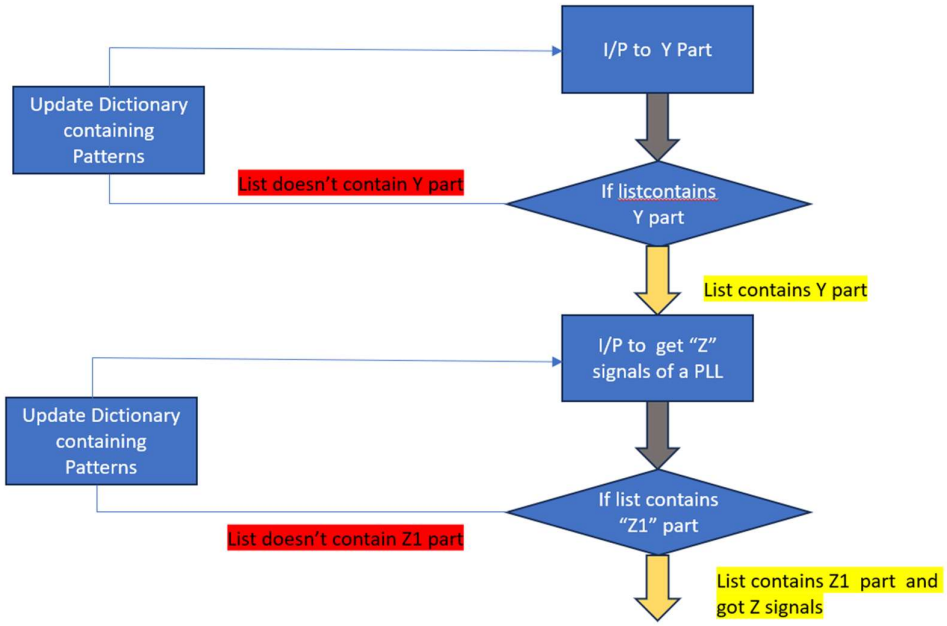

Figure 5.8: Flowchart Third Part

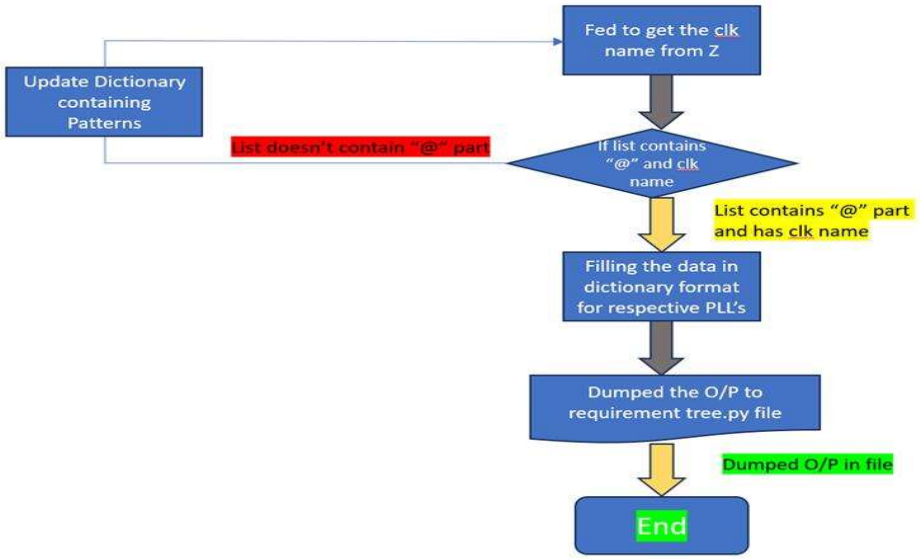

Figure 5.9: Flowchart Fourth Part

RESULTS

The requirement tree.py file generated after running the script for a particular project.

db_S0ix:{

db_pll1={

Parent.db_pll1={

'Data_Path': {

```
        "Register": "NULL",
        "Register_Bit": "NULL",
        "z1": "NULL",
        "Scan": "NULL",
        "Register_Required_Value": "NULL",
        "z1_Required_Value": "NULL",
        "Scan_Required_Value": "NULL",
        "Emulation_Value_c0": "NULL",
        },
    "Sub_Requirement":
        {
          "sub_req1"=["Mux_sel1",1,"NULL"],
          "sub_req2"=["Mux_sel0|mux_sel2",2,"NULL"],
          "sub_req3"=["NULL",0,"NULL"],
        }
},
"sub_req1"={
    'Data_Path': {
        "Register": "NULL",
        "Register_Bit": "NULL",
        "z1": "NULL",
        "Scan": "NULL",
        "Register_Required_Value": "NULL",
        "z1_Required_Value": "NULL",
        "Scan_Required_Value": "NULL",
        "Emulation_Value_c0": "NULL",
        },
    "Sub_Requirement":
        {
          "NULL":["Mux_sel0&mux_sel1",1,"NULL"]
        }
},
"sub_req2"={
    'Data_Path': {
        "Register": "NULL",
        "Register_Bit": "NULL",
        "z1": "NULL",
        "Scan": "NULL",
        "Register_Required_Value": "NULL",
```

```
        "z1_Required_Value": "NULL",
        "Scan_Required_Value": "NULL",
        "Emulation_Value_c0": "NULL",
        },
    "Sub_Requirement":
        {
          "NULL":["NULL",0,"NULL"]
        }
},
"sub_req3"={
    'Data_Path': {
        "Register": "NULL",
        "Register_Bit": "NULL",
        "z1": "NULL",
        "Scan": "NULL",
        "Register_Required_Value": "NULL",
        "z1_Required_Value": "NULL",
        "Scan_Required_Value": "NULL",
        "Emulation_Value_c0": "NULL",
        },
    "Sub_Requirement":
        {
          "NULL":["NULL",0,"NULL"]
        }
}
}
db_pll2={
Parent.db_pll2={
'Data_Path': {
        "Register": "NULL",
        "Register_Bit": "NULL",
        "z1": "NULL",
        "Scan": "NULL",
        "Register_Required_Value": "NULL",
        "z1_Required_Value": "NULL",
        "Scan_Required_Value": "NULL",
        "Emulation_Value_c0": "NULL",
        },
    "Sub_Requirement":
```

```
        {
          "sub_req1"=["NULL",0,"NULL"],
          "sub_req2"=["Mux_sel0&mux_sel1",1,"NULL"]
        }
    },
    "sub_req1"={
      'Data_Path': {
        "Register": "NULL",
        "Register_Bit": "NULL",
        "z1": "NULL",
        "Scan": "NULL",
        "Register_Required_Value": "NULL",
        "z1_Required_Value": "NULL",
        "Scan_Required_Value": "NULL",
        "Emulation_Value_c0": "NULL",
        },
      "Sub_Requirement":
        {
          "NULL":["NULL",0,"NULL"]
        }
    },
    "sub_req2"={
      'Data_Path': {
        "Register": "NULL",
        "Register_Bit": "NULL",
        "z1": "NULL",
        "Scan": "NULL",
        "Register_Required_Value": "NULL",
        "z1_Required_Value": "NULL",
        "Scan_Required_Value": "NULL",
        "Emulation_Value_c0": "NULL",
        },
      "Sub_Requirement":
        {
          "NULL":["NULL",0,"NULL"]
        }
    },
    }
```

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

6.1 Conclusion

We looked at different ways to manage power in this report, with a focus on Dynamic Voltage and Frequency Scaling (DVFS), Idle Power Management, and Active Power Management. The goal of this study was to find out how these methods can lower power use without hurting system performance. We found several important things by looking closely at each technique and how it is used in modern systems. We talked a lot about Dynamic Voltage and Frequency Scaling (DVFS) and how important it is for balancing power and performance. We looked closely at the ideas behind DVFS, especially how power, voltage, and frequency are related. We also looked at the many ways it can be used in different areas, like mobile devices, data centers, and embedded systems, to show how it can help save energy and make batteries last longer. In addition, the architecture needed to effectively manage voltage and frequency scaling was shown, which emphasized how important it is to have a strong power controller design. But, like any other technology, DVFS has its limits. Even though it saves power and extends battery life, switching between different frequency states can cause latency, which can slow down the system. Also, using DVFS in multi-core processors makes things even more complicated because it requires careful management of power distribution and synchronization. The report also talked about how important it is to manage idle and active power well. We talked about how different types of power management strategies can help lower power use when there is little or a lot of activity. To get the most out of modern devices' energy use, it's important to know how to manage and understand idle power.

i. **Power Management Cycles**: The specific cycles tested include:
   - **S4 (Hibernate) Cycle**: Ensures the system can save the RAM state to disk, power down, and restore the state upon waking.

ii. **Reset Cycles**: reset scenario is tested:
   - **Warm Reset**: A soft reboot that doesn't turn off the power completely.

iii. **Failure Detection and Classification**: After executing these cycles, it's crucial to verify their success. If any cycles fail, the logs must be examined to classify the failure into two types:
   - **Soft Hang**: The system is unresponsive but still powered on, often due to minor software or hardware issues.
   - **Hard Hang**: The system is completely unresponsive, requiring a power cycle to recover, indicating more severe issues.

iv. **Debugging Tools and Techniques**: To resolve failures, systematic debugging is necessary using:
   - **Logic Analyzer**: Helps trace the system's digital signals to understand failure points and events leading up to the failure.
   - **Oscilloscope**: Observes signal voltages to identify issues related to timing, voltage

levels, and signal integrity.

v. **Root Cause Analysis and Resolution**: Detailed analysis with these tools helps identify the root cause, whether due to firmware bugs, hardware issues, or system-level problems. Appropriate corrective actions are then taken to resolve the issues.

vi. **Ensuring Robustness**: This comprehensive process of testing, diagnosing, and debugging is crucial for maintaining system robustness. It ensures that the system performs reliably under various power states and reset conditions.

The report also showed how to automate the requirement tree (req tree) flowchart, which was meant to check the sub-requirements for different power states, especially G-states and how they relate to deeper low-power states like S0ix. The automated flow starts with specification files and then goes through signal equations and PLL categorizations to find the paths that lead to successful or failed low-power transitions. This automation makes sure that all requirements are met and points out any signals that are missing or wrong. In the end, this automation is very important when debugging. It gives you a powerful way to find out which signal or group of signals stops the system from going into deeper S0ix states. We can see exactly how signals depend on each other by bringing these sub-requirements to the surface through automation. This makes debugging faster and more focused. This speeds up development and makes the system more reliable by making sure that power state transitions happen correctly. In conclusion, power management techniques, especially DVFS, idle/active power strategies, and requirement tree automation, are not only important for designing modern systems, but they are also necessary for testing and debugging. Because of how they all affect energy efficiency, system performance, and the ability to debug, they are the building blocks of next-generation computing systems.

In conclusion, upgrading the system and running thorough power and reset cycle tests, followed by careful failure diagnosis and debugging with advanced tools, are important steps to make sure the system is stable and reliable.

6.2 Future Scope

There are several areas where we can explore further :

AI-based Power Management: We can use the concept of artificial intelligence and machine learning to predict workloads and dynamically adjust power settings.

Multi-Core Optimization: As the usuage of multi-core processor is increasing we can develop flow to manage power consumption in them..

Advanced DVFS Algorithms: There are some limitation with current DVFS algorithm, a more advanced DVFS algorithm is required that can manage wider range of performance and load.

Energy Harvesting and Optimization: Combining power management with energy harvesting techniques to further extend battery life and improve overall energy efficiency

Simulation and Modeling: A extensive use of simulation to consider the different scenarios of reset and power is needed before actual hardware testing, saving time and resources.

REFERENCES

[1] https://wiki.ith.intel.com/display/DDGLevelUpTraining/BIOS+and+OS

[2] A. S. Y. L. Lai, Dynamic Voltage Scaling for Power Management in Mobile Systems. IEEE Transactions on Computers, vol. 52, no. 6, pp. 779-788, June 2003. [Online]. Available: https://ieeexplore.ieee.org/document/1208832

[3] M. S. Y. L. Chen and H. C. Chang, A Survey of Power Management Strategies for DVFS and CPU Design. ACM Computing Surveys, vol. 47, no. 4, Article 69, 2015. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/2801062

[4] P. M. Menezes and V. C. V. S. Kumar, Dynamic Voltage Frequency Scaling (DVFS) Techniques for Mobile Devices: Energy Efficiency Challenges. Journal of Mobile Computing and Application, vol. 11, no. 3, pp. 224-235, March 2019. [Online]. Available: https://journals.sagepub.com/doi/abs/10.1177/2158244018825056

[5] S. G. H. Song, Optimizing Idle Power Consumption in Embedded Systems Using Dynamic Voltage and Frequency Scaling. IEEE Embedded Systems Letters, vol. 12, no. 2, pp. 40-43, April 2010. [Online]. Available: https://ieeexplore.ieee.org/document/5472862

[6] C. K. Lee, K. H. Choi, and J. H. Lee, P-State and C-State Management for Power Optimization in Multi-core Systems. ACM Transactions on Embedded Computing Systems, vol. 18, no. 1, pp. 1-20, 2019. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3300137

[7] History of Intel (Explore Intel's history) https://timeline.intel.com/

[8]https://uefi.org/htmlspecs/ACPI_Spec_6_4_html/08_Processor_Configuration_and_Control/processor-power-states.html

[9] A. Katyal and N. Bansal, "A Self-Biased Current Source Based Power-On Reset Circuit for On-Chip Applications," in VLSI Design, Automation and Test, 2006 International Symposium *on*, April 2006, pp. 1–4.

[10] Juan Carlos Saez and Manuel Prieto-Matias. "Evaluation of the Intel thread director technology on an Alder Lake processor". 13th ACM SIGOPS Asia-Pacific Workshop on Systems (APSys 2022). Association for Computing Machinery, New York, NY, USA, 61–67.https://doi.org/10.1145/3546591.3547532.

[11] R. K. Gupta, Challenges in Implementing Dynamic Voltage and Frequency Scaling for Power Efficiency. Proceedings of the IEEE International Conference on Power and Energy Systems, pp. 312-318, 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8776234

[12] J. M. A. Lu, R. R. C. Ahuja, Power Management Techniques for Data Centers: Leveraging DVFS for Efficient Resource Utilization. IEEE Transactions on Cloud Computing, vol. 5, no. 3, pp. 650-662, July-Sept. 2017. [Online]. Available: https://ieeexplore.ieee.org/document/7878483

[13] J. Liang, Y. Gan, Y. Li, M. Tan, and J. Wang, ''Thermal and electrochemical performance of a serially connected battery module using a heat pipe-based thermal management system under different coolant temperatures,'' Energy, vol. 189, Dec. 2019, Art. no. 116233, doi: 10.1016/j.energy.2019.116233.

[14] G. A. Kumar, "Secure, Scalable and Low-Power Junction Temperature Sensing for Multi-Processor Systems-on-Chip," 2020 21st International Symposium on Quality Electronic Design (ISQED), Santa Clara, CA, USA, 2020, pp. 179-182, doi: 10.1109/ISQED48828.2020.9136996.

[16] L. L. Wang, Energy-Efficient Systems and Dynamic Power Management in Modern Processors. Springer, 2019. [Online]. Available: https://www.springer.com/gp/book/9783030234245

[17] P. Pratik, V. Atmakuri, R. Kumar, A. Khanam, S. V. E and A. M, "Application of PID controller for thermal control in modern Intel® Core™ SoCs," 2022 IEEE 2nd Mysore Sub Section International Conference (MysuruCon), Mysuru, India, 2022, pp. 1-5, doi:10.1109/MysuruCon55714.2022.9972504.

[18]. Y. Zhao and L. J. Sun, Design of Power Controllers in DVFS Systems. Journal of Computer Architecture and Design, vol. 34, no. 2, pp. 115-130, March 2021. [Online]. Available: https://journals.elsevier.com/journal-of-computer-architectureand-design

[19] S. Chakraborty, V. Satnur, A. B. Lingambudi, A. Khandekar and S. H.Gunther, "Enhance Power and Temperature Control During Thermal Interference Events in Hybrid Core SoC Architecture," 2023 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), Bangalore, India, 2023, pp. 1-6, doi: 10.1109/CONECCT57959.2023.10234732.

[18]] G. Saharawat, P. Shukla, S. Jain and S. Nayak, "Emulation - Smart Way of Power Estimation and Power Aware Verfication," *2016 29th International Conference on VLSI Design and 2016 15th International Conference on Embedded Systems (VLSID)*, Kolkata, India, 2016, pp. 28-29, doi: 10.1109/VLSID.2016.142.

[20] K. Myung, S. Kim, H. Y. Yeom and J. Park, "Efficient and Scalable External Sort Framework for NVMe SSD," in *IEEE Transactions on Computers*, vol. 70, no. 12, pp. 2211-2217, 1 Dec. 2021, doi: 10.1109/TC.2020.3041220

[21] "Toward Bug-free Multicore SoC Architectures: System Validation with Transaction-Level Models," in *IEEE Design & Test of Computers*, vol. 28, no. 3, pp. 4-4, May-June 2011, doi: 10.1109/MDT.2011.68.

Hardeep Singh

Vibha_second

Paper

ece faculty

Thapar Institute Of Engineering and Technology, Patiala

Document Details

Submission ID
trn:oid:::1:3285373236

Submission Date
Jun 26, 2025, 7:05 PM GMT+5:30

Download Date
Jun 26, 2025, 7:10 PM GMT+5:30

File Name
AI_content_Report_vibha.pdf

File Size
2.8 MB

42 Pages

12,218 Words

66,482 Characters

Vibha_second

ORIGINALITY REPORT

5%	4%	2%	2%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.livewiredev.com Internet Source	<1%
2	Submitted to Kolej Universiti Poly-Tech MARA Student Paper	<1%
3	www.neuralconcept.com Internet Source	<1%
4	Georgios Kornaros. "Multi-Core Embedded Systems", CRC Press, 2019 Publication	<1%
5	classes.cs.uoregon.edu Internet Source	<1%
6	docplayer.net Internet Source	<1%

Hardeep Singh

Vibha_second

Paper

ece faculty

Thapar Institute Of Engineering and Technology, Patiala

Document Details

Submission ID

trn:oid:::1:3285373236

Submission Date

Jun 26, 2025, 7:05 PM GMT+5:30

Download Date

Jun 26, 2025, 7:10 PM GMT+5:30

File Name

AI_content_Report_vibha.pdf

File Size

2.8 MB

42 Pages

12,218 Words

66,482 Characters

*% detected as AI

AI detection includes the possibility of false positives. Although some text in this submission is likely AI generated, scores below the 20% threshold are not surfaced because they have a higher likelihood of false positives.

Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.