Application Note

CPU Cooling Using Dynamic Frequency Scaling In Linux

Keerthy J, Udit Kumar, Josiitaa RL

ABSTRACT

This application note describes how the Texas Instruments TDA4VL/J721S2 processor can leverage Linux based Dynamic Frequency Scaling (DFS) and the thermal framework to implement efficient CPU thermal management. This document addresses the current thermal sensor support available through the k3_j7xxx_bandgap driver and outlines steps to enable passive cooling using DFS. Targeted at Embedded Linux developers and system architects, this guide is intended for applications in Automotive and Industrial markets where power and thermal efficiency are critical. The information helps developers understand how to monitor thermal zones and implement scalable software controlled cooling strategies on TDA4VL/J721S2 based systems. This example demonstrates the implementation on TDA4VL and can be extended across other TDA4 and Sitara devices for thermal management on Linux.

Table of Contents

Trademarks

1 Introduction

1.1 Overview of TDA4VL SoC

The Texas Instruments TDA4VL SoC, part of the K3 family is designed for high performance automotive and industrial applications. Featuring dual-core Arm® Cortex® - A72 CPUs, a scalable memory architecture and integrated accelerators, the SoC provides both compute power and flexibility. Effective power and thermal management are critical in these applications, particularly when running high compute Linux based systems under variable workloads and environmental conditions.

1.2 Purpose of the Document

This application note explains how Linux based Dynamic Frequency Scaling (DFS) and the thermal management framework can be used on the TDA4VL device to manage CPU power and temperature effectively. The document highlights the current state of thermal sensor support, discusses DFS functionality, and provides guidance on integrating these features into embedded Linux systems.

The document also outlines an example implementation on the TDA4VL platform, showcasing how thermal CPU frequency scaling can be achieved using the Linux CPUFREQ and thermal frameworks.

1.3 Target Audience and Applications

This document is intended for:

- Embedded Linux developers
- System architects
- Application engineers
- Engineers working on TI's TDA4VL or other ARM based SoCs

This is relevant for use cases in:

- Automotive (ADAS, central compute ECUs)
- Industrial automation and robotics

1.4 Problem Statement

In thermally constrained environments, sustained high CPU performance can lead to overheating, system throttling, or shutdown. DFS, along with Linux thermal framework integration, provides a software-based design to dynamically control CPU frequency in response to system temperature, verifying operational reliability and efficiency without requiring extensive hardware cooling designs.

2 Dynamic Frequency Scaling (DFS) in Linux

2.1 What is DFS?

Dynamic Frequency Scaling (DFS) also known as CPU Frequency Scaling, is a power management technique that allows a processor to dynamically adjust the operating frequency based on performance demand or thermal conditions. Lowering the frequency reduces power consumption and heat generation, while improving performance.

On the Texas Instruments TDA4VL SoC, which features multiple Arm® Cortex® - A72 cores, DFS can be used to balance performance and power by scaling CPU frequencies at runtime. This is especially useful in embedded systems that operate under varying workload and thermal constraints.

2.2 Linux CPUFREQ Framework

The Linux kernel provides a standardized interface called the CPUFREQ framework to enable and control DFS. The framework allows users and system software to switch between different hardware-supported CPU frequencies during runtime.

Key components of the CPUFREQ framework include:

- CPU frequency drivers: These interface with hardware-specific mechanisms to control CPU clocks.
- Governors: Algorithms that decide when and how to change CPU frequency
- Sysfs Interface: Allows user-space interaction with the CPUFREQ subsystem (for example, /sys/devices/system/cpu/cpu*/cpufreq/)

2.3 Supported CPUFREQ Governors

The Linux kernel supports several governors for DFS, each designed for different use cases:

- Performance Governor:
  - Always sets the CPU to the highest supported frequency.
  - Best for applications where maximum performance is always required.
- Userspace Governor:
  - Allows user-space applications or scripts to set the CPU frequency manually.
- On Demand Governor:
  - Dynamically scales frequency based on current CPU load.
  - Commonly used in general-purpose systems.

Each governor can be selected and configured through the sysfs interface or automatically by system policy.

2.4 DFS Support Status on TI SoCs

- AM62 devices:
  - Full DFS support is available.
  - The CPUFREQ framework can dynamically adjust frequencies based on selected governor.
- TDA4 devices:
  - As of the current release (SDK 11.0), DFS support is not enabled.
  - The thermal monitoring infrastructure is in place via the k3_j7xxx_bandgap driver, but runtime frequency scaling is pending enablement.
  - Once supported, DFS allows TDA4 based systems to reduce power and temperature under thermal stress.

3 Linux Thermal Framework

The Linux thermal framework is a subsystem within the Linux kernel designed to monitor and manage the thermal state of a device. The framework provides a platform-agnostic infrastructure to define thermal zones, register temperature sensors, and apply thermal mitigation strategies such as dynamic frequency scaling (DFS), CPU hot plugging or fan control.

3.1 Thermal Zones and Trip Points

A thermal zone represents a temperature-monitored area in the system, such as a CPU cluster, GPU or PMIC. Each thermal zone is associated with:

- One or more temperature sensors
- A set of trip points, which are threshold temperatures
- Associated cooling devices triggered when trip points are reached

Trip points define the action thresholds (for example, passive or active cooling). Trip points can be:

- Passive: Trigger software based mitigation (for example, reduce CPU frequency)
- Active: Trigger hardware mechanisms (for example, start or increase fan speed)
- Critical: Used to shutdown the system to prevent hardware damage

These are typically defined in the device tree and handled through the thermal governor logic within the kernel.

3.2 Cooling Mechanisms: Passive vs Active

The thermal framework supports various cooling device types, categorized broadly into:

Passive Cooling:

- Reduces system temperature by lowering performance or power consumption
- Examples:
  - Lowering CPU frequency using DFS
  - Disabling or hot plugging CPU cores
- Passive cooling is preferred where silent or low power operation is required.

Active Cooling:

- Uses external mechanism to lower temperature, such as:
  - Turning on fans
  - Increasing fan speed through PWM control
- Requires external hardware and driver support (for example, a fan controller)

3.3 Role of DFS in Passive Cooling

On platforms such as the TI TDA4VL, DFS can serve as a passive cooling device under the Linux thermal framework. When a thermal zone crosses a defined trip point, the thermal framework can request a reduction in the CPU frequency to bring down the temperature.

This integration involves:

- Registering the CPU frequency driver as a cooling device
- Associating the CPU frequency with the relevant thermal zone
- Defining trip points that link to cooling states

Once DFS support is enabled on TDA4VL, a user can:

- Dynamically lower CPU frequency as temperature increases
- Maintain system stability and prevent overheating
- Operate in thermally constrained environments without requiring additional hardware cooling

4 Thermal Support on TDA4VL Devices

The Texas Instruments TDA4VL platform includes on-chip thermal sensing hardware and supports temperature monitoring through the Linux thermal framework. While full dynamic frequency scaling (DFS) support is not yet enabled for the A72 cores, foundational thermal monitoring capabilities are already available.

4.1 VTM and Bandgap Sensor Initialization

The Voltage and Temperature Monitor (VTM) subsystem in the TDA4VL SoC provides hardware support for thermal sensing. The subsystem includes:

- Multiple on-die temperature sensors (bandgap-based)
- Infrastructure for monitoring thermal zones

In Linux, VTM hardware is initialized via the k3_j7xxx_bandgap driver. This driver enables:

- Reading temperature values from multiple thermal sensors on the SoC
- Exposing the thermal zones to the Linux thermal framework

Once initialized, temperature data becomes accessible through the sysfs interface at: /sys/class/thermal/thermal_zone*/temp

4.2 Temperature Monitoring via k3_j7xxx_bandgap Driver

The k3_j7xxx_bandgap driver performs low level initialization of thermal sensors and provides software hooks to register thermal zones. Features include:

- Reading real-time temperatures from the A72 cores, GPU, and other key regions of the SoC.
- Mapping each physical sensor to a logical thermal zone.
- Integration with Linux thermal core to trigger trip points

This functionality enables developers to observe thermal behavior and plan cooling strategies, even before active/passive cooling mechanisms are implemented.

5 Enabling CPU Cooling on TDA4VL

This section provides the necessary patch to enable passive CPU cooling on the TDA4VL device using Dynamic Frequency Scaling (DFS) and outlines steps to verify the functionality.

5.1 Patch to Enable CPU Cooling

```
From e9ffcedb3b567110fda18f4a49fb8e66672910e4 Mon Sep 17 00:00:00 2001
From: Keerthy <j-keerthy@ti.com>
Date: Fri, 20 Jun 2025 14:52:39 +0530
Subject: [PATCH] arch/arm64/boot/dts/ti/k3-j721s2-thermal.dtsi: Add CPUFREQ
 based cooling
Add CPUFREQ based cooling
Signed-off-by: Keerthy <j-keerthy@ti.com>
---
 arch/arm64/boot/dts/ti/k3-j721s2-thermal.dtsi | 14 ++++++++
 arch/arm64/boot/dts/ti/k3-j721s2.dtsi | 33 +++++++++++++++++++
 2 files changed, 47 insertions(+)
diff --git a/arch/arm64/boot/dts/ti/k3-j721s2-thermal.dtsi b/arch/arm64/boot/dts/ti/k3-j721s2-
thermal.dtsi
index e3ef61c16..69727a5a5 100644
--- a/arch/arm64/boot/dts/ti/k3-j721s2-thermal.dtsi
+++ b/arch/arm64/boot/dts/ti/k3-j721s2-thermal.dtsi
@@ -39,12 +39,26 @@ main0_thermal: main0-thermal {
 thermal-sensors = <&wkup_vtm0 2>;
 trips {
+ main0_alert: main0-alert {
+ temperature = <45000>; /* millicelsius */
+ hysteresis = <2000>; /* millicelsius */
+ type = "passive";
+ };
+
 main0_crit: main0-crit {
 temperature = <125000>; /* milliCelsius */
 hysteresis = <2000>; /* milliCelsius */
 type = "critical";
 };
 };
+
+ cpu_cooling_maps: cooling-maps {
+ map0 {
+ trip = <&main0_alert>;
+ cooling-device =
+ <&cpu0 THERMAL_NO_LIMIT THERMAL_NO_LIMIT>;
+ };
+ };
 };
 main1_thermal: main1-thermal {
diff --git a/arch/arm64/boot/dts/ti/k3-j721s2.dtsi b/arch/arm64/boot/dts/ti/k3-j721s2.dtsi
index 4f416bb18..71dcb9e6f 100644
--- a/arch/arm64/boot/dts/ti/k3-j721s2.dtsi
+++ b/arch/arm64/boot/dts/ti/k3-j721s2.dtsi
@@ -51,6 +51,10 @@
 d-cache-line-size = <64>;
 d-cache-sets = <256>;
 next-level-cache = <&L2_0>;
+ clocks = <&k3_clks 202 0>;
+ clock-names = "cpu";
+ operating-points-v2 = <&cpu0_opp_table>;
+ #cooling-cells = <2>; /* min followed by max */
 };
 cpu1: cpu@1 {
@@ -65,6 +69,35 @@
 d-cache-line-size = <64>;
 d-cache-sets = <256>;
 next-level-cache = <&L2_0>;
+ clocks = <&k3_clks 203 0>;
+ clock-names = "cpu";
+ operating-points-v2 = <&cpu0_opp_table>;
+ #cooling-cells = <2>; /* min followed by max */
+ };
+ };
+
+ cpu0_opp_table: opp-table {
+ compatible = "operating-points-v2";
+ opp-shared;
+
+ opp6-2000000000 {
+ opp-hz = /bits/ 64 <2000000000>;
+ clock-latency-ns = <300000>;
+ };
+
+ opp4-1000000000 {
+ opp-hz = /bits/ 64 <1000000000>;
+ clock-latency-ns = <300000>;
+ };
+
+ opp2-500000000 {
+ opp-hz = /bits/ 64 <500000000>;
+ clock-latency-ns = <300000>;
+ };
+
+ opp1-250000000 {
+ opp-hz = /bits/ 64 <250000000>;
+ clock-latency-ns = <300000>;
 };
 };
--
2.17.1
```

5.2 Testing the Cooling Functionality on TDA4VL

Once the patch is applied and the kernel is rebuilt and deployed, follow these steps to validate CPU cooling operation:

1. Verify cooling device registration.

   ```
   cat /sys/class/thermal/cooling device*/
   ```

   Look for entries such as cpu-freq indicating that the CPU frequency driver is registered as a cooling device.
2. Check thermal zones trip point.

   ```
   cat /sys/class/thermal/thermal_zone1/trip_point_0_temp
   45000
   ```

   In this example, the trip point temperature has been set to 45°C
3. Check the current and maximum state.

   ```
   cat /sys/class/thermal/thermal_zone1/cooling_device0/cur_state
   0
   cat /sys/class/thermal/thermal_zone1/cooling_device0/max_state
   3
   ```

4. Check the current and available scaling frequencies.

   ```
   cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_cur_freq
    2000000
   cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_available_frequencies
    250000 500000 1000000 2000000
   ```

   The current CPU frequency is 2GHz, with the available frequencies ranging from 250MHz to 2GHz.
5. Check the current temperature.

   ```
   cat /sys/class/thermal/thermal_zone2/temp
   44753
   ```

   The current temperature is around 44.753ºC
6. Trigger CPU Load.

   ```
   cpuloadgen 100 100 &
   [1] 1246
   ```

   Using the cpuloadgen command, the CPU core is pegged at 100% for 100 seconds to increase the CPU load.

7. Check the current temperature.

```
cat /sys/class/thermal/thermal_zone2/temp
45209
```

   As the CPU load increases, the SoC temperature increases to 45.209ºC, triggering the dynamic CPU cooling using frequency scaling. This then brings down the temperature to 44.981ºC.

```
cat /sys/class/thermal/thermal_zone2/temp
44981
```

8. Check the current scaling frequency and cooling state.

```
cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_cur_freq
250000
cat /sys/class/thermal/thermal_zone1/cooling_device0/cur_state
3
```

The current frequency has been brought down to 250MHz and the device has reached the maximum cooling state.

Note

The trip point is deliberately programmed as 45°C just to showcase the full functionality at room temperature. The user must calibrate the trip temperature based on the system.

6 Scalability Across TDA4 and Sitara Devices

The methodology described in this application note for enabling DFS-based passive CPU cooling on the TDA4VL platform can be adapted to other Texas Instruments TDA4 and Sitara family SoCs with similar CPUFREQ and thermal framework support.

6.1 Adapting the Implementation

When porting the design to other devices in the TDA4 or Sitara families:

1. Update Device Tree Nodes
   - Adjust CPU OPP tables according to the supported frequencies of the target SoC.
   - Configure thermal zones based on available test conditions
2. Verify DFS Support
   - Make sure that the CPUFREQ driver of the target SoC supports runtime frequency changes.
   - Enable the correct governors in the kernel configuration.
3. Thermal Zone Mapping
   - Associate the CPU cooling device with the correct thermal zones for the new SoC.
4. Test Under Load
   - Repeat the functional testing procedure described in Testing the Cooling Functionality on TDA4VL to confirm correct cooling behavior.

7 Summary

This application note demonstrated the implementation of DFS-based passive CPU cooling on the TDA4VL platform using the Linux thermal framework. By integrating CPUFREQ cooling devices with on-die thermal sensors through device tree configuration, the system can dynamically adjust CPU frequency to maintain safe operating temperatures without requiring active cooling components.

The provided patches and test procedures offer a practical reference for developers to replicate the design on similar TDA4 and Sitara devices. With minor modifications to CPU OPP tables, thermal zone configurations, and CPUFREQ governors, the same approach can be adapted to a wide range of TI SoCs.

Adopting this method not only improves system reliability and longevity but also enables silent, power-efficient operation in thermally constrained environments.

8 References

1. Samsung Electronics Co,*CPU cooling APIs How To*
2. Intel Corporation, *Generic Thermal Sysf driver How To*
3. Texas Instruments, *TDA4VE TDA4AL TDA4VL Jacinto™ Processors, Silicon Revision 1.0*, data sheet.