SOC Fault Indication Strategy to the MCU for the Adaptive Partitions during Bootup Stage

Vinoot Handiganoor and Basavana Gowda B[2]

[1]Department of SWX SPDO, Stellantis India.

[2]Department of SWX SPDO, Stellantis India.

ABSTRACT

The abstract introduces the concept of adaptive partitioning in System on Chip (SoC) technology. It means dividing the chip's resources into different sections to meet the specific needs of different software tasks. The main purpose is to improve performance and flexibility in scenarios with multiple applications or tasks running simultaneously. Adaptive partitioning also has advantages like efficient resource use, scalability, security, and system design flexibility. This paper emphasizes the importance of a fault indication mechanism between the SoC and the MCU (Microcontroller Unit) to handle bootloader issues during startup, especially with efficient Over-The-Air updates. The paper's focus is on how the SoC and MCU communicate and address problems when bootloaders fail within partitions. It stresses the need for a recovery mechanism to handle potential issues with corrupted updates, ensuring a smooth SoC bootup process.

KEYWORDS

Network System on Chip, Adaptive Partition, Fault Indication Mechanism, Microcontroller Unit, Bootloader, Adaptive AUTOSAR

1. INTRODUCTION

This study focuses on the communication between the MCU (TC37XX series) and the SOC (6155 QC series) during the bootup process. Partition A and Partition B are the two partitions that reside in the eMMC region. Communication between the MCU and the SOC can be accomplished via GPIO or SPI. When the 6155 Chipset encounters challenges during the boot up process, this study focuses on GPIO signalling. The overall block diagram is shown below, and the subsequent paragraphs describe what adaptive partition is and why it is required.

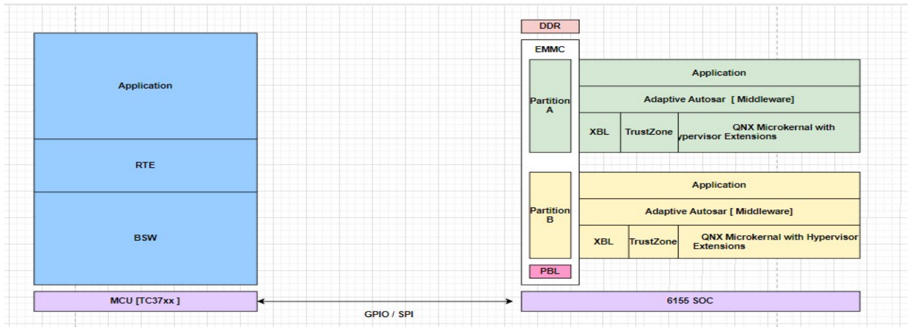
Application
RTE
BSW
MCU [TC37xx]
GPIO / SPI
DDR
EMMC
Partition A
Application
Adaptive Autosar [Middleware]
XBL
TrustZone
QNX Microkernal with Hypervisor Extensions
Partition B
Application
Adaptive Autosar [Middleware]
XBL
TrustZone
QNX Microkernal with Hypervisor Extensions
PBL
6155 SOC

Figure 1: Overall block diagram and communication between MCU and SOC

As Over the Air update capabilities advance, the complexity rises and better bootup and communication between MCU and SOC are required. It is described in the following subsections why adaptive partitioning is necessary. The study goes on to discuss how communication will be handled if the booting process halts on the SOC side, as well as how partition management and job distribution need to be handled while handling adaptive partitions.

1.1 What is Adaptive Partition?

An adaptive partition is a concept used in embedded systems to dynamically allocate system resources, such as processing power, memory, and peripherals, among different partitions or domains based on the changing workload and requirements of the system. Each partition is a logical unit that is allocated a portion of the system resources and runs specific tasks or applications. The need for adaptive partitions arises due to several reasons:
Adaptive partitioning provides a number of benefits to the design, development, running, and debugging of your system.

- Engineering product performance: Adaptive partitioning allows us to optimize the performance of the system.
- Dealing with design complexity: Designing large-scale distributed systems is inherently complex. Typical systems have a large number of subsystems, processes, and threads developed in isolation from each other. The design is divided among groups with differing system performance goals, different schemes for determining priorities, and different approaches to runtime optimization.
- Providing security: Many systems are vulnerable to Denial of Service (DOS) attacks. For example, a malicious user could bombard a system with requests that need to be processed by one process. When under attack, this process would overload the CPU and effectively starve the rest of the system.
- Debugging: Adaptive partitioning can even make debugging an embedded system easier—during development or deployment—by providing an “emergency door” into the system.

1.2 Why Adaptive partition needed?

The following are the reasons why we need adaptive partitions.

Efficient Resource Utilization: Adaptive partitioning allows for the optimal utilization of system resources. By dynamically allocating resources based on the specific requirements of different tasks or applications, it ensures that each partition receives the necessary resources to perform its designated functions. This prevents resource over-provisioning and under utilization, leading to improved system efficiency.

Isolation and Security: Adaptive partitioning provides strong isolation between different tasks or applications running on the system. Each partition operates independently, with its own allocated resources and memory space. This isolation prevents interference and unauthorized access between partitions, enhancing system security and protecting sensitive data.

Real-Time Performance: In real-time systems, where tasks must meet strict timing requirements, adaptive partitioning is essential. By dedicating resources specifically to real-time tasks, it ensures that critical tasks are executed within their specified deadlines. The partitioning scheme enables the system to prioritize and allocate resources accordingly, guaranteeing real-time performance.

Flexibility and Scalability: Adaptive partitioning offers flexibility and scalability in system design. As the system requirements evolve or change over time, new partitions can be added or existing partitions can be resized without affecting the overall system functionality. This adaptability enables the system to accommodate varying workloads and adjust resource allocation accordingly.
Fault Isolation and Reliability: By isolating tasks or applications within separate partitions, adaptive partitioning enhances fault isolation and system reliability. If a failure or error occurs in one partition, it remains contained within that partition and does not impact the operation of other partitions. This isolation improves overall system reliability and prevents system-wide failures.
Mixed-Criticality Systems: Adaptive partitioning is particularly beneficial in systems with mixed-criticality requirements. Different tasks or applications can be assigned to separate partitions based on their criticality levels. Critical tasks can be allocated dedicated resources and strict scheduling policies, ensuring their reliable execution without being affected by less critical tasks.
Improved Development and Maintenance: Adaptive partitioning simplifies the development and maintenance of embedded systems. Each partition can be developed, tested, and maintained independently, allowing for modular development and easy updates. Changes or updates in one partition do not affect the operation of other partitions, reducing the impact on the overall system.

2. How Adaptive Partition Works with respect to MCU and SOC communication?

Coordination and data interchange between the Microcontroller Unit (MCU) and the various adaptive partitions in the System on Chip (SoC) are made possible by the communication between the two. Here is an overview of this communication with respect to Partition Configuration, Partition Management and Task Distribution.

2.1. Partition Configuration:

To set up each adaptive partition's unique parameters this enables distributing resources to each partition in accordance with their needs, such as computing cores, memory, and peripherals.
It is vital to remember that the specific techniques and tools for partition setting may change based on the target hardware platform and the QNX OS version being utilized. Partition configuration general overview is described below in the figure 2.

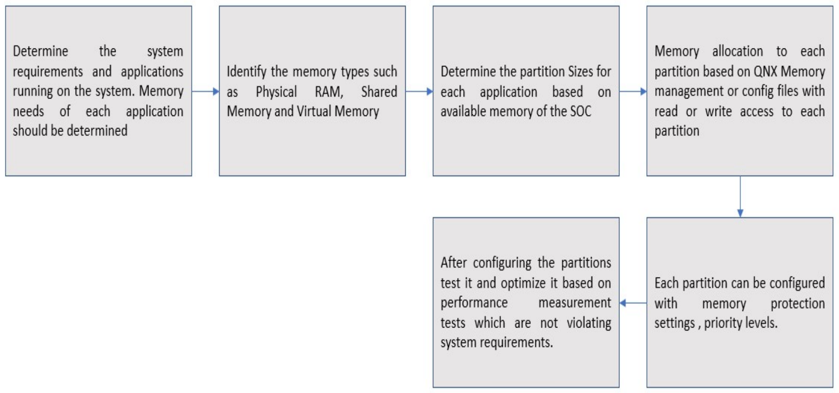
Determine the system requirements and applications running on the system. Memory needs of each application should be determined
Identify the memory types such as Physical RAM, Shared Memory and Virtual Memory
Determine the partition Sizes for each application based on available memory of the SOC
Memory allocation to each partition based on QNX Memory management or config files with read or write access to each partition
Each partition can be configured with memory protection settings , priority levels.
After configuring the partitions test it and optimize it based on performance measurement tests which are not violating system requirements.

Figure 2: Partition Configuration General Overview

2.2. Partition Management

The title is Using adaptive partitioning, the SoC resources are divided into various partitions, each of which is assigned to a certain job or set of apps. The MCU assists in controlling these partitions by orchestrating their actions and facilitating Inter-Partition Communication.

Inter-Process Communication can be achieved using message passing, shared memory, signals and events mechanisms.

- Message Passing: It permits the exchange of messages between the processes using primitives for sending and receiving messages.
- Shared Memory: To alert another partition of a certain condition or request, a partition can send a signal or raise an event.
- Signals and Events: Shared memory areas can be explicitly constructed and mapped to different partitions, allowing them to gain access and share data in that area.

I/O Operations: using standard I/O operations such as read, write, or ioctl we can achieve the communication between MCU and SOC. Its another way to achieve communication.

2.3. Task Distribution

The MCU is in charge of allocating tasks or applications to the proper partitions. It chooses which partition to carry out a specific task depending on scheduling policies, resource availability, and priority.

Fault Monitoring and Recovery: The MCU watches for faults or errors in the adaptive partitions. It can receive partition status updates, detect anomalies, and take recovery operations as needed. Fault recovery may entail reallocating resources, restarting a partition, or performing other corrective actions to restore system stability.

The communication between the MCU and adaptive partitions in a SoC allows for effective system management, coordination, and data exchange. The end-user needs are increasing, and software updates will be critical, with a focus on fault monitoring and recovery between MCU and SOC as a solution to address increased deployment use cases. In some circumstances, attackers may update faulty software or bootloaders; if partitions are not available, there is a possibility that the SOC will not act as intended, and the bootloader will not be able to execute if we do not have recovery from the MCU side.

Below figure depict the handling of communication between MCU and MPU.

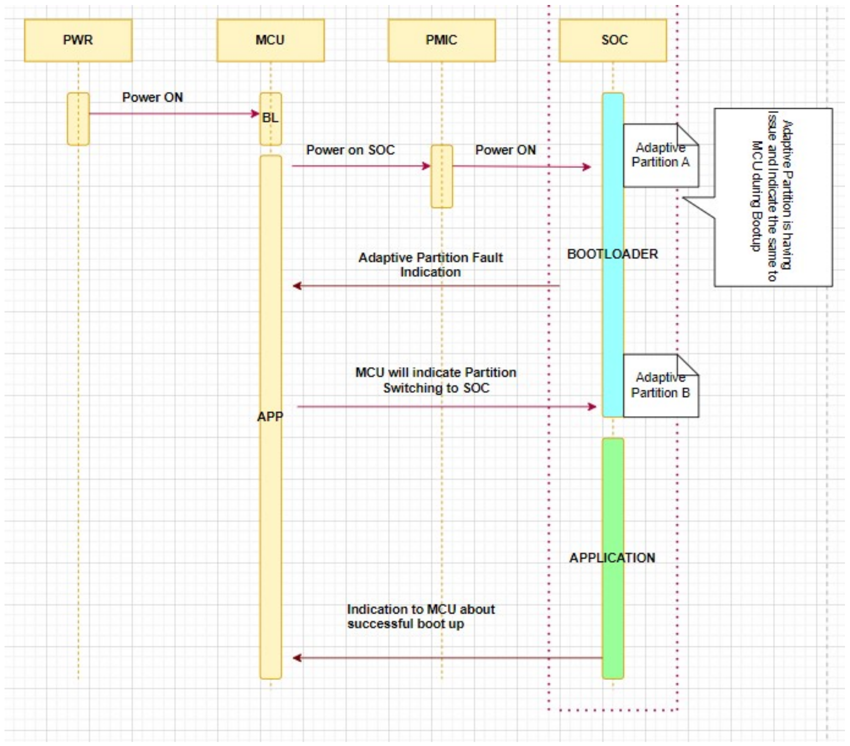
PWR
MCU
PMIC
SOC
Power ON
BL
Power on SOC
Power ON
Adaptive Partition A
Adaptive Partition is having issue and Indicate the same to MCU during Bootup
BOOTLOADER
Adaptive Partition Fault Indication
MCU will indicate Partition Switching to SOC
Adaptive Partition B
APP
APPLICATION
Indication to MCU about successful boot up

Figure 3: Communication between MCU and SOC

Partition A in Adaptive Partition contains the functioning binary or default image, and Partition B is utilized for over-the-air upgrades. There should be a mechanism to alert the MCU during the boot-up phase if the update is corrupted, as there are several ways to detect it. Otherwise, the SOC won't exit the boot-up phase. It is always preferable to create a system that can handle a variety of scenarios robustly. Here, we will be exhibiting two such techniques shown in Figure 4 and Figure 5. Below Figure 4 handle the communication between MCU and SOC adaptive partitions.

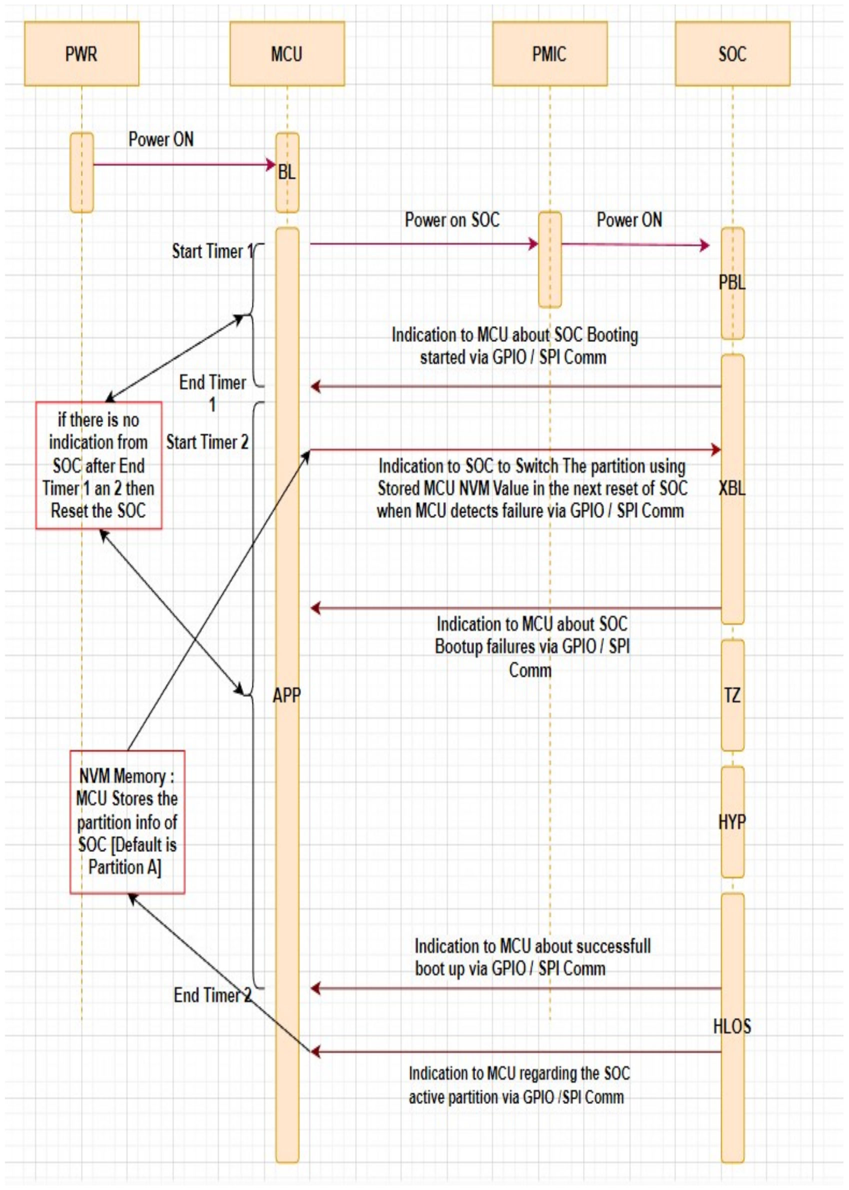
PWR
MCU
PMIC
SOC
Power ON
BL
Power on SOC
Power ON
Start Timer 1
PBL
Indication to MCU about SOC Booting started via GPIO / SPI Comm
End Timer 1
if there is no indication from SOC after End Timer 1 an 2 then Reset the SOC
Start Timer 2
Indication to SOC to Switch The partition using Stored MCU NVM Value in the next reset of SOC when MCU detects failure via GPIO / SPI Comm
XBL
Indication to MCU about SOC Bootup failures via GPIO / SPI Comm
APP
TZ
NVM Memory : MCU Stores the partition info of SOC [Default is Partition A]
HYP
Indication to MCU about successfull boot up via GPIO / SPI Comm
End Timer 2
HLOS
Indication to MCU regarding the SOC active partition via GPIO /SPI Comm

Figure 4: Start-up sequence and error indication between MCU and SOC bootloader via GPIO's

In the image above, after the PMIC power on, the SOC primary bootloader will inform the MCU about soc booting through GPIO pins. There will be timers running on the MCU side; if no answer is received from the SOC indicating successful boot-up, it will store the failure information in non-volatile memory and switch to a different partition during the next boot-up phase. This way, we can ensure that a boot failure is detected and create a partition switch to exit the failure scenario.

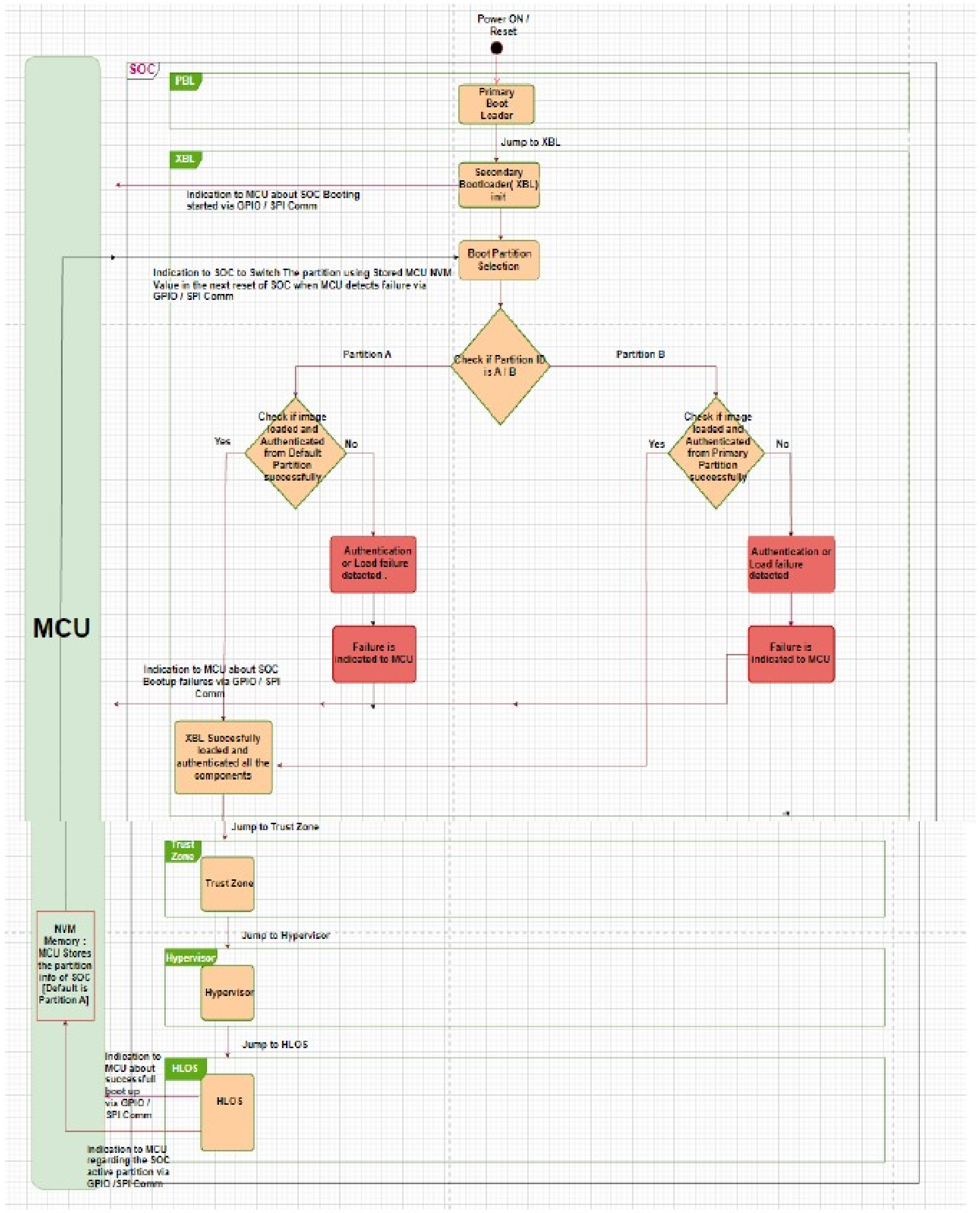
Power ON / Reset
SOC
PBL
Primary Boot Loader
Jump to XBL
XBL
Secondary Bootloader(XBL) init
Indication to MCU about SOC Booting started via GPIO / SPI Comm
Boot Partition Selection
Indication to SOC to Switch The partition using Stored MCU NVM Value in the next reset of SOC when MCU detects failure via GPIO / SPI Comm
Check if Partition ID is A / B
Partition A
Partition B
Check if image loaded and Authenticated from Default Partition successfully
Check if image loaded and Authenticated from Primary Partition successfully
Yes
No
Yes
No
Authentication or Load failure detected .
Authentication or Load failure detected
Failure is indicated to MCU
Failure is indicated to MCU
MCU
Indication to MCU about SOC Bootup failures via GPIO / SPI Comm
XBL Succesfully loaded and authenticated all the components
Jump to Trust Zone
Trust Zone
Trust Zone
NVM Memory : MCU Stores the partition info of SOC [Default is Partition A]
Jump to Hypervisor
Hypervisor
Hypervisor
Jump to HLOS
HLOS
HLOS
Indication to MCU about successfull boot up via GPIO / SPI Comm
Indication to MCU regarding the SOC active partition via GPIO /SPI Comm

Figure 5: Adaptive Partitions Handling

The above flow diagram depicts the internal failure check and partition switching mechanism.In 2030, 99 OTA upgrades are anticipated, and this approach will aid in determining whether the software is authentic or functional. If a software update is not working well, the MCU may elect to instruct the SOC to switch to an adaptive partition that has the default software, which is working properly or operating without any issues.

In conclusion, a trusted bootloader file starts the device startup procedure, and every phase is only executed after the one before it has been correctly completed.

3. Adaptive Partition Challenges during start-up sequences

During an embedded system's startup process, adaptive partitioning might pose a number of difficulties. The following are some typical issues that could occur:

Boot Order and Dependencies: Dependencies between various partitions must be taken into account throughout the startup sequence of adaptive partitions. Some partitions might depend on particular assets or services offered by other partitions. When there are intricate interdependencies between partitions, managing the boot order and making sure all dependencies are appropriately addressed can be difficult.

Resource Contentions: During start-up, multiple partitions may compete for shared resources, such as memory or peripherals. Resource contentions can occur when partitions simultaneously request resources, leading to potential delays or conflicts. Efficient resource allocation and scheduling mechanisms are required to handle these contentions and ensure fair access to shared resources.

Initialization Sequence: The initialization states of certain partitions may be dependent on those of others. The right starting sequence must be chosen in order to satisfy these dependencies. Incorrect behavior or system problems may occur if the initialization order is not appropriately controlled.

Start-up sequences are prone to faults and failures. Maintaining system integrity requires finding and dealing with faults like resource allocation issues or partition initialization failures. It is a considerable problem to implement efficient error detection algorithms and recovery techniques to manage such circumstances. partitions. Determining the correct order of initialization to satisfy these dependencies is crucial. If the initialization order is not properly managed, it may result in incorrect behavior or system failures.

Error Handling and Recovery: Detecting and handling errors, such as partition initialization failures or resource allocation errors, are essential for maintaining system integrity. Implementing effective error detection mechanisms and recovery strategies to handle such situations is a significant challenge.

System Complexity and Design Verification: The processes involved in system design and design verification are getting more difficult in adaptive partitioning. To guarantee the system works as intended, it is necessary to thoroughly verify the behavior and performance of many interacting partitions, especially during the startup stage.

It takes careful system design, resource management methods, synchronization mechanisms, error handling techniques, and extensive testing to overcome these obstacles. In order to guarantee a successful and dependable start-up sequence in an embedded system with adaptive partitioning, these issues must be properly taken into account and mitigated.

4. Best Practices to address above challenges

There are few best practices that can be used to create the start-up sequence for adaptive partitions in an embedded system that will assist guarantee a quick and effective initialization process. Here are some crucial actions to think about:

a) Clear Initialization Order Definition:

Based on the dependencies and requirements of the adaptive partitions, create a clear initialization order for them. Based on resource dependencies and inter-partition communications, determine which partitions must be initialized first and which can be initialized later. This order's documentation and enforcement assure correct startup and prevent dependencies-related problems.

b) Use Initialization Flags or Events:

Use initialization flags or events to show when each partition has finished initializing. When a partition has finished its startup procedures and is ready to move on, these flags or events can be used to indicate that fact. Then, before starting their own initialization, other partitions can verify these flags or events to make sure that dependencies have been met.

c) Early Allocation of Critical Resources in the startup process:

Identify and assign the crucial resources needed by various partitions. By doing this, it is made sure that crucial resources, such as shared memory or communication interfaces, are accessible when they are required. Early resource allocation lowers the likelihood of resource conflict and potential delays during partition initialization.

d) Implement Resource Management techniques:

To deal with resource contention during startup, implement resource management techniques. Implement algorithms or scheduling rules that effectively and equitably manage shared resources, avoiding conflicts and providing equitable resource access among partitions.

e) Test and Validate Startup Sequences:

Validate the adaptive partition startup sequences in great detail. To ensure that all partitions initialize correctly, make sure all dependencies are met, and the system functions as intended during startup, conduct thorough testing. Verify the dependability and robustness of the startup process by doing boundary case testing, stress testing, and simulating various failure scenarios.

f) Implement reliable error detection systems:

Add error recovery techniques to your system to handle errors and restore it to a known and stable state. Restarting failed partitions, reallocating resources, or commencing the proper error handling procedures may all be necessary in this situation.

g) Implement logging and monitoring systems:

To record startup events and keep track of the start-up sequence's development implementing the logging and monitoring systems is crucial. This enables thorough debugging and troubleshooting in the event of any problems or startup failures.

h) Documentation and maintenance:

To give clear instructions for future maintenance and troubleshooting, document the start-up sequence, including the order of initialization and any specific considerations. As the system develops or undergoes modifications, keep the documentation current.

You can improve an embedded system's adaptive partition start-up sequence's dependability, effectiveness, and maintainability by adhering to these best practices. It aids in ensuring proper system initialization, proper handling of dependencies, and system readiness for effective operation.

5. Acronyms

Table 1. Abbreviations.

Sl No	Acronym	Abbreviation
1	SoC	System on Chip
2	MCU	Micro Controller Unit
3	MPU	Micro Processor Unit
4	TZ	Trust Zone
5	HLOS	High Level Operating System
6	XBL	Secondary Boot Loader
7	HYP	Hypervisor
8	GPIO	General Purpose Input Output
9	PBL	Primary Bootloader
10	PMIC	Power Management Integrated Chip
11	NVM	Non-Volatile Memory

6. Conclusions

If the communication between MCUs and SOC is handled effectively with an error handling mechanism, adaptive partitions play a significant role in the over-the-air update capabilities. It is possible to do what is outlined in the best practices with the appropriate tools and implementations. The importance of adaptive partitions and communication between the MCU and SOC must be considered early in the design process. Stellantis is already active and providing solutions to allow secure communication between adaptive partitions.

Acknowledgements

The authors would like to thank everyone from the SPDO dept team. Special thanks to Joseph Antony, Sivapalan Balanayagam and Tara Vatcher for their continuous supports.

REFERENCES

[1] Christian Menard; Andrés Goens; Marten Lohstroh; Jeronimo Castrillon, (2020) "*Achieving Determinism in Adaptive AUTOSAR*", Automation & Test in Europe Conference & Exhibition (DATE)

[2] Anand Bhat & Soheil Samii, (2020), "*Fault-Tolerance Support for Adaptive AUTOSAR Platforms using SOME/IP*", IEEE 26th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)

[3] Jelena Jovicic , Mila Kotur & Milan Z. Bjelica, (2018), *"Visualizing Functional Verification in Adaptive AUTOSAR",* IEEE 8th International Conference on Consumer Electronics

[4] Anna Arestova, Maximilian Martin, Kai-Steffen Jens Hielscher, & Reinhard German1,(2021), "*A Service-Oriented Real-Time Communication Scheme for AUTOSAR Adaptive Using OPC UA and Time-Sensitive Networking*" Sensors (Basel). 2021 Apr; 21(7): 2337.

[5] QNX website from the blackberry for the technicalities

[6] Adaptive Autosar Consortium for the Adaptive Partition concepts.

Authors

Vinoot V Handiganoor

Software Development Manager – Stellantis India

Basavana Gowda B

Technical Specialist - Stellantis India