**Meteorlake Memory Subsystem Performance and Power Analysis Learnings**

Memory subsystem is one of the complex systems to validate and tune for performance and power. On MTL, we faced multiple challenges due to the various architectural changes – disaggregated architecture, hetero cores, introduction of the NetSpeed Fabric, large PL1 ranges, AI workloads in the mix – to name a few.

This paper focuses on two of these challenges and learnings from these that are being considered for future architecture.

1. **Memory Geyserville (MemGV) Tuning:** MemGV is a concept to modulate the operating frequency of domains during run time in order to extract the best performance in a given power budget. The concept for MemGV has been around for a while; and a single set of weights are applied across all the SKUs on a client product across varying PL1 limits.

Historically, this was fine since the range of supported power envelopes was tighter together. On a product like Meteorlake (MTL), where the supported envelopes range from 9W all the way to 45W, we have seen to be leaving about 5% performance on the table if we characterize for the bookend envelopes. Another complexity in the architecture is introduced due to the disaggregated architecture, which brings in two clock domains that need to go through frequency transitions at the same time.

This happens since the best operating points for the same workload for a 9W SKU is different from the operating points for a 45W SKU. For a 9W SKU – we have seen that the best operation for a workload like Spec is at a lower memory operating point while for a 45W SKU we need to operate at the best bandwidth point.

With the same set of weights – it is difficult for the algorithm to navigate to the best operating point unless we have a power aware algorithm that will put a constraint on the memory operating points based on the power budgets available.

This can be done using various methods – two of these ideas could be:

* Power aware memory Geyserville algorithm
* SoC budget should include memory subsystem along with core and graphics while allocating power

1. **Sensitivity to core stall metrics:** MemGV algorithm uses memory bandwidth (MemBW) requirement and latency introduced as a result of compute die stalls (Core Stall) to make decisions to move between the frequency points. Our observation on MTL architecture indicated a very high sensitivity to the Core Stall metrics. This drives memory to use the latency point more often – on MTL this was the Gear2 point that had the highest memory and fabric clock frequencies – driving the voltage requirement also higher and thus increasing the power on the memory rail.

On MTL, this translated to a customer issue where we saw a small percentage of the latency point while running a battery life workload – in this case it was Windows Idle. This had a power impact of ~350mW in the scenario.

For MTL and other products in flight – we had to architect a solution that was specific to battery life (BL) workloads – this was implemented in the firmware that made MemGV decisions for BL workloads based only on the MemBW and not Core Stalls.

For future architectures – we are looking at below to make sure what we use in decision making is robust:

* Definition of the Core Stalls to check if the right metrics for hetero architecture is being used.
* Interaction of static algorithms like prefetcher tuning and dynamic algorithms like MemGV
* Latency requirements from IPs other than the core – namely Graphics and NPU