Gen-Z envisages next-generation of memory management

Parmeshwr Prasad
DellEMC
An influx of data is driving the change
Gen-Z solving memory related issues

- System memory is getting flat
- Real-time analysis is growing
- Needed for open architecture
- Eliminates single point of failure
- Reduce data movement, improve latency, envisages new solutions
Gen-Z connector

- Processor to component
- Gen-Z bus
- Req./Resp. protocol Engine
- Link initiator
- Physical
- Fabric
Gen-Z Memory initialization

1. Power-on / Reset (SEC)
2. CPU/Chipset Init (PEI)
3. Enumerate and configure Gen-Z Control Structures. Configure bridge CID / OpCode Set for In-band Configure PMCID = bridge CID
4. Detect Gen-Z Components via I2C
5. PCIe PHY?
   - yes
     - Init/Train PCIe PHY
   - no
     - Init/Train 802.3 PHY
6. Traverse Switches
7. For Each Memory Component, Update Available Memory Structures to Program Requester
8. Configure CPU Addressable Memory incl. Gen-Z
9. DXE
10. BDS
11. OS Hand-off
12. End
Gen-Z data transfer

- All the operation in packet format
- Data originator-Responder
- Data consumer-Requester
- Data management unit (ZMMU)
Requestor ZMMU

- Maps responder address space
- Enables application transparent access
- ZMMU applicable only for explicit OpClass Operations
Requestor ZMMU Cont…
Responder ZMMU

- Translate packet’s address to media address
- Enforce access permission
- P2P-Core, P2P-Coherent
Responder ZMMU Cont...
## P2P Core OpClass

- Acknowledgment, Unreliable Write, Persistent Flush Read/Write,
- Read (16, 32, 64, 128 and 256)
- Read Offset (32, 64, 128 and 256)

### Table: P2P Core OpClass Structure

<table>
<thead>
<tr>
<th>Bit Position</th>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-3</td>
<td>OpCode</td>
<td>4:3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1:0</td>
</tr>
<tr>
<td>1</td>
<td>Tag [10:8]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LEN [6:3]</td>
<td></td>
</tr>
<tr>
<td>7-5</td>
<td>Tag [7:4]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LEN [2:0]</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>L</td>
<td>2:0</td>
</tr>
<tr>
<td>13-11</td>
<td>Tag [3:0]</td>
<td></td>
</tr>
<tr>
<td>63-32</td>
<td>Address</td>
<td>63:32</td>
</tr>
<tr>
<td>31-0</td>
<td>Address</td>
<td>31:0</td>
</tr>
<tr>
<td>54</td>
<td>ECRC</td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>R0</td>
<td></td>
</tr>
</tbody>
</table>

### Diagram: P2P Core OpClass Structure

![Diagram of P2P Core OpClass](image-url)
Coherent Read/Write

- Should support all Coherent OpClass
- Shell use cache line size address.
- Coherent should support Link-Level-Reliability (LLR)
- Has standard acknowledgment package
Address translation

Invalidate Package

Translation Package
Address translation Cont...

- Translation service
- TA invalidates the translated address
- Translation request
- Translation response
Page service

- On-demand page residency service
- PRI
- PRG
Thank you!