OS-aware IP Development Methodology

Hyunjae Woo, Woojoo Kim, Youngsik Kim, Seonil Brian Choi
Samsung Electronics Co., LTD.
Index

• Related Research
  – Hybrid SoC Emulation Methodology
  – IP development perspective

• Proposed Methodology
  – Basic concept
  – Transactor
  – Latency Issues and Memory I/F
  – Validation Platform

• Application Result

• Conclusion
Hybrid SoC Emulation Methodology(1)

• Pure SoC Emulation.
  – It is 1000 times faster than simulation
  – But it is not fast enough to develop SW
    (It takes around ten hours to boot Linux and Android)

• Hybrid SoC Emulation Methodology.
  – CPU and Memory are virtualized to accelerate performance.
  – It is 15 times faster than Pure SoC emulation.
    (It takes 49 min to boot Linux and Android)
Hybrid SoC Emulation Methodology (2)

• SW development perspective.
  – It is the same environment as that of real silicon mobile AP.
  – It can be supports Camera module and mobile storages such as eMMC, UFS, USB by additional transactors.

![Diagram of Hybrid SoC Emulation Methodology]
IP development perspective

• IP developers both SW/HW want to focus on their own IP.
  – Some other IPs occur side effects such as Kennel panic, lock up and target IP mal-function (over-write wrong address).
• Not enough time to develop and verify
Proposed Methodology
Basic Concept

• Focusing on Target IP.
  – Verified QEMU based on Common OS system.
  – Only put Target IP and related memory sub-system at emulator.
    • Saving Time to debug other IPs.
    • Saving emulator capacity
    • Saving compilation time
Proposed Methodology
Transactor

• Transactor is to communicate Virtual and Emulator.
  – It make possible to communicate tranaction level not cycle level.
  – What if we use cycle level communication ~????
• There are CPU, GIC and memory transactor.
Proposed Methodology

Latency Issues

– The Bandwidth is not a bottleneck, PCIe is up to 32 GB/s as current emulator.
– when communication is occurred, emulator’s clocks are held until it finished.
– It is impossible to reduce Latency, but we can avoid it by reducing counts.
Proposed Methodology
Memory I/F

• Memory should be allocated both Virtual and Emulator.
  – As the latency issues, the counts of transaction should be minimized.
  – The important things to synchronize by events.
    • Sync up When CPU Xtor access IPs.
    • Sync up When Timer and CPU’s instruction reach the setting values

• Memory which is at emulator side can be implementation cache or full memory with score boards.
Proposed Methodology
Validation Platform(1)

- GPU IP Development.
  - The Target IP has own firmware to control Job Managements, KMD (Kernel Mode Driver) and UMD (User Mode Driver).
  - All IPs except GPU related are on QEMU.
  - Running real BM and Apps
Proposed Methodology Validation Platform (2)

- HW & SW co-development, co-validation and co-verification, just put IP HW (RTL + FW) and Driver at the Platform.
### Application results(1)

<table>
<thead>
<tr>
<th></th>
<th>Simulation</th>
<th>Pure Emulator</th>
<th>Hybrid SoC Emulation Methodology</th>
<th>Proposed Methodology</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Kernel Boot-up (Prompt)</strong></td>
<td>125,867 min*</td>
<td>96 min</td>
<td>2 min</td>
<td>1 min</td>
</tr>
<tr>
<td></td>
<td>(87.4 days)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Android Platform (Home Screen)</strong></td>
<td>741,517 min*</td>
<td>661 min</td>
<td>47 min</td>
<td>9 min</td>
</tr>
<tr>
<td></td>
<td>(514.9 days)</td>
<td>(11.0 hours)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Total Consumed Time</strong></td>
<td>867,384 min*</td>
<td>757 min</td>
<td>49 min</td>
<td>10 min</td>
</tr>
<tr>
<td></td>
<td>(602.4 days)</td>
<td>(12.7 hours)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Estimated Value
Application results(2)
Conclusion.

- Proposed Methodology.
  - 5 times faster than Hybrid SoC emulation methodology.
  - On time delivery by doing early SW/HW co-verification.
  - Find out various bugs which are FirmWare<-> KMD, KMD<-> RTL and KMD<->UMD.

- Future Work.
  - Power Profile flow over android.
  - Peak and Average power analysis and optimize.