Monitoring Specification and Software for ASICs & Chiplets

OCP ODSA Project Workshop

Ashwin Poojary, Facebook Inc.
Dharmesh Jani, Facebook Inc.

12.09.2019

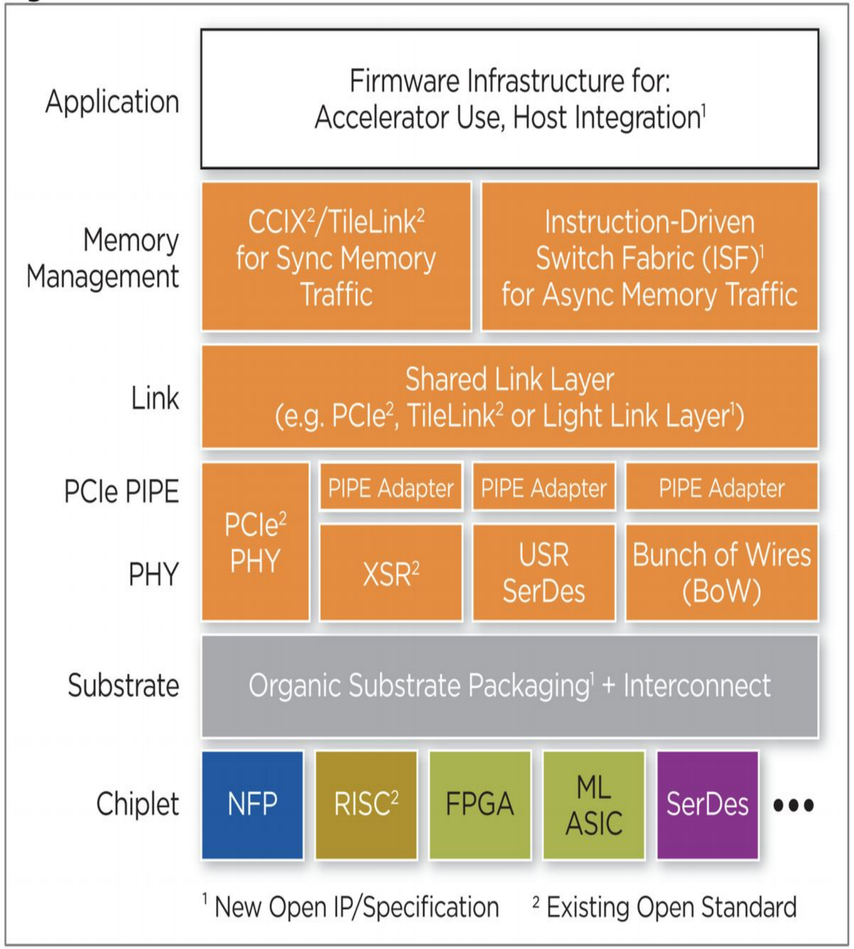
Agenda

ODSA Stack

Integrate into best-of-breed components - processors, memory, hardware engines, I/O peripherals, etc.

Effective DSAs require the integrated development of silicon, systems, firmware and **software**.

They also need good **tooling** & **monitoring** infrastructure


Application
Firmware Infrastructure for: Accelerator Use, Host Integration[1]
Memory Management
CCIX[2]/TileLink[2] for Sync Memory Traffic
Instruction-Driven Switch Fabric (ISF)[1] for Async Memory Traffic
Link
Shared Link Layer (e.g. PCIe[2], TileLink[2] or Light Link Layer[1])
PCIe PIPE
PHY
PCIe[2] PHY
PIPE Adapter
PIPE Adapter
PIPE Adapter
XSR[2]
USR SerDes
Bunch of Wires (BoW)
Substrate
Organic Substrate Packaging[1] + Interconnect
Chiplet
NFP
RISC[2]
FPGA
ML ASIC
SerDes
•••
[1] New Open IP/Specification
[2] Existing Open Standard
Source: ODSA

Spectrum

- Hyper-scale companies
- Cloud & Network Providers
- Small scale companies
- ASIC Vendors
- ODMs & OEMs
- Integrators

Telemetry and Monitoring

It is widely agreed upon that we need reliable health monitoring for ASICs to ensure highest levels of availability and reliability. Every consumer of ASIC/chiplets implements telemetry and monitoring which may include checking certain hardware properties and application level performance metrics. Whether Domain specific → Google’s TPU for AI and ML or Netronomes’ flow processor for network flow and security, the things we monitor for health would be something like Temperature, Power etc. We focus here on the hardware specific monitoring since hardware properties are common across accelerator types and applications may vary vastly.

Current methods

- **sysfs**
    - **sysfs** is a pseudo file system that exports information
- **ioctl**
    - a system call for device-specific input/output operations which cannot be expressed by regular system calls, accessing device drivers
- **smbios**
    - System Management BIOS (SMBIOS) is a standard developed by DMTF to allow the operating system to retrieve information about the PC.
- **vendor libraries & tools**
    - vendor specific software
- **open tools** like dmidecode, lspci or similar in-house tools
- **linux utils, openbmc utils,** etc.
- **log files.**
    - /var/log/...

Heterogeneity at Scale causes Issues

- No consistency
- Interoperability
- Lack of Flexibility to assemble systems in a variety of technologies
- Single vendor lock-in
- Complex integration issues
- Cost of integration
- Small companies cannot spend big dollars on creating a software stack
- Lack of Openness
- Slow Feature Velocity

- Most organizations using OCP hardware today, still work on their own proprietary software stack. Given accelerators are relatively new there aren't any standard interfaces to communicate with them. Consumers push each vendor to have a minimum set of capabilities. Each vendor has vendor specific details. The consumer of the ASIC comes up with their own solution to define and consume requirements for each vendor.

- The intent should be to remove the repetitive and error prone work of porting and re-defining interfaces, checks and tests for each ASIC or chiplet variant or new vendor deployed.

- As more chiplets become available with compatible interfaces, companies will be able to create solutions more quickly and cost effectively.

Goals

- Enable a complete ecosystem of silicon, systems, firmware and software
- Enable building tools/telemetry to have a common specification for all ASICs/chiplets.
- Ability to monitor ASICs from different vendors and generations
    - by providing a standardized view of metrics irrespective of the custom device interfaces offered by each device type.
- Ensure correct aggregation, data synchronization, fault-tolerance for failed devices.
    - (A server will have multiple ASIC modules and chiplets.)
- Able to iterate quickly on source of truth specification changes.
- Provide an IP Agnostic transaction layer

Path to the Goal

- Create an Open Specification for Hardware Monitoring and Telemetry
- Decide on the format(s) for expressing the Open Specification
- Design Scalable Software and Interfaces
- Create Ecosystem for monitoring, testing, benchmarking.
  - An open development environment where chiplets from different companies have common interfaces and support a common protocol stack
- Foster Interoperability and Agnosticity among vendors

Guiding Principles

- Abstraction
- Inclusiveness
- Genericity
- Programmability
- Extensibility and Backward Compatibility
- Efficiency
- Designed for Scale
- A daemon, a Library with APIs and a CLI

What to expose?

- System configuration and state
- Performance counters - module level, device level, domain instance level, (process level?)
- Hardware characteristics
- Error events - DIMM, PCIe, device failures etc.
- Performance-generated event streams (sampling profiles ?)

Classes of information

- Vendor details
- Chiplet capabilities
- Software versioning
- Firmware specifications
- Thermal characteristics
    - current temperature
    - max temperature seen
    - max temperature supported
    - throttle temperature(s)
    - TDP (Thermal Design Power)

Classes of information (...cont'd)

- Frequency and Power
  - current frequency
  - maximum frequency
  - current power
  - maximum power

- Error/Debugging information
  - e.g. correctable DIMM errors
  - e.g. uncorrected DIMM errors
  - e.g. PCIe errors etc.

Classes of information (...cont'd)

- Performance and Efficiency Statistics
  - utilization percentage for different metrics
- Generalized System Throughput
  - operations per second
- Memory Performance Metrics
  - memory bandwidth utilization percentage
  - memory capacity utilization percentage
- Historical value reporting (?)

We could extend the specification to details like type of the metric, number of digits of precision, the units used, etc.

Proposed Formats

Protocol buffers, XML, YANG, SOAP, REST, etc.

Thrift

- Is an interface definition language and binary communication protocol used for defining and creating services in numerous languages
- RPC for cross language scalable service development
- Developed by Facebook, adopted worldwide and maintained by Apache Software Foundation

JSON (JavaScript Object Notation)

- Open standard using human readable text to transmit serializable data objects
- Very common data transmission and machine readable parsing technique
- Language independent

Supported by languages like Python, C, C++, Java, Go, Perl, Ruby, Rust, Haskell, etc.

Advantages

- Faster integration of new vendor into the tooling ecosystem #movefast
- Better Planning for generic software and checks on live hardware.
- Adoption of common standard
- Free common software in the ecosystem
- Combine forces to solve issues faster
- More reliable and robust system
- An open chiplet ecosystem will enable wider use of chiplets and accelerated SoC development and larger user base
- Cost effectiveness

CREDITS

Aaron Miller, Ashwin Narasimha, Ashwin Poojary, Ben Wei, Brian Coutinho, Chris Peterson, Craig Ross, Cynthia Liu, Dan Forest, David Xiao, Dharmesh Jani, Gene Oden, Hao Shen, Jeremy Yang, Jerry Liu, Jonathan Killinger, Makan Diarra, Manali Kesarkar, Olof Johansson, Ray Park, Sam Naghshineh, Shishir Juluri, Soumya Padmanabha, Thiara Ortiz, William Arnold and several others.

Thank you!