oneAPI HW SIG (June 22nd 2023)

Explicit Device Model

Jaime Arteaga
Michal Mrozek
Paul Petersen
Will Damon

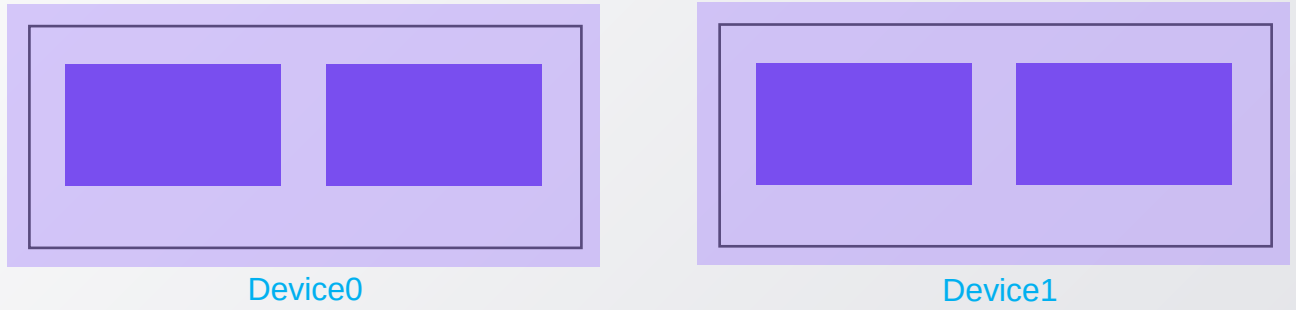
Problem Statement

- Due to low-bandwidth in cross-tile access on PVC, implicit scaling might not always provide improved performance when compared with one tile.
- Because of this, there is a proposal to change from **card-as-devices** model to a **tiles-as-devices** model, where tiles are exposed as primary devices.
- Objective: By having tiles-as-devices as default model, users would have to explicitly opt-in to go to implicit model, where lower performance is seen.
- [Expose sub-device exposed by ZE_AFFINITY_MASK as devices · Issue #1 · oneapi-src/level-zero-spec · GitHub]

Current Model: Implicit Model

A dual-PVC system is exposed with 2 devices (`zeDeviceGet`), each representing a card and with implicit scaling, where L0/OCL automatically distributed memory and kernels among tiles

Applications can access tiles by querying for sub-devices (`zeDeviceGetSubDevices`)



New Model: Explicit Model

A dual-PVC system is exposed with 4 devices (`zeDeviceGet`), each representing a tile, without implicit scaling.

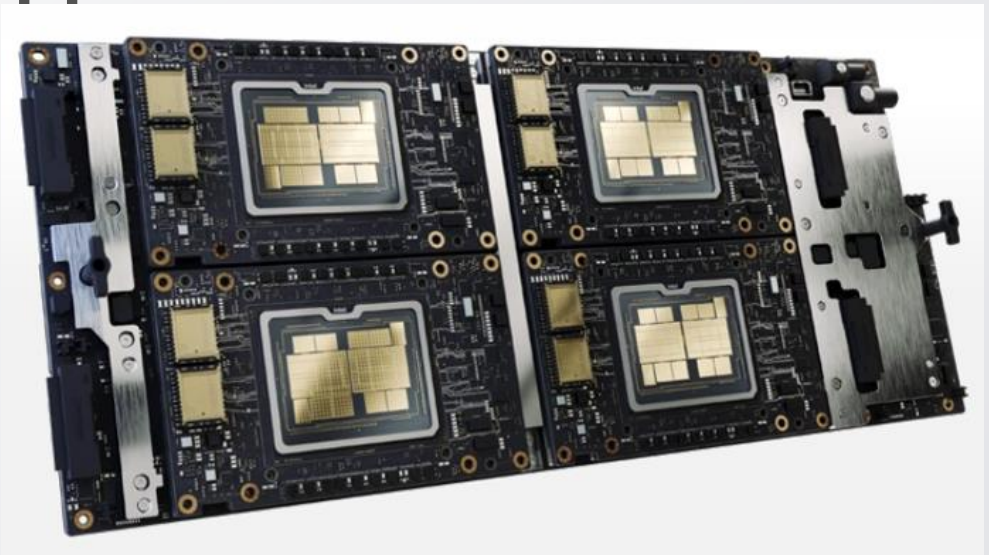
Interaction between model, affinity mask, and multi-CCS mode

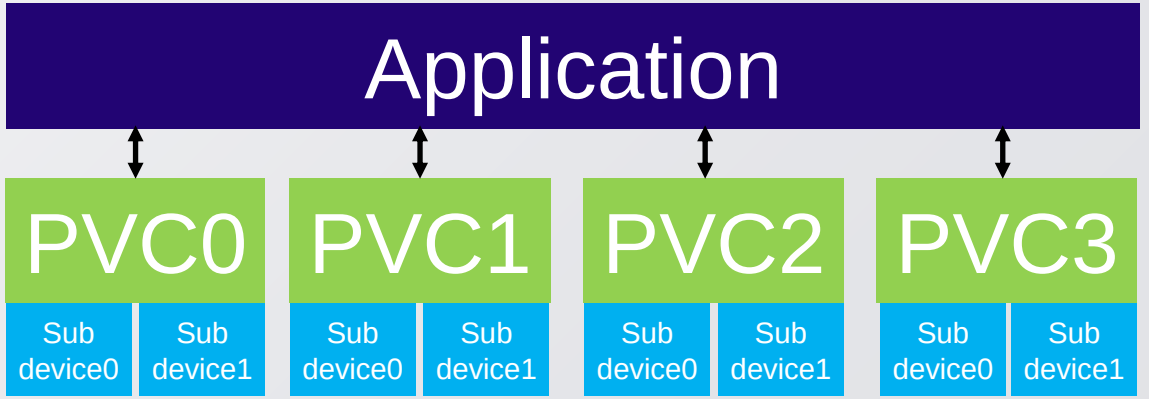
- Affinity mask associated masks devices at the default level: For instance, in a 6-PVC system:
  - In implicit model, `ZE_AFFINITY_MASK` accepts values from 0 to 6, and X.Y values are used to mask tiles.
  - In explicit model, `ZE_AFFINITY_MASK` accepts values from 0 to 11, and X.Y values are ignored, as tiles have not sub-devices.
- Multi-CCS mode is set for the whole PVC (not possible to apply individually to each tile):
  - `ZEX_NUMBER_OF_CCS` accepts values for <card>:<number of CCSs>
  - `ZEX_NUMBER_OF_CCS=0:4`
    - Explicit Model: give me 4 CCSs on all tiles of PVC 0: sub-device 0 has 4 CCSs, sub-device 1 has 4 CCSs, root device has 1 CCS (implicit scaling)
    - Implicit Model: give me 4 CCSs on all tiles of PVC 0: device 0 has 4 CCSs, device 1 has 4 CCSs

ZE_FLAT_DEVICE_HIERARCHY

- Mode 0

- Cards-as-devices model is used
- Cards contains sub devices
- Resources allocated on parent device are available on sub devices even if they are not specified in the context
- Implicit scaling possible
- Current default

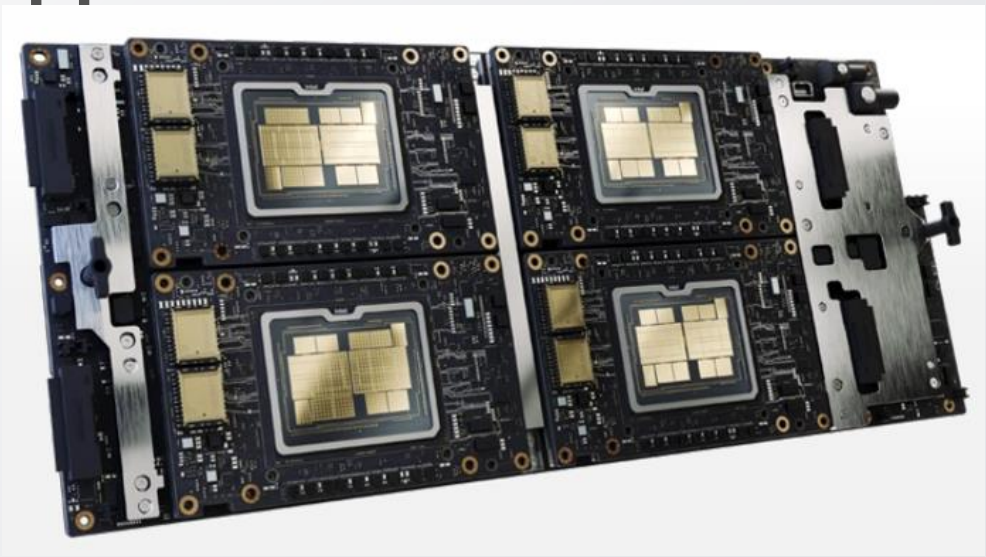


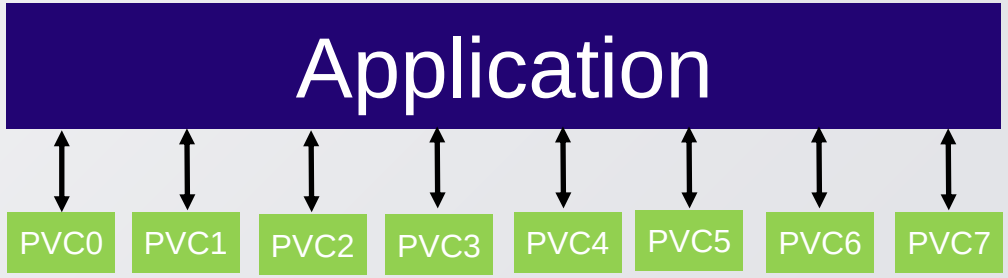


ZE_FLAT_DEVICE_HIERARCHY

- Mode 1

- Tiles-As-Devices model is used
- No Sub Devices, flat model
- zeDeviceGetRootDevice returns nullptr
- Implicit scaling not possible
- Most efficient mode of operation
- Proposed new default

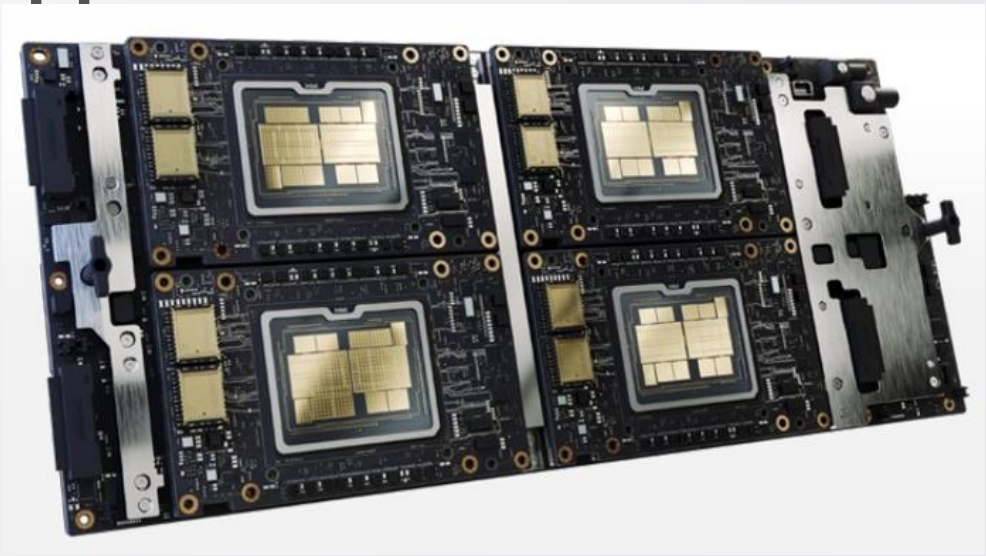


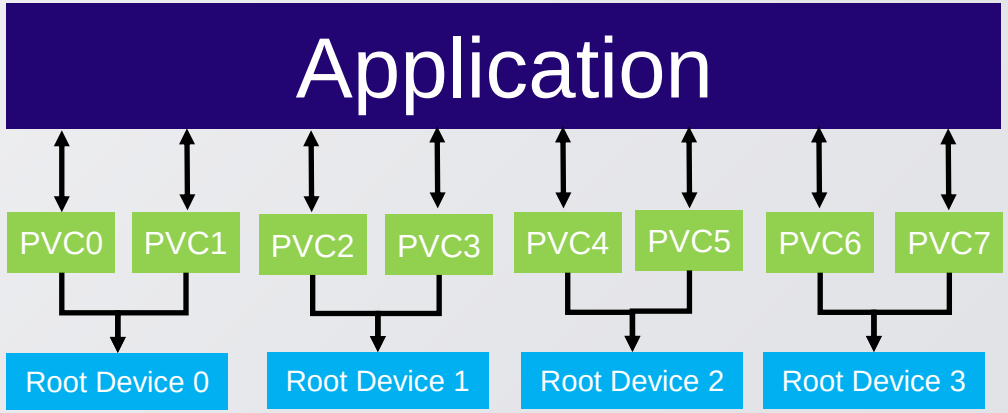


ZE_FLAT_DEVICE_HIERARCHY

- Mode 2

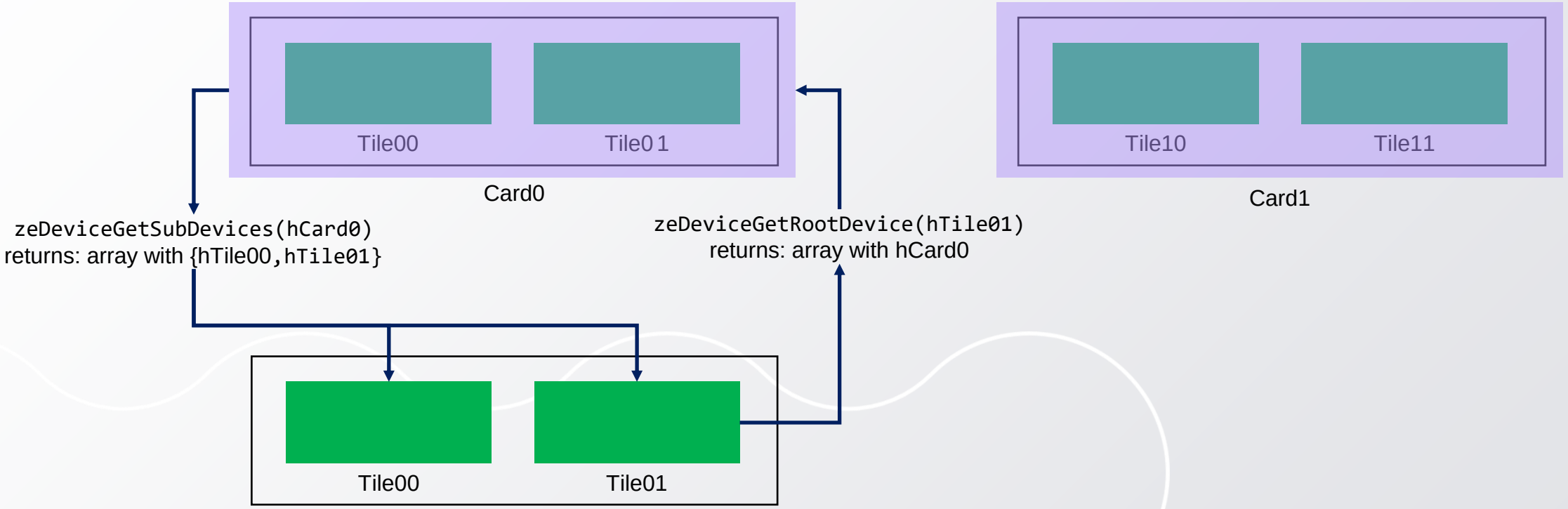
- Tiles-As-Devices model is used
- No Sub Devices, flat model
- zeDeviceGetRootDevice returns root device
- Resources allocated on tile-devices are NOT automatically available on root devices
- In order to ensure resource availability on both tile-devices, context must contain root device
- Implicit scaling available on Root Devices
- Proposed opt in mode





Enumerating GPU topology Mode 0: Cards-as-devices

`zeDeviceGet(hDriver)`
returns: array with `{hCard0, hCard1}`

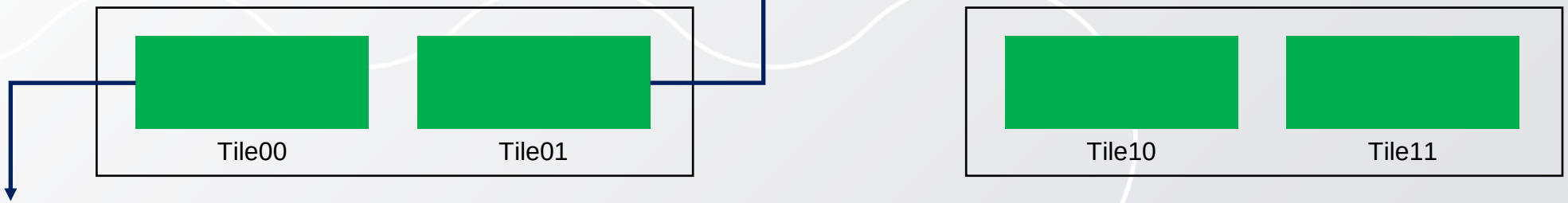


- Multiple queries for a device handle will return always the same device handle (APIs are Getters, not Creators).

Enumerating GPU topology Mode 1: Tiles-as-devices

`zeDeviceGet(hDriver)`
returns: array with `{hTile00, hTile01, hTile10, hTile11}`

Compute:
`zeDeviceGetRootDevice(hTile0)`
returns nullptr (no card available in this mode)

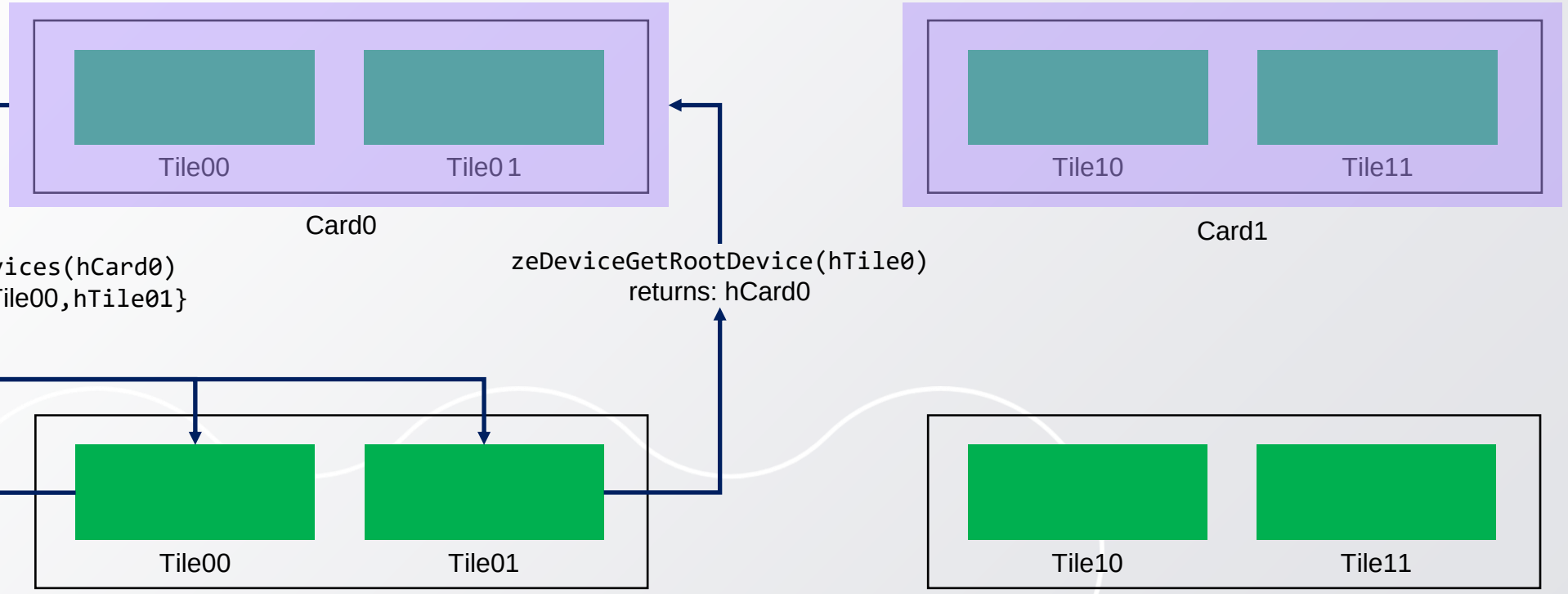


`zeDeviceGetSubDevices(hTile00)`
returns nullptr (no further divisions after tile)

- Device handles belonging to the same card have same BDF (zeDevicePciGetPropertiesExt)

Enumerating GPU topology Mode 2: Tiles-as-devices with Root devices

`zeDeviceGet(hDriver)`
returns: array with {`hTile00, hTile01, hTile10, hTile11`}



Q & A