Run-time PCI Device Reconfiguration

Wesley Shao

Senior Principal Engineer

Oracle

Disclaimer

Presentation Disclaimer: All opinions, judgments, recommendations, etc. that are presented herein are the opinions of the presenter of the material and do not necessarily reflect the opinions of the PCI-SIG®.

Agenda

- Background
- Resource Rebalance
- Requirements
- High Level Design
- Detailed Design

Background

- PCI Resources
  - ✓ Bus numbers
  - ✓ PCI memory space
  - ✓ PCI IO space
- Modern devices require more resources
  - ✓ SR-IOV devices
  - ✓ ARI devices
- Most existing system firmware are not SR-IOV capable
- OS cannot enable devices without resource reservation made by firmware

Resource Rebalance

- Move device resource assignments
  - ✓ Make enough room for additional resource requirement
  - ✓ Enlarge PCI bridges' memory map window if necessary
- Runs within OS
  - ✓ During boot – boot rebalance
  - ✓ During hotplug – hotplug rebalance
- Bus numbers and PCI memory space only
  - ✓ No IO space – design choice

Requirements

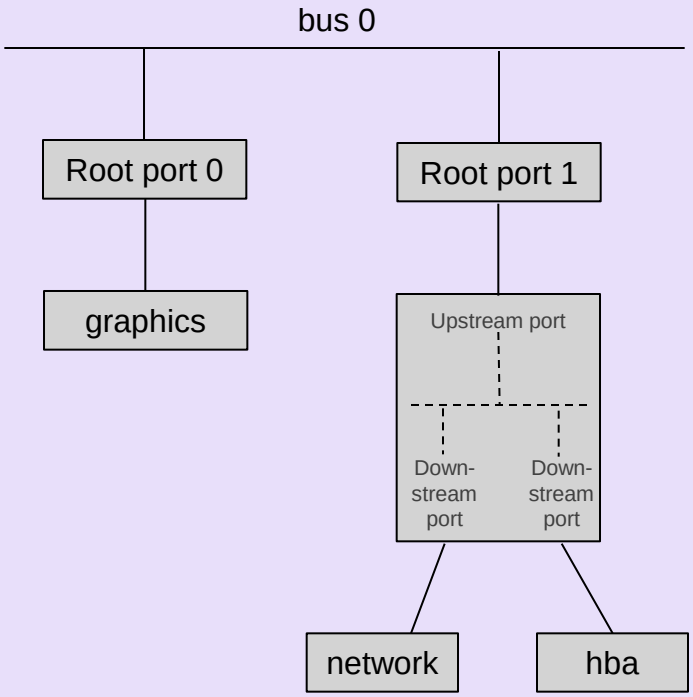
- Alignments defined by the spec
  - ✓ Device BARs
  - ✓ Bridges memory map ranges (base / limit)
- Reasonable complexity
- Fixed resources
  - ✓ USB host controllers, VGA devices (console)
  - ✓ Boot devices
  - ✓ Devices with non-spec compliant behaviors
- Per PCIe fabric
- Batch multiple requests

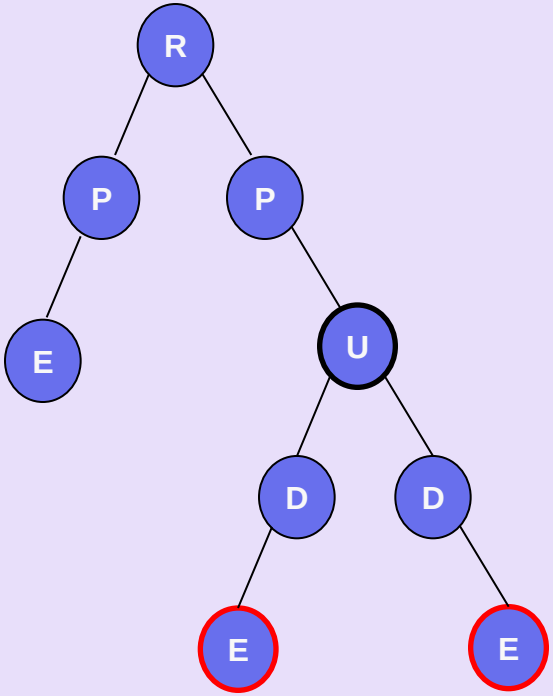
High Level Design (steps)

1) Scan fabric (tree) and identify devices that need more resources than assigned
2) Locate lowest common parent P
3) Combine and merge resource requirements R for P-tree
4) Try to fit R into P's [bridge_base, bridge_limit]
5) If R can fit, assign device resources, DONE
6) If P is already end of the fabric, FAIL
7) P = P's parent, goto step 3
8) Repeat the same steps for each fabric

Lowest Common Parent

- Lowest common parent of all devices that require resource allocation



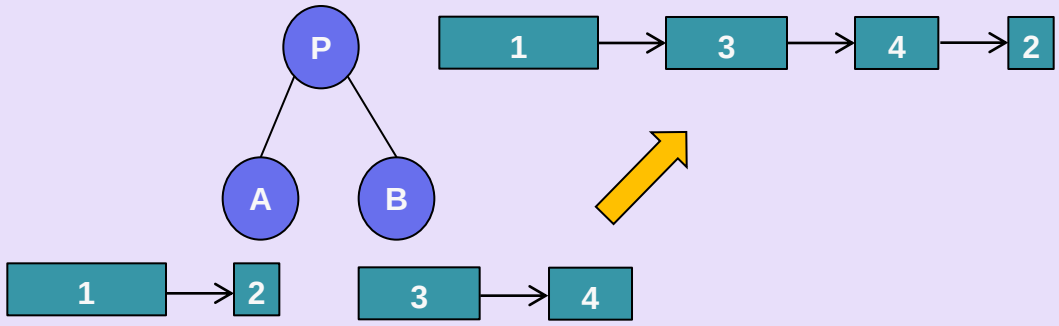


Resources and Ranges

- Resource – BAR, Bus numbers of a PCI function
- Ranges – Collection of Resources and Ranges of a sub-fabric
- Types – Bus, Prefetchable, Non-prefetchable
- Ranges of a bridge node covers all the resources and ranges of its children
- Each node can have relocatable resources, fixed resources, or both
- If all child resources and ranges are relocatable, parent bridge ranges are described by a list of combined relocatable ranges
- All children's fixed resources and ranges are merged into a single fixed resource range

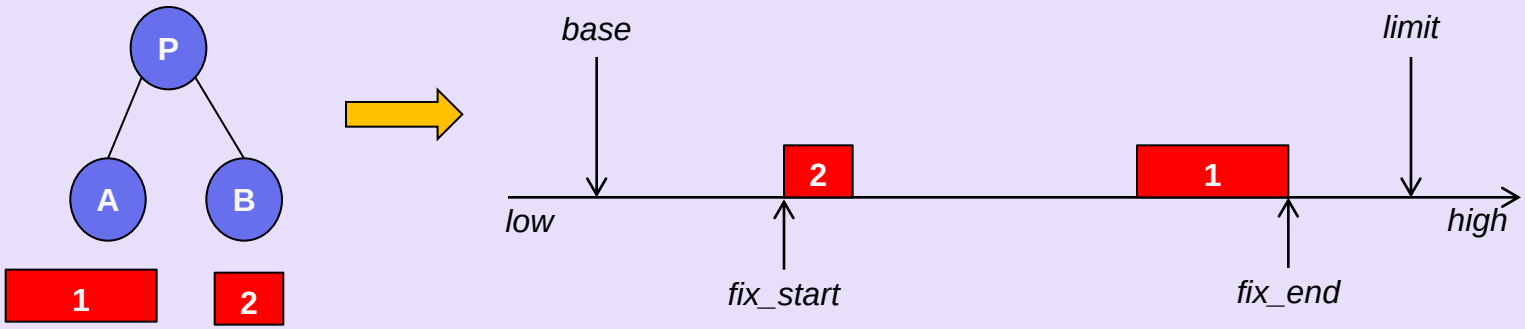
Combine Relocatable Resources & Ranges (step 3)

- For a given bridge node, *combine* resources and ranges of all children and obtain a list of ranges
- *Sort* all the ranges by size of alignment in descending order
- Filling larger requests before smaller ones reduces wasted space



Merge Fixed Resources & Ranges (step 3)

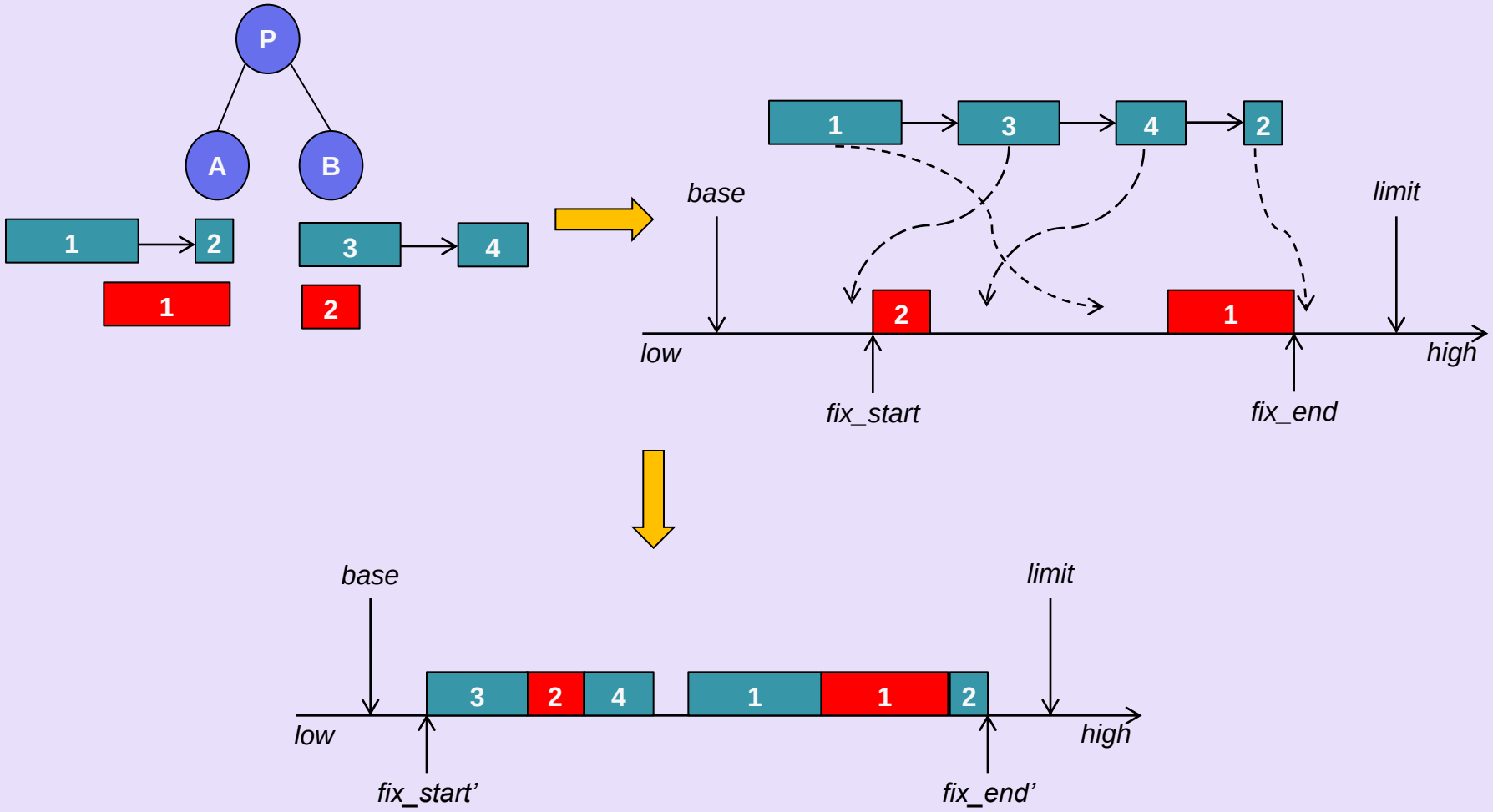
- For a given bridge node, *combine* fixed resources and ranges of all children and obtain a list of ranges
- *Sort* the fixed ranges list by addresses in ascending order
- Observe holes between records on the fixed ranges list



Fill Holes in Fixed Ranges List (step 3)

- A child could be a bridge which requires a single range to cover itself and all of its children
- First examine each child with a fixed range
  - ✓ Use the child's relocatable ranges to *fill* holes on either side of its own fixed range
- Next examine each child with only relocatable ranges
  - ✓ Use the relocatable ranges to *fill* remaining holes
- The parent (bridge) now has one fixed range plus a list of relocatable ranges

Combine and Merge Resources & Ranges (step 3)



Fit all Relocable & Fixed Ranges

- Holes on both ends
  - ✓ [bridge_base, fixed_start')
  - ✓ (fixed_end', bridge_limit]
- Try to *fill* all remaining relocatable ranges (step 4)
- If successful, DONE (step 5)
- If reached end of the fabric, FAIL (step 6)
- Go up the tree (step 7)

Thank you for attending the
PCI-SIG Developers Conference 2011.

For more information please go to
www.pcisig.com