Single Root IOV Endpoint Implementation

Anujan Varma

Denali Software, Inc.

Agenda

- SR-IOV Overview
- SR-IOV components in an Endpoint
- SR-IOV reference implementation
- Applications
  - ✓ Network interface
  - ✓ Storage controller

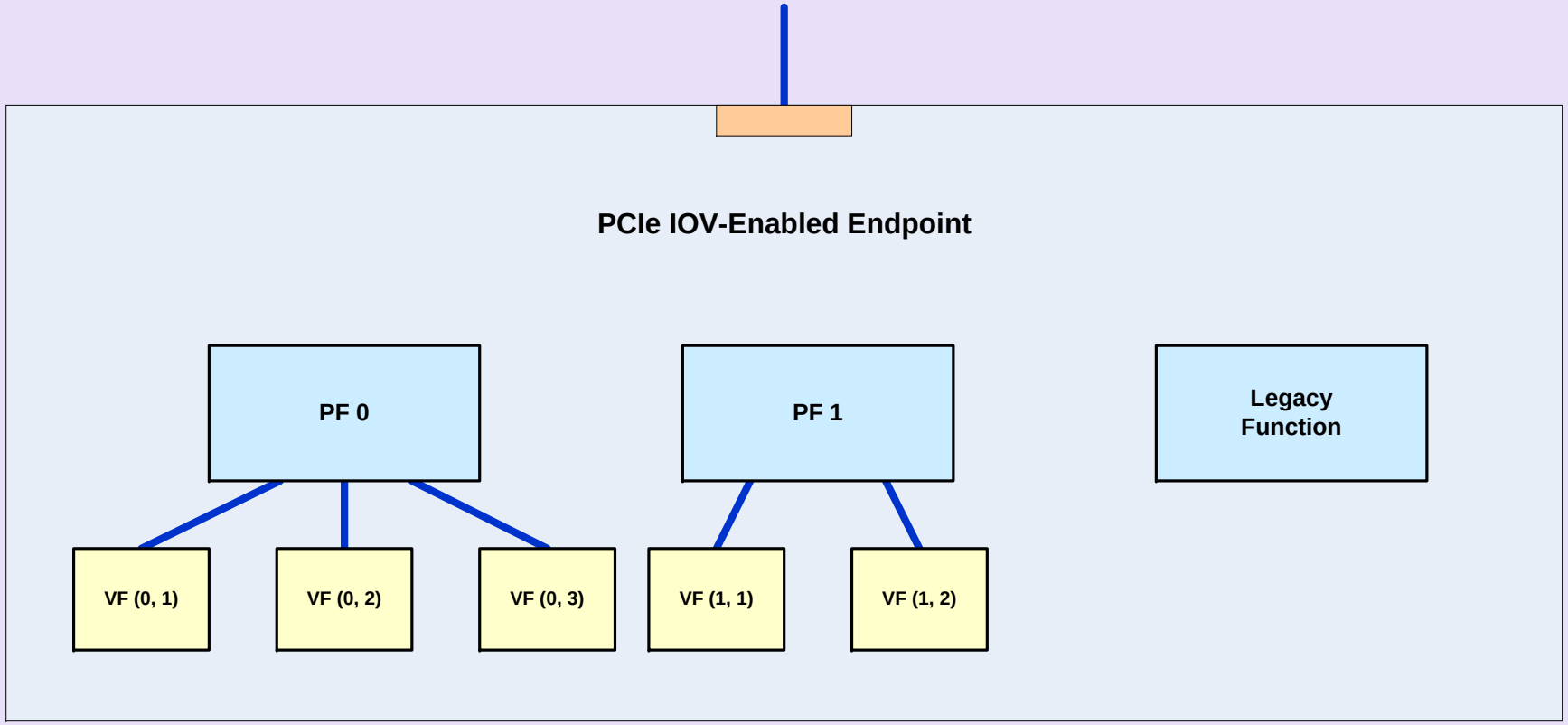
Virtualization

- Enables multiple system images (SIs) to share physical resources (CPU, storage, I/O)
- Current approaches:
  - ✓ Physical partitioning of resources
  - ✓ Hypervisor-based solutions (e.g., XEN)
    - Software virtualizes and manages system resources
    - Hypervisor must intercept all I/O transactions, interrupts, etc.
- PCIe® IOV provides virtual configuration spaces for system images

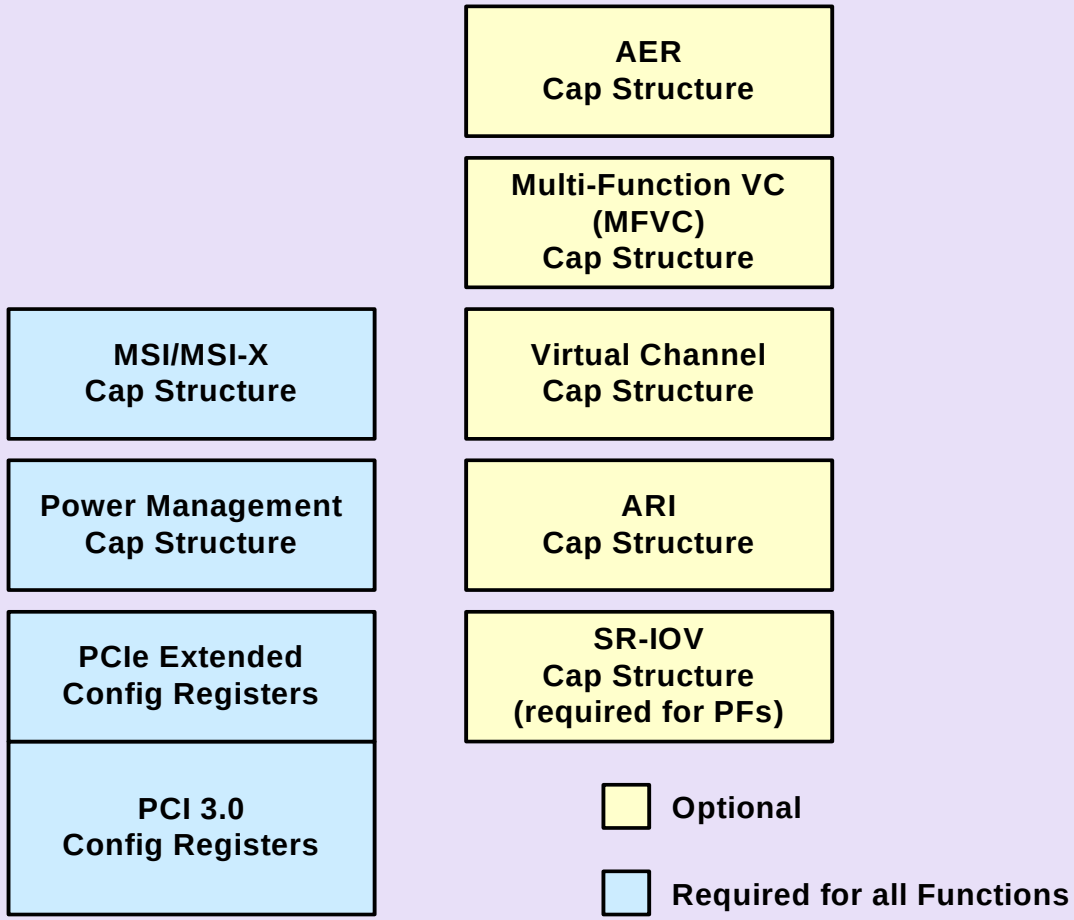
SR-IOV Overview

- Defines hierarchy of Physical/Virtual Functions with single Root
  - ✓ Mix of Legacy Functions, PFs and VFs
- SR-IOV Capability Structure defines VF capabilities associated with each PF
  - ✓ Each VF uniquely addressable with RID
  - ✓ VFs have independent Configuration Spaces and Capability Structures
- May be implemented over PCIe 1.1 or 2.0

SR-IOV Example Hierarchy

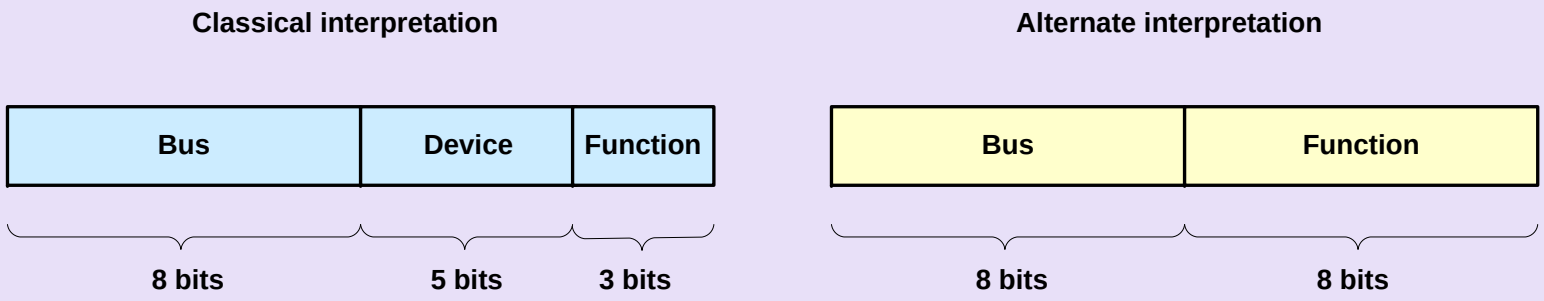


IOV Endpoint Capability Structures



ARI Overview

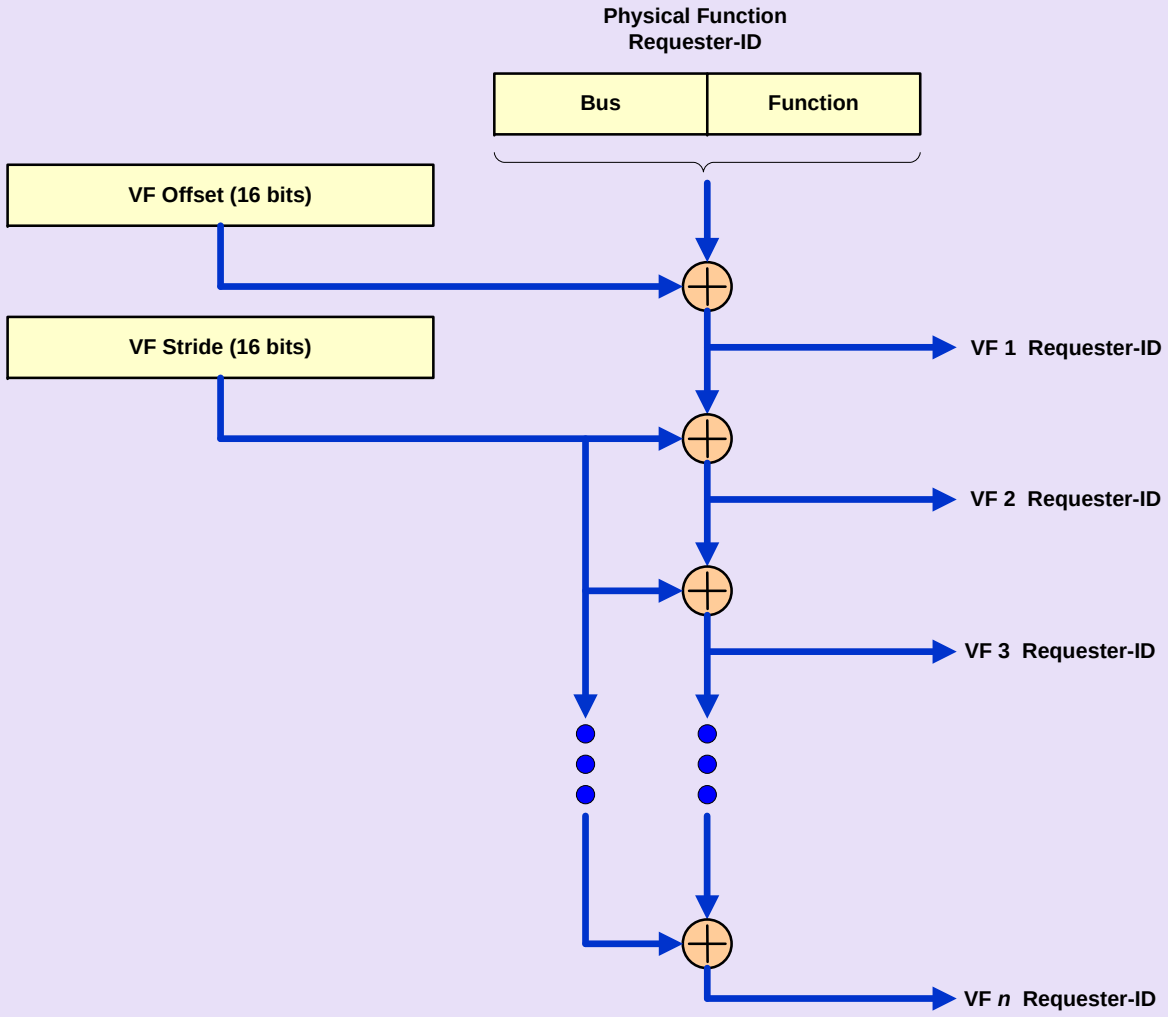
- Modifies Routing-ID format to allow more than 8 Functions
  - ✓ Functions may be of any type (Legacy, PF, VF)
  - ✓ SR-IOV defines a stride-based mechanism for RID assignment to Virtual Functions



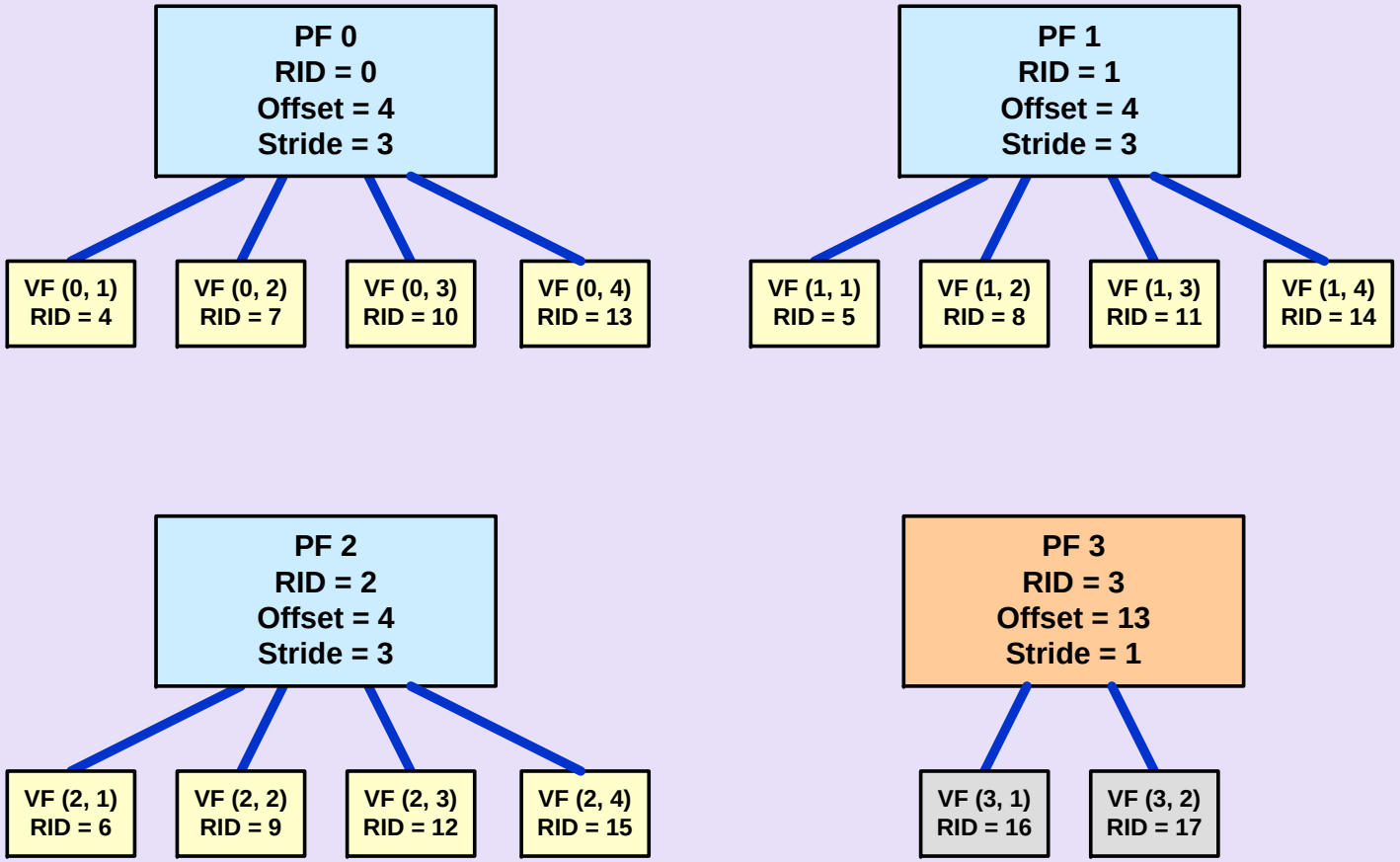
RID Assignment to Physical and Virtual Functions

- VF Routing-ID space need not be contiguous
- VF Routing-IDs assigned in strides relative to the PF they are attached to.
  - ✓ VF RIDs specified by two parameters in Physical Function's SR-IOV Capability Structure
    - 16-bit offset for first Virtual Function
    - 16-bit stride for subsequent Virtual Functions
  - ✓ Offset and stride added modulo 2^{16}

RID Assignment to Physical and Virtual Functions



RID Assignment Example



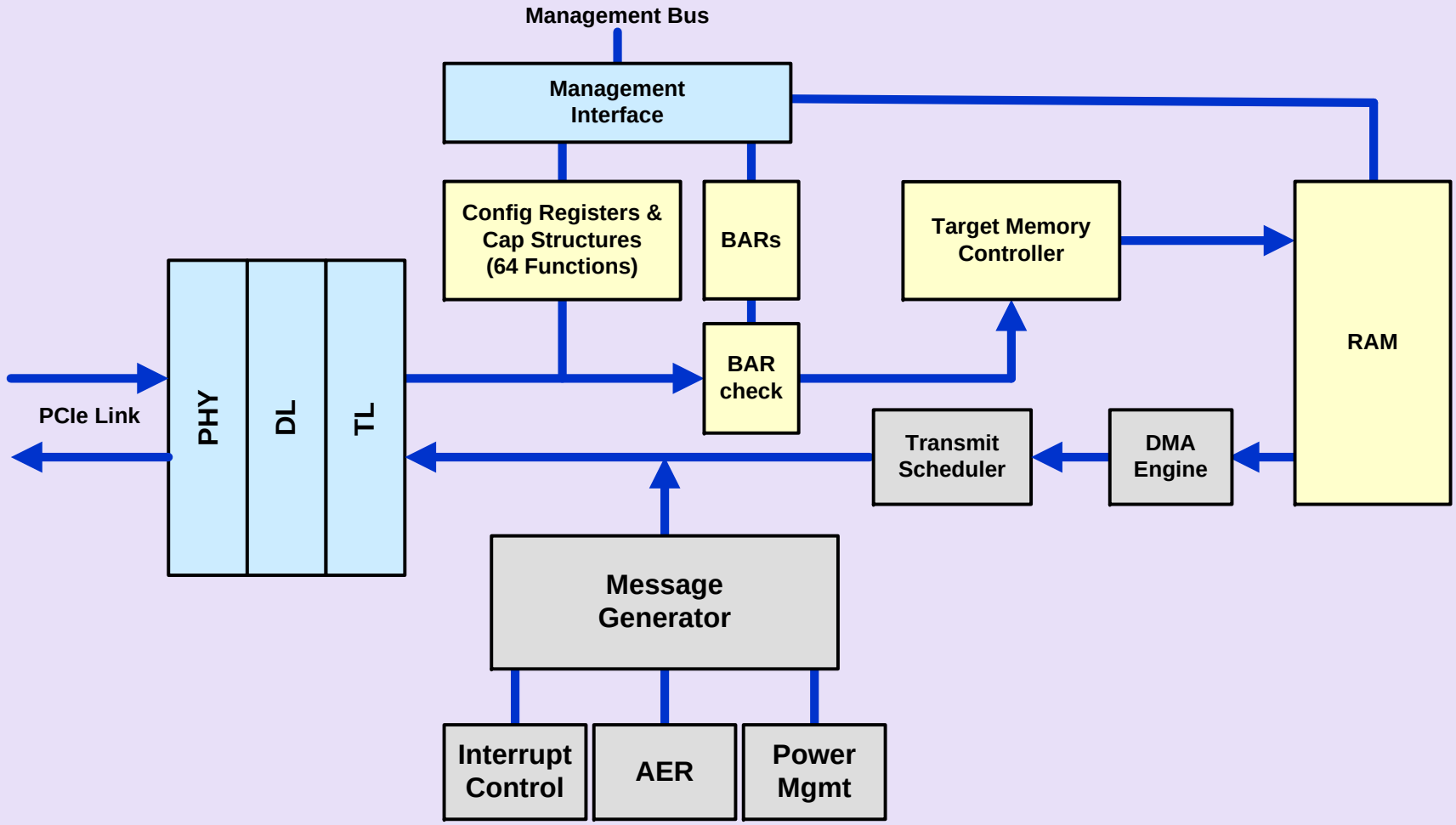
SR-IOV Components

- Configuration registers
- BARs and BAR check logic
- RID decode logic
- Function-Level Reset (FLR) logic
- Interrupt support
- Traffic Scheduler
- Power Management support
- AER support

Reference Implementation

- Based on SR-IOV 0.7 Spec and ARI draft ECR
- Implemented over standard PCIe 2.0 Phy/DL/TL
- 64 Functions
  - ✓ Each may be set up as a Legacy Function, PF or VF
  - ✓ All Functions attached to single Bus Number
- Supports all required and some optional Capability Structures
  - ✓ MSI/MSI-X, AER, PM, ARI
  - ✓ Full set of BARs

Reference Implementation



Configuration Space Implementation

- Register-based implementation inefficient
  - ✓ More than 4K 32-bit registers for 64 Functions
- RAM-based implementation preferred for large number of Functions
  - ✓ Registers accessible only one at a time through addressable port
- Hybrid implementation combines large RAM with small number of registers
  - ✓ Selected control/status bits and fields for all Functions maintained in registers
  - ✓ All other fields stored in RAM

Configuration Space Implementation (continued)

- Register fields divided into 3 types
  1) RAM-based, independent locations per VF
  2) Register-based, independent per VF
  3) Register-based, common for all VFs in a PF

BAR Implementation

- CAM-based implementation with max 64 x 7 = 448 entries
- Two options
  - ✓ Binary CAM
    - Assumes all BARs of same type have same aperture size
    - Multi-phase search
  - ✓ Ternary CAM (TCAM)
    - Allows BAR apertures of Functions to be configured individually
    - Single-phase search

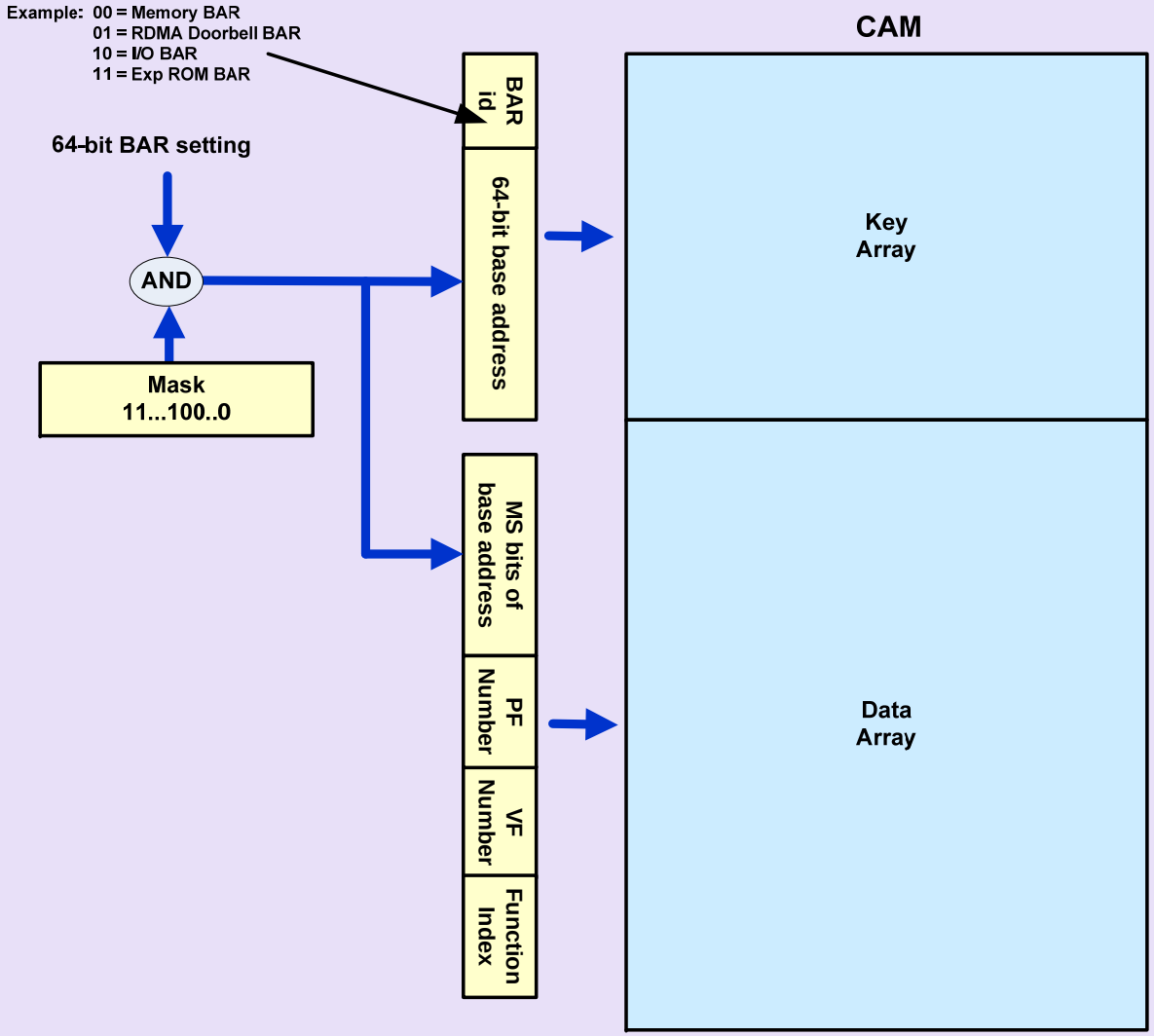
BAR Implementation (continued)

- BARs initialized independently per Function for Legacy Functions and PFs
- Internal logic sets up BARs for VFs on write to associated VF Capability Structure
  - ✓ Returns CRS on write to VF Enable bit during BAR setup
- Less flexible implementation can avoid use of CAM

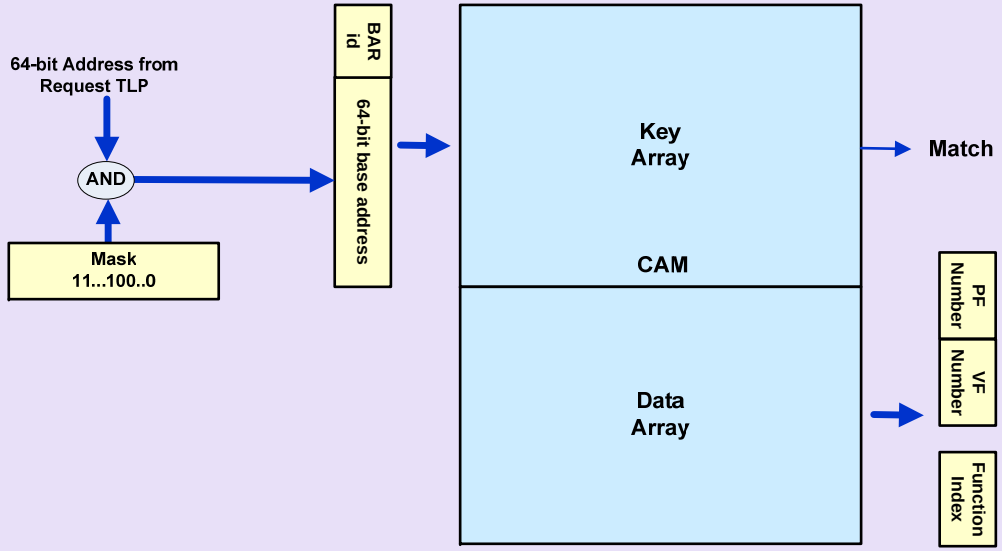
Example BAR Assignment

- Memory BAR (32 or 64 bits)
- RDMA Doorbell BAR (32 or 64 bits)
- I/O BAR (32 or 64 bits)
- Expansion ROM BAR (32 bits)

BAR Implementation with Binary CAM

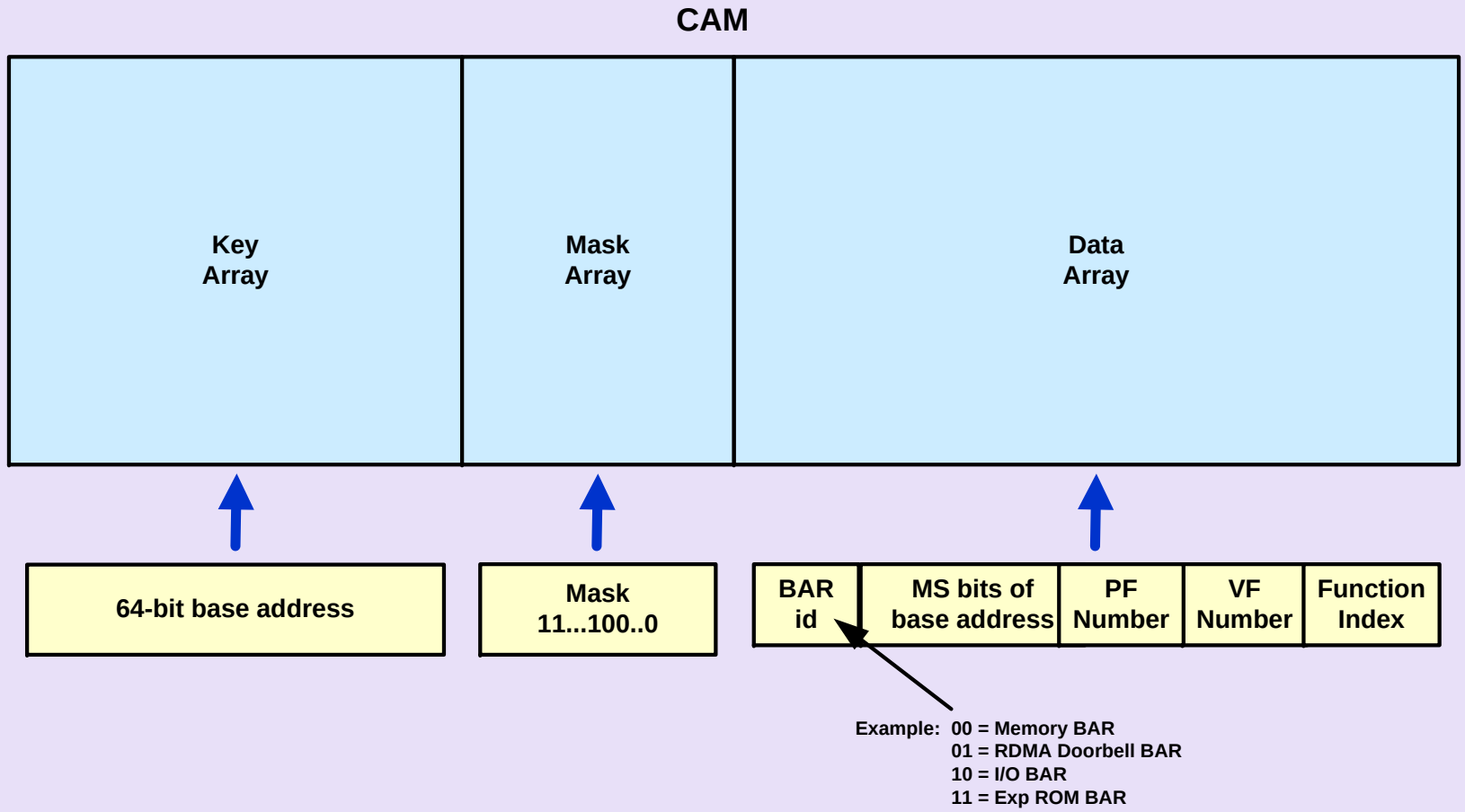


BAR Check with Binary CAM



- Search performed in multiple phases:
  - ✓ Phase 1: Memory BARs of all Functions searched in parallel by setting Mask and BAR id
  - ✓ Phase 2: If no match found, RDMA Doorbell BARs searched by changing Mask and BAR id
  - ✓ If no match, repeat for other BAR types.

BAR Implementation with TCAM



RID Decode Logic

- Reference implementation uses single bus number
- Allows non-contiguous RID assignments
- Logic maps 8 LS bits of RID into internal 6-bit address
  - ✓ Also translated RID into corresponding PF and VF Numbers.

Function-Level Reset (FLR) Implementation

- Performed by HW/SW combination
- Hardware returns Write Completion and signals FLR event to software
  - ✓ Function's BARs disabled while FLR in progress
  - ✓ Invalidates pending requests from Function
  - ✓ RID decoder flags requests to Functions with FLR in progress
    - Returns CRS for Config Requests
    - All other Requests/Completions silently discarded without noting error
- Software updates Config Registers and Capability Structures of associated Function
- Software signals end after FLR sequence completed
- Hardware re-enables Function
  - ✓ RID decoder and BARs for associated Function

Interrupt Support

- All legacy and Physical Functions must support
  - ✓ Legacy interrupts, and
  - ✓ MSI or MSI-X
- All Virtual Functions must support
  - ✓ MSI or MSI-X
- Reference implementation supports MSI interrupts for all, and legacy interrupts for F/PFs

AER Implementation

- AER support optional in SR-IOV
- Errors divided into two categories
  - ✓ Function-specific
    - Individual status bits maintained for all Functions
    - Example: Completion timeout
  - ✓ Common
    - Single set of status bits maintained
    - Example: Phy error
- Some mask bits and enable bits common across VFs in a PF.

Traffic Scheduling in IOV-Enabled Devices

- 4 Levels of hierarchy for traffic management (VF, PF, VC, Link)
- VC-to-link and PF-to-VC scheduling defined by PCIe Specifications
  - ✓ Options of static priorities or Weighted Round-Robin
  - ✓ ARI allows PFs to be grouped into Function Groups for scheduling
- VF-to-PF scheduling is implementation-specific
- Eligibility constraints must be checked:
  - ✓ Transaction order
  - ✓ VC transmit credit

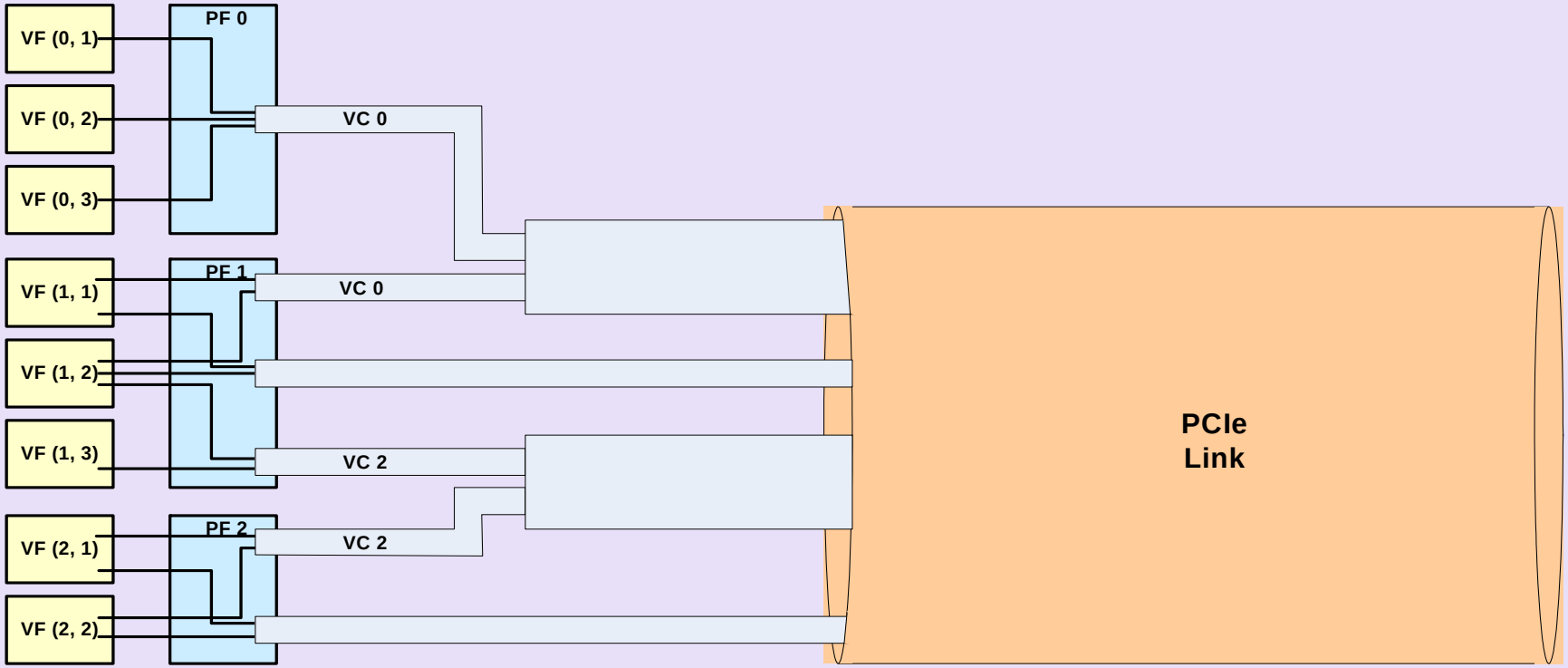
VC Scheduling

- VC-level scheduling defined by
  - ✓ VC Arbitration Table in VC Capability Structure of each Function, or
  - ✓ VC Arbitration Table in common MFVC Capability Structure
    - Defines the VC to be scheduled in each time slot
    - Max 128 entries

Function-Level Scheduling

- Function-level WRR scheduling sequence defined by *Function Arbitration Table* in MFVC Capability Structure
  - ✓ Defines which Function should transmit in a given (virtual) time slot
  - ✓ Function *i* scheduled to transmit in time-slot if
    - VC *j* is scheduled for the time-slot, and
    - Function *i* has an outgoing TLP assigned to VC *j*
  - ✓ Max 256 entries
  - ✓ PFs may be aggregated into Function Groups (with ARI)

Scheduling Model



Power Management Support

- D0 state supported by all VFs, other states optional.
- Only PFs initiate PME events.

Verification Methodology

- Many challenges to validating IOV implementations:
  - ✓ Large number of possible PF/VF configurations
  - ✓ Many scenarios require large number of transactions to be simulated.
  - ✓ Many operations performed by hardware/software combination (e.g., FLR)
  - ✓ Parts of the design (e.g., VF-level scheduling) not defined by spec.

Verification Methodology

- Requires carefully designed test suite
  - ✓ Selected representative PF/VF combinations
  - ✓ Data integrity checking of memory buffers
  - ✓ Assertion checking to verify TL and lower-level functions
  - ✓ Targeted test cases to verify scheduling algorithms
  - ✓ Performance monitoring at VF granularity

Application 1: Network Interface

- PF/VF Hierarchy allows flexible partitioning of network interfaces
- Examples:
  - ✓ Eight physical 1G Ethernet interfaces with one PF per physical interface (does not need SR-IOV)
  - ✓ Eight physical 1G Ethernet interfaces managed as a single PF (one VF per physical interface)
  - ✓ Single 10G Ethernet interface managed as a single PF, and allocated to ten SIs (one VF per SI)
- Enables support of traffic management and bandwidth guarantees
  - ✓ Hierarchical allocation of bandwidth possible

Application 2: Storage Controller

- Controller can implement virtual storage controllers for multiple System Images
- Facilitates management tasks
  - ✓ Interrupts per VF
  - ✓ Power management per VF
  - ✓ Bandwidth management per VF or Function Group
  - ✓ Access Control per VF or Function Group

Summary

- Reference implementation of SR-IOV
  - ✓ 64 Functions with configurable PF/VF hierarchy
  - ✓ General BAR implementation
- Intended to serve as platform for early software development and demonstration

Thank you for attending the
PCI-SIG Developers Conference 2007.

For more information please go to
www.pcisig.com

Single Root IOV Endpoint Implementation

Anujan Varma

Denali Software, Inc.