# GIC Based Interrupt Handling Mechanism on ARM64

dennis.chen@arm.com

Oct. 22th, 2017
CLK 17, BEIJING

# Agenda

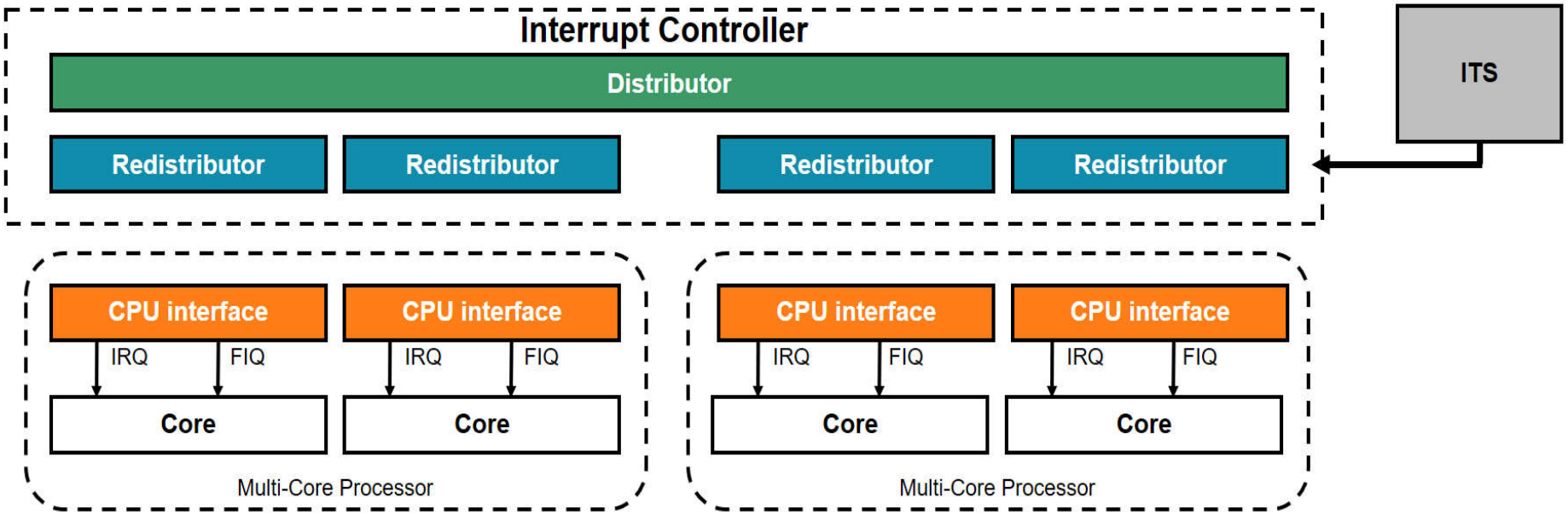
- GICv3/v4 introduction
- Interrupt Handling in Kernel(IHiK)
- Adapt GIC into the IHiK
- PCI MSIx by GIC



arm

# GICv3/v4 introduction

- The Generic Interrupt Controller (GIC) Architecture defines a programmers model for interrupt controllers

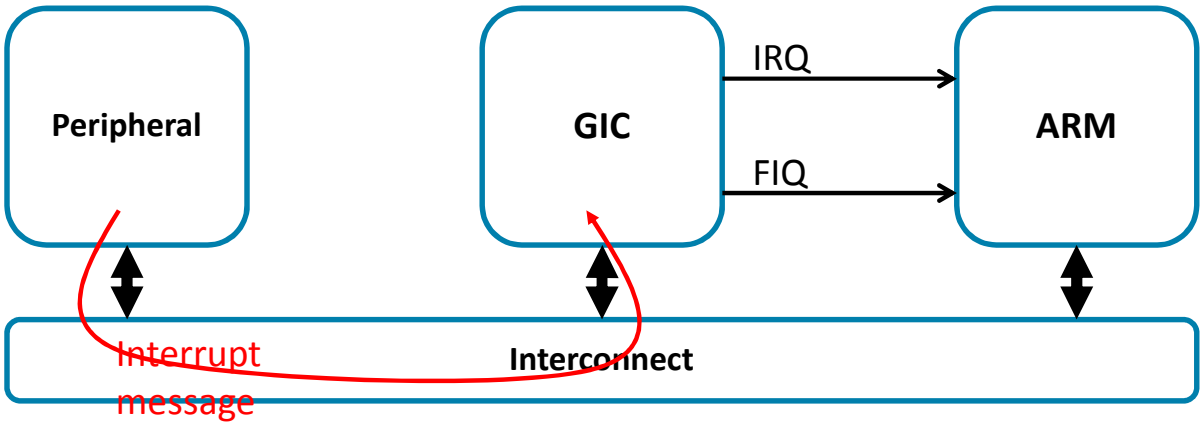





arm

# Message based interrupts - new in GICv3

- **GICv3 adds support for message based interrupts (MBI)**
  - Instead of using a dedicated signal, a peripheral writes a register in the GIC to register an interrupt

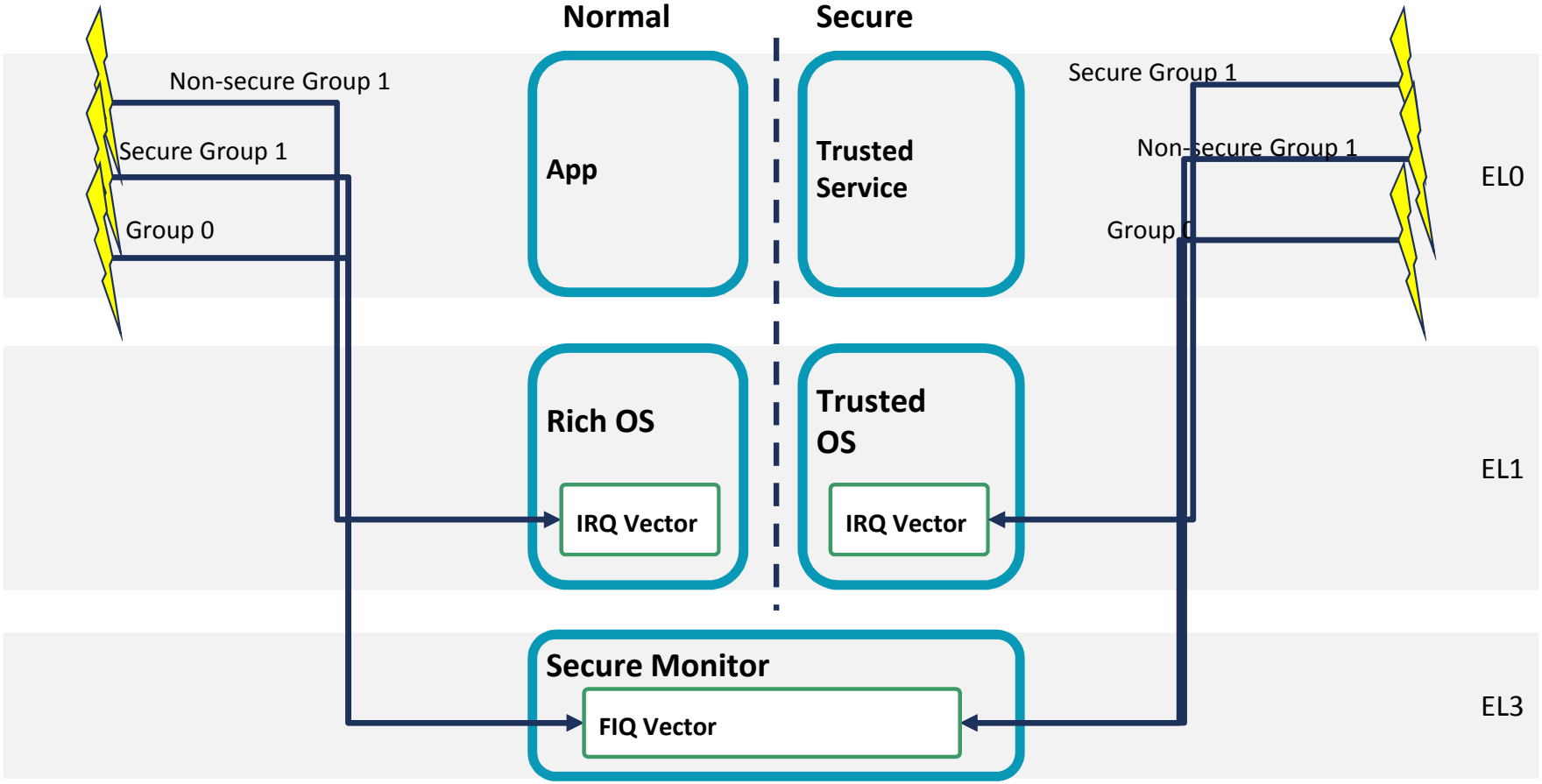



- **Why?**
  - Can reduce the number of wires needed and ease routing
  - Matches model used by PCIe
- **In most cases software should not care whether interrupt is a MBI or not**
  - Does not change how state machine operates



arm

# Interrupt security - example

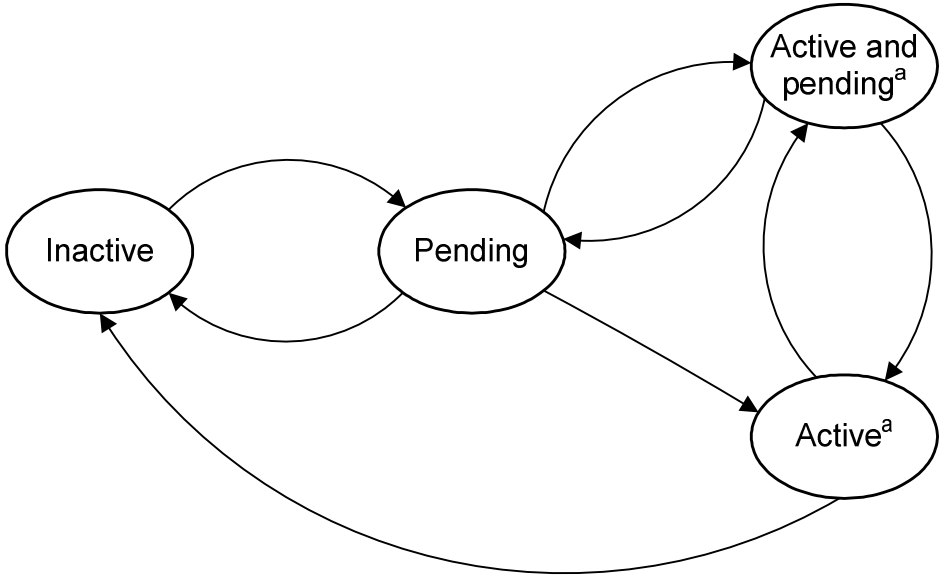





arm

# Interrupt states

- **Inactive**
  - interrupt is not active and not pending
- **Pending**
  - interrupt is asserted but not yet being serviced
- **Active**
  - interrupt is being serviced but not yet complete
- **Active & Pending**
  - interrupt is both active and pending



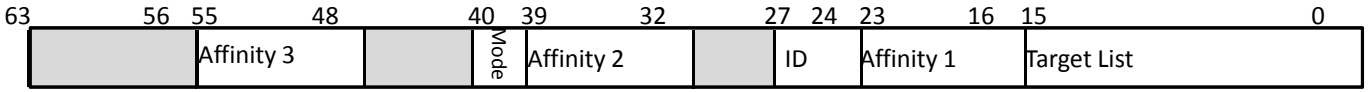

a. Not applicable for LPIs.

- **Interrupt goes:**
  - Inactive → Pending when the interrupt is asserted
  - Pending → Active when a CPU acknowledges the interrupt by reading the Interrupt Acknowledge Register (IAR)
  - Active → Inactive when the same CPU deactivates the interrupt by writing the End of Interrupt Register (EOIR)



arm

# SGI and Processor Affinity




INTIDs 0 to 15 are used for SGIs, which SGI INTID is sent is set by the ID field

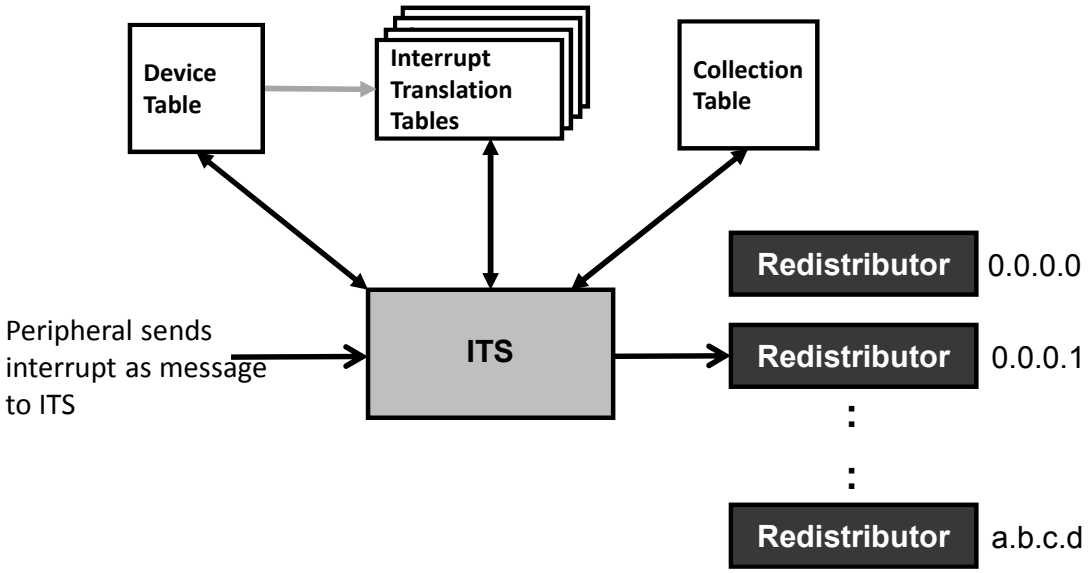
There are two routing modes for SGIs, controlled by the Mode bit

- **`Mode = 0`**
  - Interrupt sent to the cores **`<Affinity3>.<Affinity2>.<Affinity1>.<Target List>`**
- **`Mode = 1`**
  - Interrupt sent to all the cores in the system, except self



arm

# What is an ITS?




- **An Interrupt Translation Service (or ITS) maps interrupts to INTIDs and Redistributors**
- **How is an interrupt translated?**
  - Peripheral sends interrupt as a message to the ITS
    - The message specifies the DeviceID (which peripheral) and an EventID (which interrupt from that peripheral)
  - ITS uses the DeviceID to index into the Device Table
    - Returns pointer to a peripheral specific Interrupt Translation Table
  - ITS uses the EventID to index into the Interrupt Translation Table
    - Returns the INTID and Collection ID
  - ITS uses the Collection ID to index into the Collection Table
    - Returns the target Redistributor
  - ITS forwards interrupt to Redistributor



arm

# Interrupt Handling in Kernel(IHiK)

- **Legacy interrupt handling**

# Interrupt Handling in Kernel(IHiK)

- **Modern interrupt handling**

# Interrupt Handling in Kernel(IHiK)

- **Irq Domain**

# Adapt GIC into the IHiK

- **ACPI MADT**

# Adapt GIC into the IHiK

- **GIC initialization**

# Adapt GIC into the IHiK

- **Irq Domain of GIC setup**

# PCI MSIx by GIC

- **PCI MSIx**



arm

# PCI MSIx by GIC

- **PCI MSIx implemented by GIC**



arm