A Brief Intro to xHCI Driver Model

Xiaofan Li -- xli2@andrew stuff
Agenda

- USB Background
- xHCI Architecture
  - Transfer overview
  - TD (Transfer Descriptor)
- xHCI on FreeBSD
- Proposed xHCI on Plan 9
Not talking about...

- **USB init/attach/detach sequence**
  - It’s boring
  - Just read the steps in manual
  - Many are irrelevant to driver writer e.g. MSI-X

- **Power management**
  - Oh my..
  - No..

- **Different modes of the same operation**
  - Implementation detail
  - Many modes do the same thing but some are more efficient sometimes

- **Slot and endpoints**
  - Important but complicated
  - Many data structure
USB Background

- **SuperSpeed**
  - 5Gb/s 10x over USB 2.0 (480Mb/s half duplex)

- **More bandwidth**
  - split bus

- **Power management**
  - U0 to U3 States

- **Other new features**

- **USB 3.1 -- 10Gb/s … released in 2013**
xHCI Architecture

We are focusing on

But...

It’s nice to understand how the other parts interact with them
xHCI Architecture
xHCI Architecture

Your Phone
All the SMCs
Your SMC
Packages
email from sabrown
UC Basement
Bike, Car or spaceship?
Your new Laptop
Letter from CMU if you failed OS
xHCI Architecture -- TD

- **TD := Transfer Descriptor (aka “package”)**
- **TRB := Transfer Ring Block (aka “stuff in the package”)**
  - Can reference physically contiguous data
  - Use chain bit for noncontiguous data
- **Doorbell Register Array, aka “everyone’s phones”**
- **Uses ring structure so:**
  - Enqueue Pointer
  - Dequeue Pointer
Figure 5: Simple Transfer Example
xHCI Architecture -- TD

- Many uses in the driver
  - Data Transfer
  - Control Transfer
  - Event Transfer

- Many Different modes
  - Periodic (Isochronous, Interrupt)
  - Asynchronous (Control, Bulk)

- Advanced supports
  - Scatter/gather of non-contiguous data (fall on different pages)
xHCI on FreeBSD

- usb.h and pci.h are generic interfaces.
- The driver writes TRBs to different contexts to communicate.
- PCI redirection used when new devices are attached.
- Interrupts passed through to xHCI handler after setup.
xHCI on Plan 9

What’s different about Plan 9?

So, more “self-contained”...

more educational..

less modular..
Where to go from here…

- **Device Contexts**
  - Input/output context
  - Slot context vs. endpoint context

- **Register interfaces**
  - Doorbell register arrays

- **Rings**
  - Event ring usage

- **Scratchpad buffer array**

- **Extended capability**
  - Power management
  - Virtualization support
Reference


FreeBSD on OpenGrok: http://bxr.su/

Plan 9: http://9p.io