

USB 3.0 Software Architecture and Implementation Issues

Terry Moore, CEO
MCCI Corporation
tmm@mcci.com

July 2010

Agenda

- Introducing MCCI
- USB 3.0 from a Software Perspective
- USB 3.0 Software Challenges
- New Device Classes
- Conclusion

Introducing MCCI

MCCI Fact Sheet

- System engineering specialists
 - World Leader in USB software technology
- Worldwide headcount: 160
 - Headquartered in Ithaca, NY
 - Regional development sites near major product development regions
- Focus on high volume consumer products
 - Over 700 million products use MCCI technology
- Leadership in USB-related standards activities
- Test services reinforce customer relationships
 - One of nine USB-IF certified test houses world-wide
 - Microsoft WHQL test services
 - One of three ExpressCard test houses

MCCI Locations



Asia

Noida, India
Chennai, India
Shanghai, China
Taipei, Taiwan
Seoul, Korea
Tokyo, Japan



USA

San Jose, CA
San Diego, CA
Austin, TX
Ithaca, NY



Europe

Birmingham, UK
Nice, France
Stockholm, Sweden

End-to-End USB Solutions

MCCI USB DataPump

- Embedded USB Device/Host/OTG stacks
- USB 2.0 and SuperSpeed USB 3.0
- PictBridge, MTP Stacks

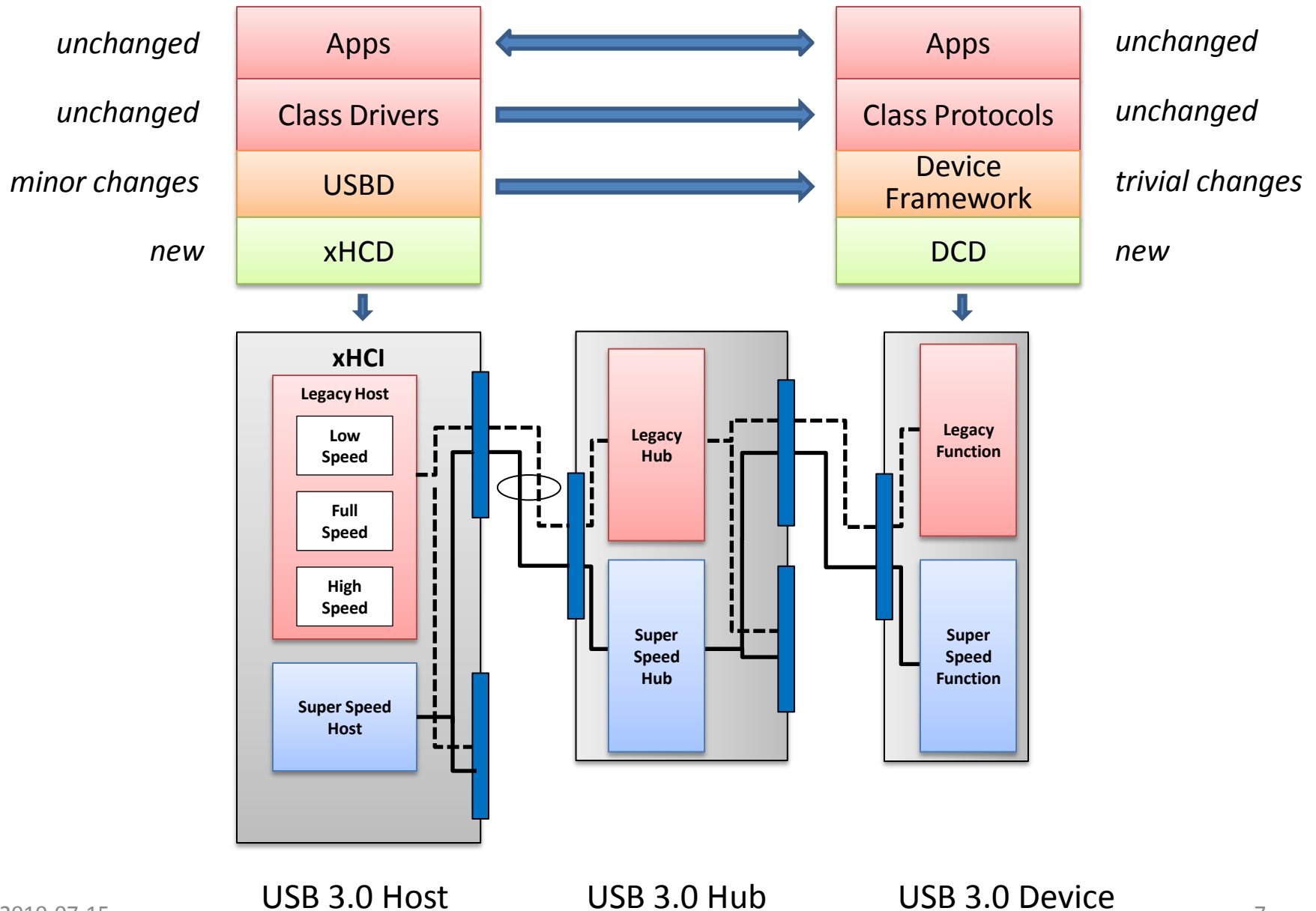
PC USB Host Drivers

- USB 3.0 xHCI Host Stack
- Host Class Drivers for Windows, Mac and Linux

Development & Test Tools

- MCCI Catena HSIC USB and USB OTG Development Kits
- Verification tools for product developers, integrators and testers

USB 3.0 from a software perspective



USB 3.0 Technology Availability

- Products have been shipping since 4Q09
 - Initial deployment in storage devices (HDD, SSD)
 - Roughly one hundred USB-IF certified products to date (June, 2010)
 - Hosts, hubs, and USB 3 disks are in volume production
- All products are using third-party xHCI host stack
 - Microsoft host stack will not ship before Windows 8
 - Microsoft host stack availability will be based on having a variety of device classes and hosts to test with

USB 3.0 Software Challenges

xHCI – Key Software Issues

- Not software compatible with EHCI, OHCI, UHCI
- Single controller supports low, full, high and super speeds
 - Scheduling and bus management handled internally
 - No system memory accesses needed to keep bus running, until data must be moved
- If legacy devices are connected to USB 3.0 port, legacy drivers need to run over the USB 3.0 USBD software stack
 - Drivers from Microsoft
 - Drivers from third parties
- All legacy devices must be tested with xHCI hardware and software
- Billions of legacy devices shipped
 - Test matrix is huge

USB 3.0 Streams

- High throughput may require out-of-order operation
 - Disk drives are the obvious example
 - Inconvenient on USB 2 – because the data over a given endpoint is just sequential.
 - Software has to get involved to set up buffers
- USB 3.0 Streams allow many requests to be sent over a single endpoint concurrently
 - Concurrent requests can be completed by the data source in any convenient order
 - No software intervention is needed at the time data becomes available – data flows directly to the buffer associated with the request
- USB 3.0 host system APIs must be extended to allow a Stream ID to be associated with a given request

Host Class Driver Issues

- USB 3.0 host adds “streams”
 - New API is needed
 - No official standard
- Headache for device vendors who want to provide class drivers
 - Each xHCI vendor may be using their own software stack
 - Product makers who use streams have to negotiate with each software vendor to get access to API, and have to test driver with each target software stack
- One approach: use an “abstract API” that can be mapped onto any host stack
 - MCCI has published an abstract API that is free for anyone to use

Device-Side Changes

- 400 MB/sec theoretical max, bidirectional, so 800 MB/sec is possible with some apps.
- Microseconds matter
 - For disk drives, 300 MB/sec, with 1MB transfers, implies at least 900 transfers per second – depending on protocol, these may be sequential.
 - For sequential access, 1 transfer is therefore 1,111 microseconds
 - If we add 10us delay to each transfer, we therefore have a new transfer time of 1,121 microseconds; we can only do 892 transfers/second, or only 297 MB/sec.
 - If we add 20us delay, to each transfer, we can only achieve 295 MB/sec
 - Competitive disadvantage!
- Going fast requires efficient hardware
 - Flexible DMA engine with scatter/gather and good support for short/zero-length packets
 - Minimize software intervention for “ordinary” events
- Going fast requires efficient software
 - Zero copy architecture
 - Minimize “round trips”
 - Careful CPU cache management
 - Able to tune for bus bandwidth and interrupt mitigation, so that Super Speed USB can co-exist with other applications

Legacy Issues (Devices)

- Devices have to work with USB 2.0 hosts
 - This will be major use case for next year or two!
 - Must include production-quality USB 2.0 device controller hardware
 - If USB 2.0 controller hardware is not integrated with Super Speed hardware, then may need two software stacks
 - Devices have to disconnect from legacy when Super Speed connect is detected
- The USB 2.0 portion must meet USB 2.0 logo test requirements

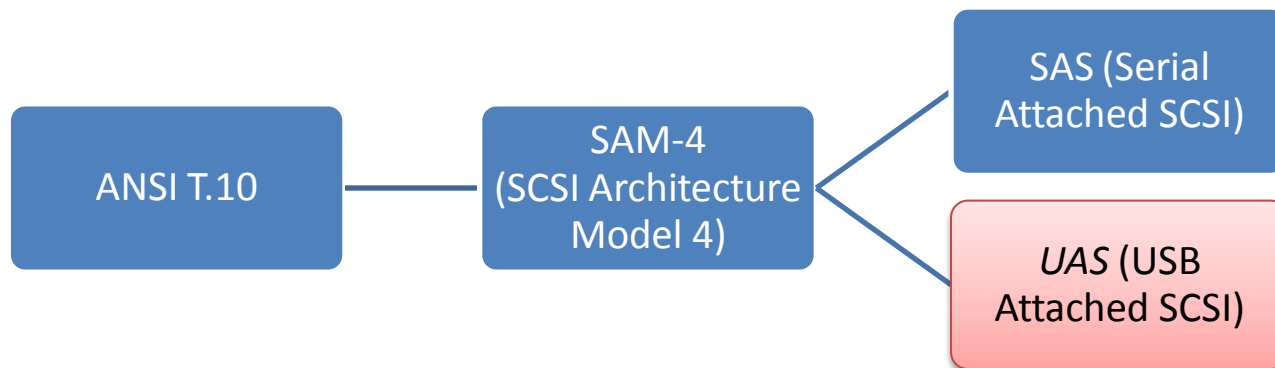
New Device Classes for USB 3.0

Why new device classes?

- Technical needs
 - USB 3.0 speed allows for new applications
 - USB 3.0 speed requires new approaches
- Business needs
 - With USB 3.0, we can replace all the connectors on a netbook with a single USB 3.0 connector (reduce cost, size and weight)
 - eSATA (the UAS class)
 - Ethernet (the NCM class)
 - HDMI, VGA, etc (the AV class)

UAS: USB Attached SCSI

- In order to get maximum benefit of USB 3.0, the mass storage protocol has to be improved
 - Current BOT (Bulk Only Transfer) protocol doesn't allow overlapped operations and out-of-order completion
 - BOT doesn't match modern SCSI architecture



Audio/Video Class

- USB 3.0 is fast enough to carry full HD uncompressed
- New workgroup at USB-IF is developing a standard device class
- Use cases
 - PC or STB host streaming HD content
 - Mobile device streaming MP-4 content
 - Mobile or embedded device using USB display for user interface
 - HD camera streaming live or captured HD content to a display device
- Publication probably in 2H2010
- For technical details prior to publication, join the USB-IF DWG “AV Class” working group

Network Control Model (NCM)

- High performance networking
- Developed initially for 4G (LTE / WiMax) telecomms applications
- Also suitable for transporting gigabit Ethernet (e.g. in docking station)
 - Can take advantage of 800 MB/s bidirectional throughput potential
- Published in August 2009; first errata pending

Conclusion

Conclusion

- Small changes have big implications
 - xHCI Architecture
 - Legacy Support
 - APIs for class drivers
- Microseconds matter
 - 10 microseconds wasted per transfer can mean 1% throughput reduction for common device classes
- New device classes are important for new applications
 - UAS for hard disks
 - AV Class for displays and multimedia
 - NCM for Wireless WAN and docking station support

Thank you!