IMPROVING USB 3.0 WITH BETTER I/O MANAGEMENT

By Sangram Keshari Maharana, Software Engineer Senior, Avineet Singh, Cypress Semiconductor

USB has been popular in the market for its simplicity, maturity and plug-and-play features. However, the 480 Mbps speed of USB 2.0 was not sufficient to support new generation storage and video. Therefore, the time was ripe for migration to a faster standard; this has led to the development of the new USB 3.0 protocol. The challenge that arises for developers is how to leverage USB 3.0's full potential. This article will explore the impact on hardware and software design to implement USB 3.0 with particular focus on handheld products. First, we will compare the capabilities of USB 2.0 and USB 3.0 and the impact of the transition on the components that interact with the USB 3.0 module.

In a common scenario, on the device side, the processor is connected to USB, storage, and peripherals directly. Keeping this architecture in mind, the impact on processor due to the transition from High-Speed to SuperSpeed can be summarized as follows:

System Property	USB 2.0	USB 3.0	Impact on Processor (due to USB 3.0)
Data Rate	Up to 480Mbps	Up to 5Gbps	- Processor should be capable for handling this data rate.
Data Flow	Unidirectional	Bidirectional and simultaneous transfer	- Processor should have rich hardware and software to handle simultaneous transfers.
Power management	Port level suspension and Device level power management	Multilevel link and Function level power management	- Overload on processor to utilize this complex and aggressive power management.
Data Transfer types	Control, Bulk, Interrupt and Isochronous	USB 2.0 types + stream support in Bulk	- The total number of channels on which data can be transferred concurrently increases. Thus, the total number of requests that come from the host per instance increases as well. - Sophisticated software and hardware is required to handle this pressure and to get the real benefits of SuperSpeed data rates.

USB 2.0 vs. USB 3.0

Data Rate

The basic difference between USB 2.0 and USB 3.0 is bandwidth. The theoretical bandwidth provided by USB2.0 is 480Mbps. In reality, the maximum throughput received is about 320Mbps (40MBps), which is roughly two third of the theoretical value. With USB3.0, the raw throughput is 4.8Gbps. If we use the same proportion rate, then the expected data speed is 3.2Gbps (400MBps). However, many developers expect to be able to provide even higher throughput. Figure 1 shows the data rate difference between USB 2.0 and USB 3.0 for a Buffalo external storage disk for different transfer sizes. It should be noted that the USB 3.0 data rate is restricted by the storage device; otherwise a data rate of 400 Mbps can easily be achieved.

Published in EE Times (http://www.eetimes.com) June 2011

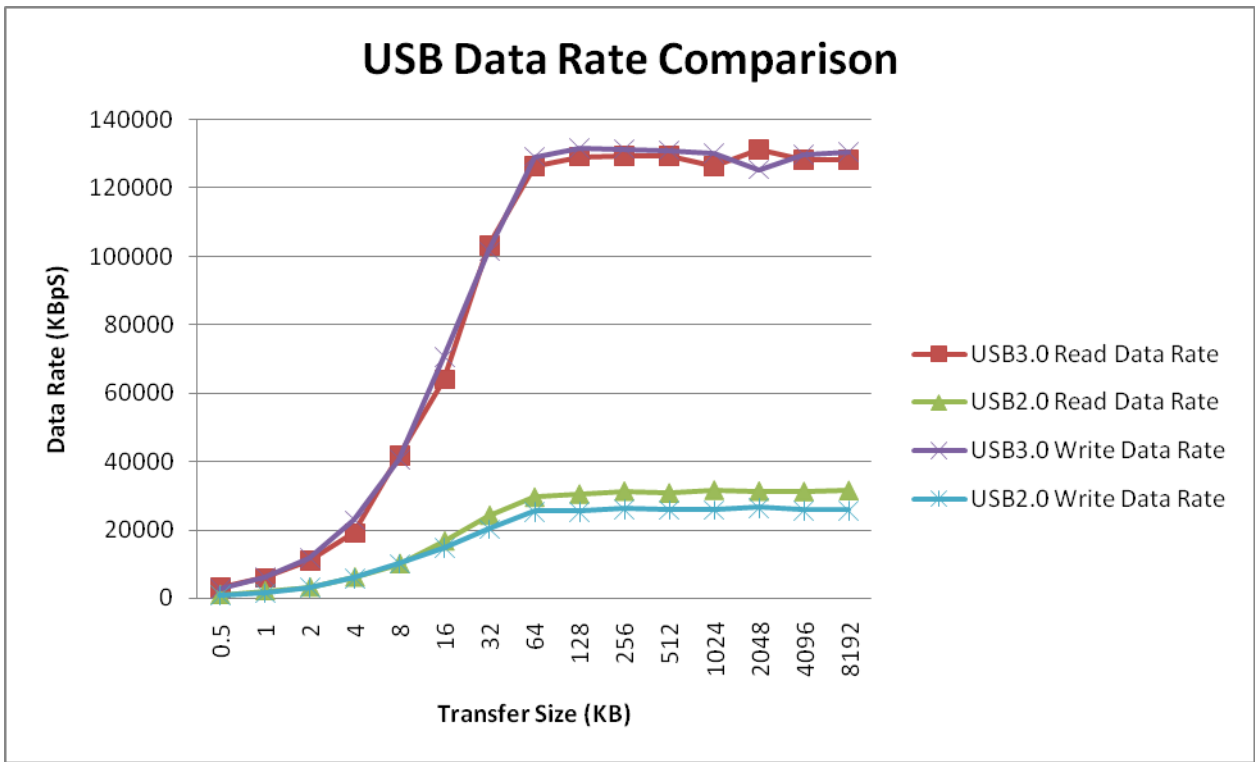


Figure 1

It can be seen that as the transfer size of a single request increases, the data rate increases in tandem. That is because as the transfer size increases per request, the number of requests and hence interrupts that the MSC device has to handle decreases, resulting in better overall performance. After a 64KB transfer size, the data rate attains saturation because the Windows driver does not request more than 64KB data in a single SCSI request. This data shows the importance and effect of interrupts on the overall system performance.

This high data rate increases the interrupt rate and data request rate which can load the processor significantly. While the core is busy processing USB-related real-time requests, latencies increase and users see applications slow down, which is not at all a desirable result.

Data Flow

Unlike the USB 2.0 standard, where data is queued in one direction at a time, USB 3.0 supports simultaneous reading and writing. That is because USB 2.0 is a half-duplex protocol while USB 3.0 is a full-duplex protocol. Full-duplex communications is achieved by adding more connections to support simultaneous data transfers. It also comes at the cost of increasing software complexity twofold as well. With USB 2.0, the processor is involved in only one transaction at a time, and the data structures and request handling are simpler. But with the arrival of full duplex USB 3.0, the data structure will now require double the information. Again the USB software module needs to be able to handle the concurrency in data handling.

Power Management

Changes in the packet transfer protocol (i.e., Broadcast to Directed), elimination of device polling, and definition of link and functional-level intermediate states enables aggressive power management in USB3.0. We will discuss the overhead the processor of the USB device must take on because of the third power reduction-change i.e. the multi intermediate state.

In USB 2.0, the states available are ACTIVE and SUSPEND. SuperSpeed has two more states: FAST EXIT IDLE and SLOW EXIT IDLE. More states mean more complexity in both hardware and software. The device can initiate a power-saving state

using link-level power management. To get the actual benefit, the processor needs to track the idle time in the USB interface and to act more intelligently. The rate of entrance to and exit from power link states can be very frequent for a device. For example, isochronous transfer allows devices to enter low power states between service intervals. This can be a significant addition to the processor's runtime loading.

Stream Support

USB3.0 extends the bulk transfer type with streams. Bulk streams provide in-band, protocol-level support for multiplexing multiple independent logical data streams through a standard bulk pipe. This facilitates the design of complex class protocols over USB. For example, the USB Attached SCSI (UAS) mass storage class uses bulk streams as opposed to the simpler BOT protocol. In BOT, there is only one pending request at a time, where in UAS, there can be n-1 outstanding request at a time, where n is the number of streams supported in the bulk endpoint. Implementing and maintaining a complex class protocol can also keep a processor busy. Where a single flat data-structure was enough for BOT, the UAS protocol demands a priority queue-based data structure to be implemented in the device-side firmware.

Analysis of general USB device architecture

Given that mass storage devices are the most common high-performance USB peripheral available on the market, we will take an example of a mass storage device to formulate a mathematical expression for analyzing performance.

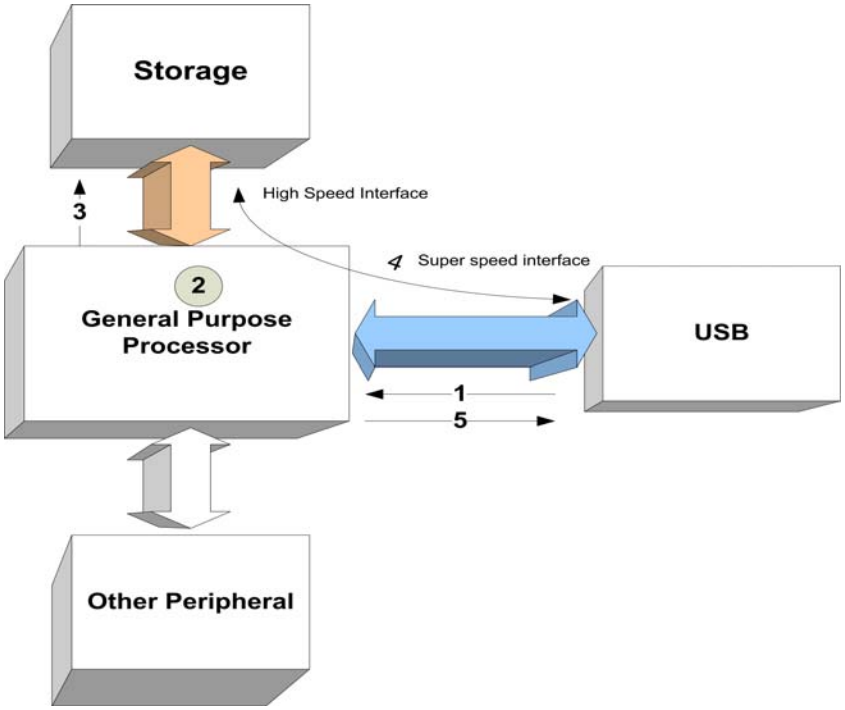


Figure 2

We shall discuss the data phase since most of the time the interface will be involved in transferring data packets rather than control packets.

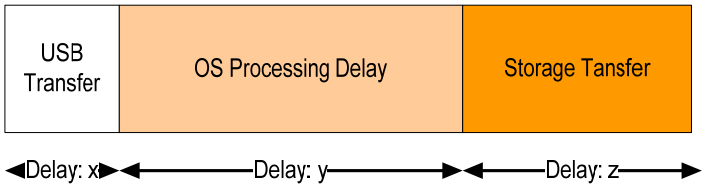
Steps for data transfer:

1. Processor gets a request from USB.
2. Processor processes the request.
3. Processor queues storage read/write request.

4. Processor waits for transfer completion.

5. Processor sends completion status to USB host

Timing notion of this transfer



Total delay = X + Y + Z.

Where, X, Y and Z are the main delay components as explained below:

- Delay X is the amount of time taken for transferring the request data packet between the host and the processor. This depends on the USB protocol and the efficiency of the USB device hardware to handle it. The request packet size is only a few tens of byte, so the delay will be in order of few nanoseconds.

- Delay Y is the amount of time required by the processor to process the USB request and to set up the direct memory access. This depends on the type of processor, number of threads/processes running on it, and the software architecture. For a general purpose processor handling a large number of processes/tasks, the OS processing delay can be very large depending on the interrupt latency, context switch latency, queue latency etc. Worst case delay Y can be on the order of hundreds of microseconds.

- Delay Z is the time required for the data transfer between USB and the storage device, depending on the request type. It also depends on the direct memory access architecture and type of storage device, not on the USB speed as the bottleneck here would be the storage speed and not the USB speed (in case of SuperSpeed). Delay Z can vary from a few microseconds to milliseconds depending on the storage device type and request data size.

Even though the speed of USB has gone up by ten times (from 480Mbps to 5Gbps), the real throughput shall be much less than the theoretical value as USB's contribution(X) towards total delay is negligible in comparison to OS processing delay (Y) and storage transfer delay (Z). Z delay can be improved by adopting better storage devices but Y delay needs to be managed aggressively through more efficient system design.

Efficiency

Utilizing the full potential of USB 3.0 will require the following changes to be incorporated:

- High-performance processor: The complexity and number of tasks to be handled by the processor due to USB 3.0 will increase dramatically. A powerful processor is required if the performance of other applications is not to be compromised.

  IMPACT*:* This will not only add to product cost but also increase the power consumption, which can prove to be a serious disadvantage for handheld devices.

- Existing system architectures would have to be changed to incorporate USB 3.0. Also, storage devices with higher capacities and better performance are required if the full potential of the USB 3.0 is to be realized.

  IMPACT: This will increase the complexity of the system and hence affect time-to-market and project risk.

Redesign to boost the performance

Instead of connecting the USB controller to the general purpose processor (GP), it can be connected to an I/O module. This type of I/O module is called an I/O channel where the I/O module is enhanced to become a separate processor. The GP directs the I/O channel to execute a program in main memory. The I/O channel fetches these instructions and executes these instructions without GP intervention. The GP is only interrupted when the entire sequence is completed.

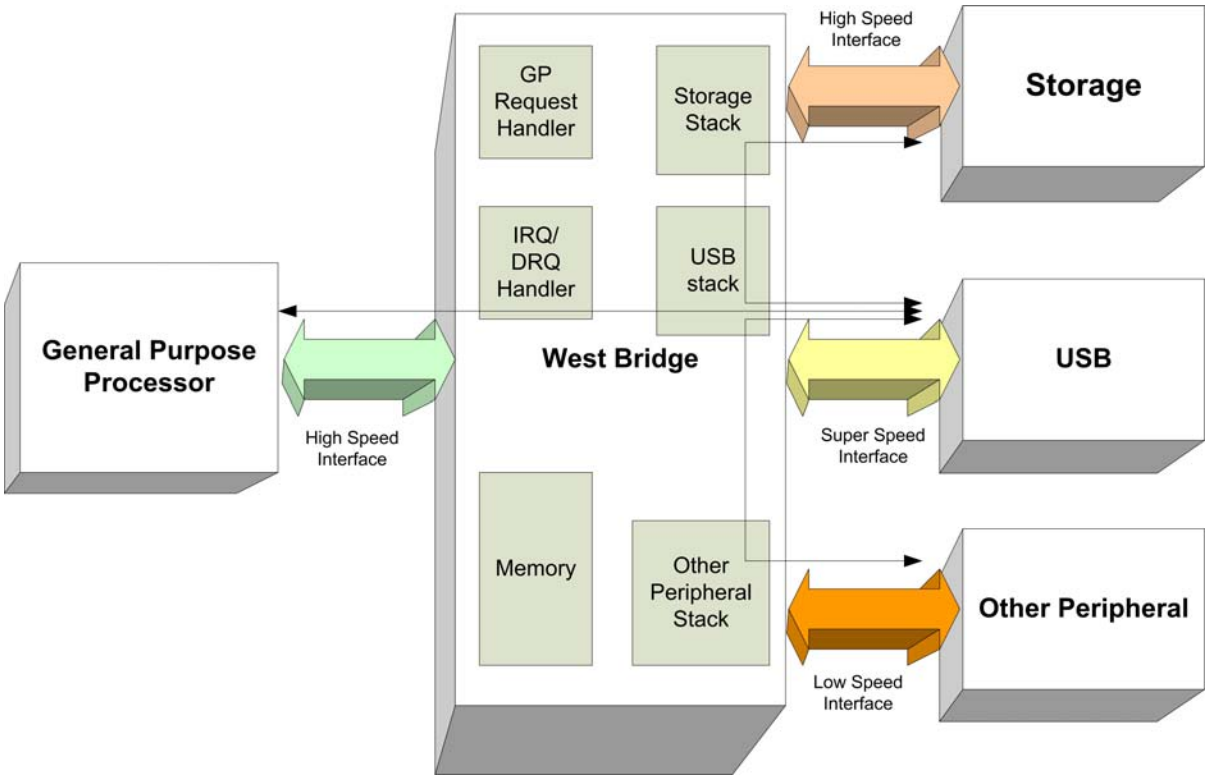


Figure 3

If the I/O module has its own local memory, then it is called an I/O processor. This set-up minimizes the general purpose processor's involvement. This way, the requirement of a high-performance processor and architecture changes can be avoided and thus the unit cost and the risk involved in production can be reduced. The West Bridge is one such intelligent I/O Processor that enhances and modularizes a peripheral controller in an embedded computer architecture. Much in the same way a South Bridge improves data throughput in a PC architecture, a West Bridge topology improves throughput for high throughput data transfers between USB, the general purpose processor, storage, and other peripherals.

The West Bridge device is specially designed for this kind of operation and significantly boosts performance. As the total delay in a data transfer is dependent upon the processing delay, this delay is greatly reduced when a West Bridge architecture is used.

The major factor affecting the performance of a GP depends on the frequency of interrupts. In short, each time an interrupt is received by the GP, a context switch is required and the ISR has to be called, thus increasing the total execution time for other applications running over it. When a West Bridge device is used, most of the USB-specific interrupts are handled by it, thus improving the performance of GP.

A test was performed where a 15.1 GB Embedded Multi Media Card (eMMC) card was enumerated using a mass storage class driver. A comparison was done using the number of interrupts a GP had to handle with and without a West Bridge. The following figure depicts the result for various tasks performed on that system. The individual interrupt numbers are given in $\log_2$ units.

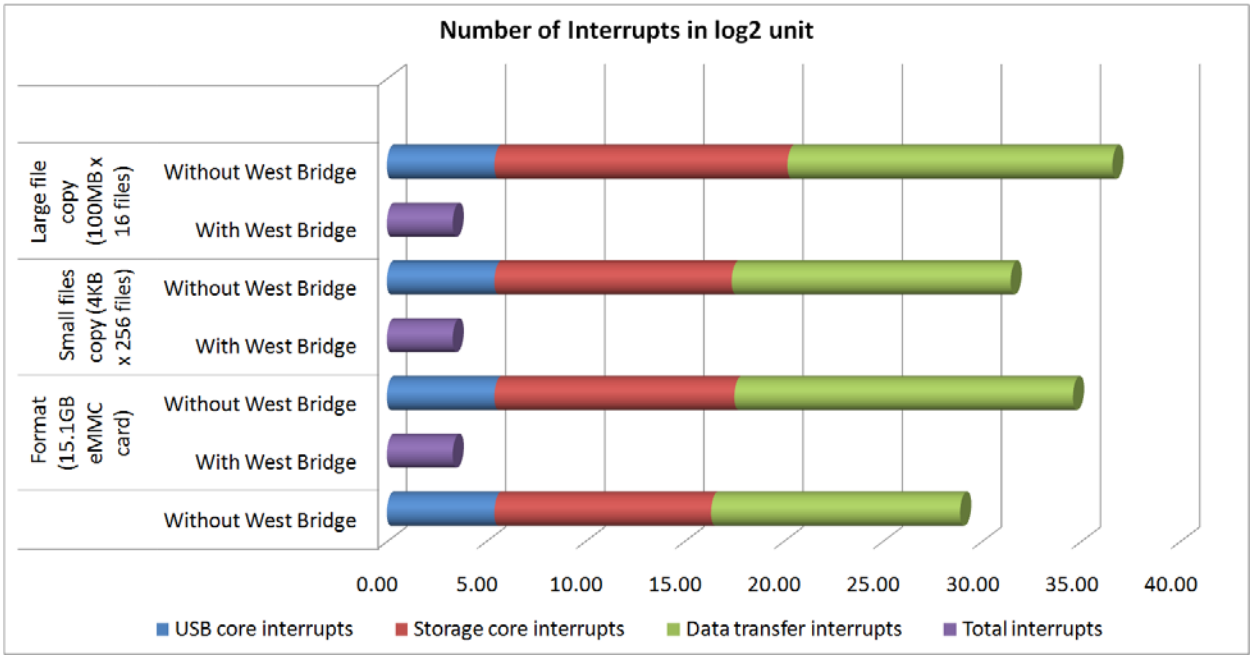


Figure 4

The above table reflects the reduction in the number of interrupts a GP has to handle when it uses a application-specific I/O processor such as West Bridge. Without West Bridge, the GP will have to handle a large number of interrupts, generated at 'super speed' that force the GP to remain in idle state for longer time because of repetitative context switchings. Instead, the GP can offload this responsibility to the West Bridge and maintain its efficiency in handling other real-time tasks while leveraging USB 3.0. Not only does a West Bridge architecture facilitate simplifying the overall architecture of the system platform, it also boosts overall performance and lowers project risk.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709
Phone: 408-943-2600
Fax: 408-943-4730
http://www.cypress.com