Power Savings in USB Hub Through a Proactive Scheduling Strategy

Authors: Bikrant Das Sharma, Saratoga High School, Saratoga, CA bikrantds20@gmail.com
Abdul Rahman Ismail, Principal Engineer, Intel Corporation, and USB-IF Chief Technology Officer
Chris Meyers, Senior Design Engineer, Fresco Logic

Introduction

USB has been the backbone for I/O interconnections in different computing systems over the past two decades. In addition to offering power-efficient performance, it also provides a backwards compatible way for performance growth with interoperability guarantees, while providing low power states to meet the power-performance trade-offs in different applications. With the rising popularity of USB-C ports, USB hubs are gaining prominence for fan-out, even among the high-performance USB 3.x and USB4 devices. With the increasing adoption of these hubs, there are additional power optimization strategies that can be adopted to save power at the platform level without sacrificing performance. Existing mechanisms of turning on low power states due to prolonged inactivity do not save much power when there is a steady bandwidth demand across the devices. This paper proposes an approach where the power savings is commensurate with bandwidth usage, resulting in significant power savings.

Proposed Approach

We propose a proactive approach to scheduling the devices based on their bandwidth demands. The host schedules the transfers in the devices in each Downstream-Facing Port (DFP), taking into account the bandwidth demand in either direction and the available bandwidth of the hub's Upstream-Facing Port (UFP) to optimize the power savings. We also propose some minor enhancements to the existing power management states to maximize the power savings in the entire platform, even in the presence of devices that may not implement the power savings mechanisms well.

Figure 1a illustrates an example configuration where four types of devices are connected to a Hub. The storage device has equal bandwidth demand in each direction, the audio and display devices are dominated by outbound traffic, and the camera has primarily inbound traffic. With existing power savings mechanism, none of the devices or the hub may get into any low power state if each of the devices has a steady stream of I/O traffic even though the link utilization may be low. The link never gets idle for a long enough time to trigger the existing low power state transitions.

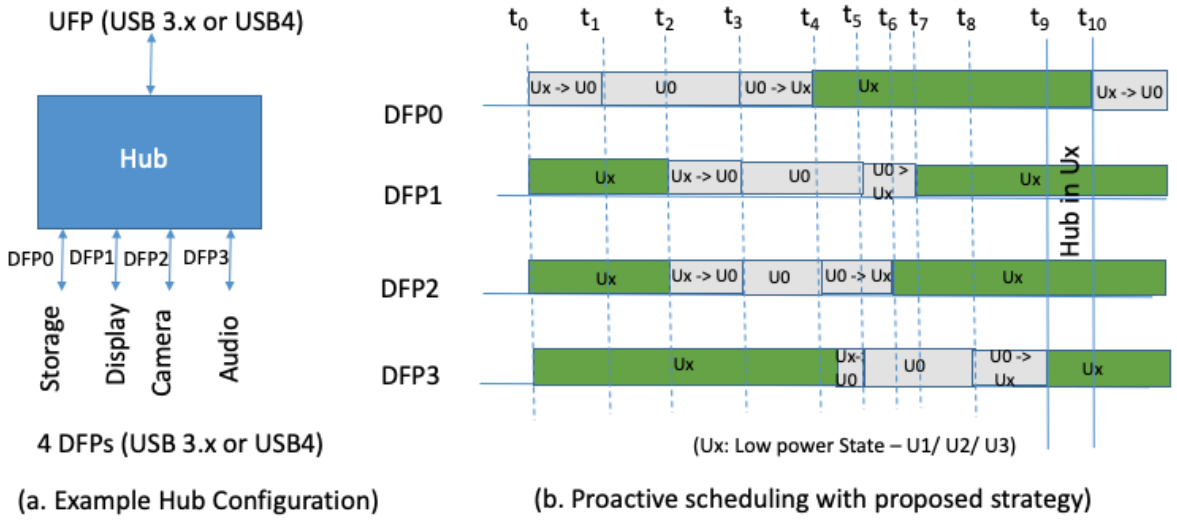
In our proposed approach, the host figures out the bandwidth demand based on the usage and orchestrates the devices to operate in U0 state or in U1/U2/U3 states for USB 3.x devices or CL0$ state or in CL1/CL2/CLd states for USB4 devices. The proposed approach requires a bandwidth tracking mechanism in either direction independently for each DFP by monitoring the transfers scheduled from the host after every scheduling interval to schedule for the next scheduling interval. When a device is

being scheduled for the first time after reset, the host assumes that the device will use the theoretical maximum bandwidth in each direction and schedules accordingly.

Figure 1b represents a sample scheduling interval over time t_0-t_{10} in. Assuming the bandwidth demand of the devices remains the same, the exact scheduling interval t_1-t_{10} will be repeated after t_{10}. If the bandwidth demand changes, a new type of scheduling interval will be implemented following the same algorithm used to determine the scheduling interval t_0-t_{10}.

It should be noted that even though we describe our approach in terms of the USB 3.x power states, one can substitute the corresponding USB4 power states (CLx) for the corresponding USB 3.x power states, Ux.

In this example (Figure 1b), the host through the hub orchestrates DFP0 to exit its low power state Ux->U0 during t_0-t_1. t_0 signifies the start of a new scheduling interval which ends at $t_{10.}$ DFP0 performs its I/O operations in U0 state during t_1-t_3, after which it is scheduled to initiate entry to Ux state during t_3-t_4. DFP0 remains in Ux state from t_4-t_{10}, after which the scheduling interval will repeat. For our analysis, we assume that the only power savings for DFP0 occurs only during this t_4-t_{10} interval. The interval t_1-t_3 satisfies the bandwidth demand of the storage device connected to DFP0 over the t_0-t_{10} interval. While DFP0 is continuing to perform its I/O operations, DFP1 and DFP2 are scheduled to exit from their Ux state to U0 state starting at time t_2, which continues till time t_3. Thus, when DFP0 has completed its I/O at time t_3, the host orchestrates the devices connected to DFP1 and DFP2 to simultaneously start their I/O operations starting at time t_3. While DFP2 completes its I/O operations at t_4, DFP1 continues past t_4 to $t_{5.}$ The host scheduled the I/O operations of the devices connected to DFP1 and DFP2 to overlap since they have traffic in opposite directions and the host can sustain their bandwidths simultaneously. DFP3 has been put to U0 state by t_5 and it continues to perform its I/O operations through t_8 after which it enters Ux state at time $t_{9.}$ Since all devices are in Ux at between t_9 through $t_{10,}$ the hub itself will enter the Ux state and exit to U0 state at $t_{10.}$ Thus, the hub is either in entry to Ux state, in Ux state, or exit from Ux to U0 during t_{9-} t_{10}. While the proposed algorithm seeks to optimize the power savings through scheduling devices simultaneously when their bandwidth demands are within the available UFP bandwidth, it is possible that the UFP link may not be fully utilized while the Hub is in U0 state. For example, in Figure 1b, the UFP link in inbound direction during t_{4-} t_5 may be completely idle while the outbound direction would be 100% utilized since the display device D1 has only outbound traffic. Similarly during the interval t_{5-} t_8 the UFP may have its inbound direction idle since the audio traffic is primarily outbound in nature.



(Net bandwidth consumed by devices connected to DFPs of Hub is less than its available UFP bandwidth)

Figure 1: Example System Configuration and scheduling strategy of our approach: Net bandwidth consumed by the devices connected to the DFPs of the Hub is less than the available BW in its UFP

A second example is demonstrated in Figure 2. Here each of the devices connected to the DFPs can consume the full bandwidth available to the UFP. However, since we have 4 DFPs, each will effectively have a link utilization of 25% even though each DFP could sustain 100% of its bandwidth. Under the existing mechanism, the links will be busy enough even at 25% utilization that no power savings would be possible. With our algorithm, the host will realize that each DFP can have a utilization of 25%. Thus, it schedules each DFP for 25% of the recurring scheduling interval after which it forces the DFP to enter into a low-power state Ux. Since the UFP is always busy serving a DFP, the hub itself does not enter a low-power state. With this algorithm, each device will be in a low-power state Ux for some period of time less than 75% (75% minus the time it takes to enter and exit from the low-power state Ux), which can be substantial.

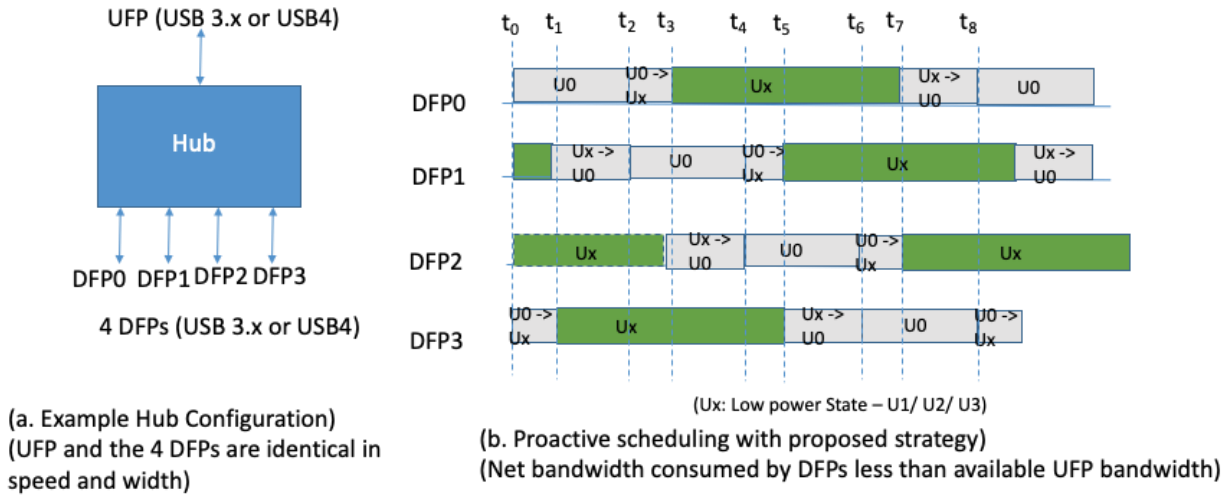


Figure 2 Example System Configuration and scheduling strategy of our approach: Net bandwidth demand by the devices connected to the 4 DFPs is 4 times the available BW in the UFP

The following algorithm describes the proposed approach.

1. Gather/Update the inputs for the algorithm:
    a. Width and Speed for UFP: UFP width (*ufp_width*), UFP frequency (*ufp_freq*)
    b. Number of downstream ports (*num_dfp*) that are populated
    c. Power parameters for each USB port type (USB3.x, USB4) for each of the low-power states (U1/U2/U3) for each newly added port: the entry and exit times to/from each Ux, as advertised initially, the power consumption in each Ux state, and the power consumption in U0
    d. Number of downstream ports (*num_dfp*)
    e. Width, Speed, and Bandwidth demand for each DFP Port i (0 <= *i* < *num_dfp*): width (*dfp_width[i]*), frequency (*dfp_freq[i]*), projected bandwidth demand in each direction (*dfp_bw_demand_ib[i], dfp_bw_demand_ob[i])*
    [The projected bandwidth demand in each direction for a DFP represents how much actual bandwidth the device connected to the DFP of the hub will consume. For example, a webcam connected to a hub DFP may have a projected bandwidth demand of 80 Mb/s inbound and 0.1 Mb/s outbound.]
    f. For each newly added DFP i (0 <= *i* < *num_dfp*): width (*dfp_width[i]*), frequency (*dfp_freq[i]*). Initialize the projected bandwidth demand in each direction (*dfp_bw_demand_ib[i], dfp_bw_demand_ob[i])* to the theoretical maximum bandwidth of the DFP. i.e., *dfp_bw_demand_ib[i] = dfp_bw_demand_ob[i] = dfp_bw [i]*
    g. Calculate the theoretical maximum bandwidth of the UFP and each of the DFP ports, adjusted for encoding overhead (*ufp_bw*, and *dfp_bw [i]*)
    h. Calculate the *tot_dfp_bw* as the sum of the *dfp_bw[i]* for each DFP
    i. Pick the largest *scheduling_interval* that would still meet the maximum tolerable latency (irrespective of bulk or isochronous traffic) for each DFP
2. Calculate the fair bandwidth allocation to each DFP, as a fraction, assuming each DFP wants to stream full bandwidth, irrespective of its current bandwidth demand. This should be proportionate to the width and speed (theoretical bandwidth) of the DFP. *frac_bw_allot [i] = dfp_bw[i]/ tot_dfp_bw*
3. Initialization of variables to be used:

a. *frac_bw_allot_ib_after_first_pass = 1.0*
b. *frac_bw_allot_ob_after_first_pass = 1.0*
[The above two variables represent the fraction of hub UFP bandwidth to be allocated after the first pass in the inbound/outbound direction.]
c. *tot_bw_allot_ib=0*
d. *tot_bw_allot_ob=0*
[The above two variables represent the actual hub UFP bandwidth allocated thus far in each direction.]
e. *bw_remaining_after_first_pass_ib=ufp_bw*
f. *bw_remaining_after_first_pass_ob=ufp_bw*
[The above two variables represent the UFP bandwidth remaining after the first pass in each direction.]

4. Pass number one of the two pass bandwidth allocation for DFP *(i)* for each direction (inbound and outbound). If Bandwidth Demand *dfp_bw_demand_ib(ob)[i] <= frac_bw_allot[i]*ufp_bw:*
[Since the projected bandwidth demand is less than the proportionate bandwidth the device needs in that direction, it is allotted in the first pass – that way we will not allot more than what the device needs, resulting in unused bandwidth. Whatever remains will be allotted proportionately to the remaining devices, in each direction independently, which would be more than or equal to the allocation they would have received if we allotted everything in one pass.]
   a. Allot the full projected bandwidth demand: *dfp_bw_allotted_ib(ob)[i] = dfp_bw_demand_ib(ob)[i]*
   b. *allocated_bw_ib(ob)[i] = dfp_bw_demand_ib(ob)[i]*
   c. *frac_bw_allot_ib(ob)_after_first_pass = frac_bw_allot_ib(ob)_after_first_pass - frac_bw_allot[i])*
   d. *tot_bw_allot_ib(ob) = tot_bw_allot_ib(ob) + dfp_bw_demand_ib(ob)[i]*
   e. *bw_remaining_after_first_pass_ib(ob) = bw_remaining_after_first_pass_ib(ob) - dfp_bw_demand_ib(ob)[i]*
5. Pass number two of the two pass bandwidth allocation for each DFP *(i)* for each direction (inbound and outbound) which was not allotted in Pass 1 (i.e., *dfp_bw_demand_ib(ob)[i] > frac_bw_allot[i]*ufp_bw)*
   a. *allocated_bw_ib(ob)[i] = (frac_bw_allot[i] / frac_bw_allot_ib(ob)_after_first_pass) * (bw_remaining_after_first_pass_ib(ob)*
6. Calculate the Utilization and project bandwidth savings for each DFP:
[Utilization denotes the fraction of time the corresponding link is doing actual data transfer as opposed to sending idle packets.]
   a. *dfp_util_ib(ob)[i] = allocated_bw_ib(ob)[i]/dfp_bw[i]*
   b. *dfp_util[i] =* higher of { *dfp_util_ib[i], dfp_util_ob[i]}*
   c. *Fraction in Ux*: [{ *scheduling_interval - (dfp_util[i]*scheduling_interval + UxEntryTime_DFP[i] + UxExitTime_DFP[i])} / scheduling_interval]*
   d. Projected Power Savings for Ux: *power_Ux_dfp[i] = Fraction in Ux * (U0 Power[i] – Ux Power[i])*
7. Calculate the utilization and projected bandwidth savings of UFP:
   a. Add the *allocated_bw_ib(ob)[i]* for each DFP and pick the larger of the two (ib vs ob) as *allocated_bw_ufp*
   b. Link utilization of UFP: *ufp_util = allocated_bw_ufp/ ufp_bw*
   c. *Fraction in Ux =* [{ *scheduling_interval - (ufp_util*scheduling_interval + UxEntryTime_UFP + UxExitTime_UFP)} / scheduling_interval]*
   [This variable denotes the fraction of time the Link will be in Ux power savings state.]
   d. Projected Power Savings for Ux: *power_Ux_ufp = Fraction in Ux * (U0 Power – Ux Power)*
8. Use the highest net power savings (both UFP and all DFPs) Ux using a greedy scheduling algorithm as follows if the power savings is greater than 0

9. Read the Ux entry and exit time of each DFP and the UFP in the Hub register, if available, and update the *UxEntryTime_DFP[i], UxExitTime_DFP[i], UxEntryTime_UFP, UxExitTime_UFP.* Set up the registers for the next round of readings if needed. (Note that variables such as UxEntryTime_DFP[i] will also be used for the corresponding CLx states if the device is a USB4 device)
10. Read the bandwidth used in each direction in the Hub DFP and update the *dfp_bw_demand_ib[i]* and *dfp_bw_demand_ob[i]* to the minimum of actual measured usage in the corresponding direction multiplied by 1.05 and *dfp_bw[i].*
    [The multiplier 1.05 (or any other value greater than 1) is used to allocate slightly higher bandwidth than previously used to enable a device to increase its bandwidth usage going forward.]
11. If there is a hot-plug, go to Step 1b; else go to Step 2

The following example explains the algorithm above.

Example 1: Consider a hub with a USB4 UFP which is 2 Lanes of 20 Gb/s. Thus, the *ufp_bw* is 40 Gb/s. There are 4 DFPs (*num_dfp* = 4), with the width, speed and bandwidth demand as shown in the first 3 rows of Table 1. Initially, the scheduler assumes the bandwidth demand in the DFPs to be 20 Gb/s, 4 Gb/s, 40 Gb/s and 40 GB/s respectively. However, after the first scheduling interval, Rows 2 and 3 in Table 1 have been updated to the values from actual measurement in the Hub DFP. The total DFP theoretical bandwidth (*tot_dfp_bw*) is 104 Gb/s versus the UFP theoretical bandwidth of 40 Gb/s against a net projected bandwidth demand of 84 Gb/s inbound and 74 Gb/s outbound. Note that since DFP2 is a USB 3.0 device (single Lane at 5.0 Gb/s), its dfp_bw[2] is listed as 4 Gb/s, taking into account the 8b/10b encoding overhead. For the sake of simplicity, we are ignoring the 3% overhead of 128b/132b encoding with USB 3.1, USB 3.2, and USB4 here.

Each DFP gets a fractional bandwidth allocation, proportionate to its link and width based on Step 2, shown in row 4 (*frac_bw_allot[i] = dfp_bw[i] / tot_dfp_bw*). It should be noted that the fractional bandwidth allocations add to 1. The UFP will allocate its bandwidth to each DFP based on this fractional bandwidth allocation and actual bandwidth demand in each direction (inbound and outbound).

During the first pass of bandwidth allocation, if a DFP's inbound or outbound projected bandwidth demand is less than or equal to its fractional bandwidth allocation, it is allocated the projected bandwidth demand in that direction, as described in Step 4, and shown in rows 5 and 6 in Table 1. The total bandwidth allotted after first pass is 0 Gb/s inbound (*tot_bw_alllot_ib)* and 2.4 Gb/s outbound (*tot_bw_alllot_ob).* The bandwidth remaining after first pass is 40 Gb/s inbound and 37.6 Gb/s outbound (*bw_remaining_after_first_pass_ib* and *bw_remaining_after_first_pass_ob* respectively). The *frac_bw_allot_ib_after_first_pass* and *frac_bw_allot_ob_after_first_pass* after first pass are 1.0 and 0.769231 respectively.

During the second pass of bandwidth allocation, each of the DFP's inbound and outbound direction that has not been allotted bandwidth during the first pass, is allocated bandwidth from the remaining bandwidth in each direction (*bw_remaining_after_first_pass_ib/ob)* based on its fractional bandwidth allocation (*frac_bw_allot[i])* and the *frac_bw_allot_ib(ob)_after_first_pass,* as described in Step 5, and shown in rows 7 and 8 in Table 1 below. The link utilization of each DFP (inbound, outbound, and overall) is computed based on Step 6 and shown in row 9 in Table 1. The resulting power savings for each device in the DFP is shown in row 10, with U1 state, an entry time to U1 from U0 of 0.1 µsec, and U1 to U0 exit time of 4 µsec. This results in the entire *ufp_bw* of 40 Gb/s allotted in either direction and the hub UFP fully utilized in each direction, following the procedure in Step 7.

Table 1: Bandwidth Allocation example for a Hub with 40 Gb/s bandwidth

		DFP 0	DFP 1	DFP 2	DFP 3
1	Link width, Freq (*dfp_bw[i])*	x2, 10 Gb/s (20 Gb/s)	x1, 5 Gb/s (4 Gb/s)	x2, 20 Gb/s (40 Gb/s)	x2, 20 Gb/s (40 Gb/s)
2	*dfp_bw_demand_ib*	18.0 Gb/s	2.0 Gb/s	36.0 Gb/s	28.0 Gb/s
3	*dfp_bw_demand_ob*	2.0 Gb/s	0.4 Gb/s	36.0 Gb/s	36.0 Gb/s
4	*frac_bw_allot*	0.192308	0.038462	0.384615	0.384615
5	1st pass b/w: inbound (i/b)	0 Gb/s	0 Gb/s	0 Gb/s	0 Gb/s
6	1st pass b/w: outbound (o/b)	2.0 Gb/s	0.4 Gb/s	0 Gb/s	0 Gb/s
7	2nd pass b/w: inbound	7.69 Gb/s	1.54 Gb/s	15.38 Gb/s	15.38 Gb/s
8	2nd pass b/w: outbound	2.0 Gb/s	0.4 Gb/s	18.8 Gb/s	18.8 Gb/s
9	Link Util: inbound, outbound, (overall)	0.38, 0.1 (0.38)	0.38, 0.1 (0.38)	0.38, 0.47 (0.47)	0.38, 0.47 (0.47)
10	Power Savings with 125µs scheduling interval with U1 low power state	2.04 W	2.04 W	1.74 W	1.74 W

The largest scheduling interval in Step 1i above is chosen to be less than the maximum latency tolerance of any DFP. For example, if a hub is connected to an isochronous audio device, which needs certain amount of outbound data (the audio bandwidth) every 250 µsec due to the amount of buffer it has and three bulk devices such as storage, then the scheduling algorithm must choose the scheduling interval <= 250 µsec and ensure that the audio device gets the desired level of outbound bandwidth.

The proposed scheduling algorithm schedules the data transfers in the devices in a way that they would get the bandwidth they would have received, subject to the worst case latency constraints, while maximizing the power savings by taking the devices to low-power state when they are not scheduled to transfer data.

The proposed approach requires the following enhancements to the power management states to be effective, even in the presence of USB devices connected to the DFP that are sub-optimal:

- The hub would force each hub DFP to enter the power savings Ux state after its scheduled U0 interval is over and wake up the port just in time for the scheduled U0 interval from its Ux state. Even if a USB device wants to wake up the Link, the hub DFP will not wake up the link
- An architected mechanism to force a device connected to a hub DFP to stay in the Ux state until the hub initiates an exit from electrical idle state
- An architected mechanism to measure the actual entry and exit latency in each of the ports to the Ux state that the scheduling algorithm reads. Note that this is done to get actual numbers based on the existing operating conditions and not the conservative numbers a device vendor may provide.
- A shallow U1 state implementation with fast entry and exit times (not to exceed 0.1 µs and 3.0 µs respectively). This amount of time is enough to keep the circuits in low-power state (presumably with PLL on), while retaining the state information and keeping the logic clock-gated so that we save 80+% power but maintain fast entry/exit times. This will ensure that the links can be put into the low power states even for devices that have low tolerance for latency (i.e. short scheduling time intervals) and still realize the power savings.

- An architected mechanism (through configuration registers) for devices to report their Ux entry and exit times along with their maximum latency tolerance. These will be used for the proposed algorithm to arrive at the appropriate scheduling interval and the appropriate Ux states for power savings
- An architected mechanism (configuration register) for each DFP to report the actual bandwidth usage in each direction, over one or multiple scheduling interval. The proposed algorithm can monitor this and modulate the bandwidth demand in either direction (*dfp_bw_demand_ib[i], dfp_bw_demand_ob[i])* up or down, if needed.

Results

The C code for our proposed approach can be accessed on GitHub (https://github.com/Bikds/HubPowerSavings). For the ease of representation of power savings in a graphical format, we made each of the DFPs identical in terms of the theoretical bandwidth as well as the projected bandwidth demand in either direction.

Except Figure 8, the times for entry to Ux state from U0 are: U1: 0.1 µs, U2: 10 µs, U3: 3 ms and the times for exit from Ux state to U0 are: U1: 4 µs, U2: 700µs, U3: 30ms. The power assumptions are: U0 power for a hub/device is 4.5 W for USB 3.0/3.1 devices and 7.5 W for USB 3.2 and USB4 hub/devices; U1 power is 1W for USB 3.0/3.1 and 1.67W for USB 3.2 and USB4; U2 power is 0.1 W for USB 3.0/3.1 and 0.167W for USB 3.2 and USB4; U3 power is 0.0125W for USB 3.0/3.1 and 0.02W for USB 3.2 and USB4. Once again, to avoid repetition, we are referring to the power states using USB3.x terminology even for USB4 devices in this section.

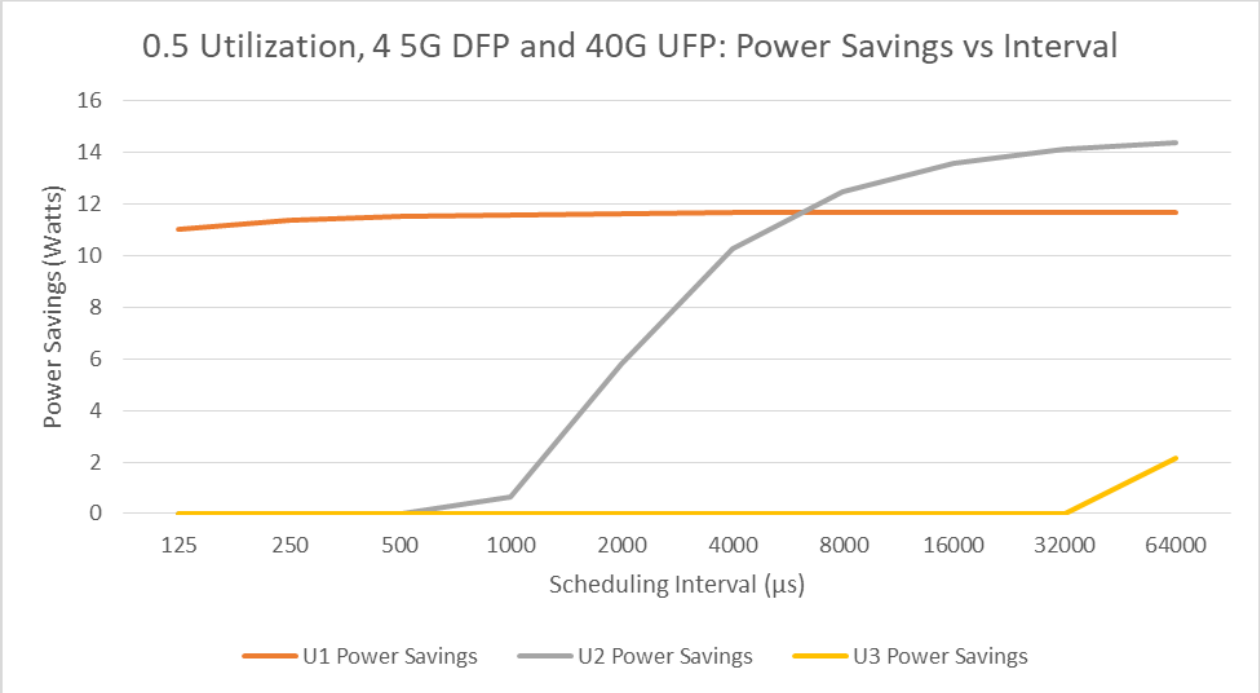


Figure 3: Power savings vs scheduling intervals for a Hub with a UFP to net DFP bandwidth ratio of 2.5:1 (40Gb/s Upstream bandwidth connected to 4 downstream devices each 5Gb/s)

Figure 3 shows the power savings versus scheduling interval time for the three power savings states. As expected, the U2 (and U3) power savings is 0 when the scheduling interval is less than 500 µs. The UFP bandwidth is more than the bandwidth demand of the 4 DFPs combined (8Gb/s at 50% utilization). When the scheduling interval is 125 (or 250 or 500) µs, the time each DFP needs to be in U0 is 62.5 (or 125 or 250) µs. With an entry+exit time of 710 µs for U2, the U0 time plus the entry/exit time is greater than the scheduling interval both for each DFP as well as the UFP. Hence the power savings for U2 is 0, as the port cannot enter U2 state. However, with an entry+exit time of 4.1 µs, power savings in U1, though less than U2 or U3, is realizable. Thus, the device stays in U1 longer than it can stay in U2, even if U2 is feasible. As the scheduling interval increases beyond 1000 µs, the power savings in U2 keeps increasing. The rate of increase due to U2 is higher initially due to the extra time we add to the scheduling interval directly goes to the sojourn in U2 state. The power savings of U2 crosses over U1 around 8000 µs, where the power consumption during the high entry and exit times of U2 is offset by the deeper power savings it offers. The power savings in U1, due to the increase in scheduling interval, increases modestly and quickly flattens out since the entry/exit latency is a small fraction of the scheduling interval. We make the same observations in Figure 4, even though the devices connected to the DFP are higher bandwidth (20Gb/s vs 5 Gb/s) and their aggregate bandwidth demand is 80 Gb/s (ten-fold of Fig. 3) which is twice the bandwidth that can be sustained by the UFP. There is no U3 power savings in Figure 4 because each DFP needs to stay in U0 for 50% of time to deliver its allocated bandwidth. At 64000 µs, that is 32000 µs of U0 and with entry+exit time to U3 at 33 ms (33000 µs), the DFP cannot enter U3. However, as Figure 5 demonstrates, as the scheduling interval increases, the U3 power savings exceeds U2's around 4 sec.

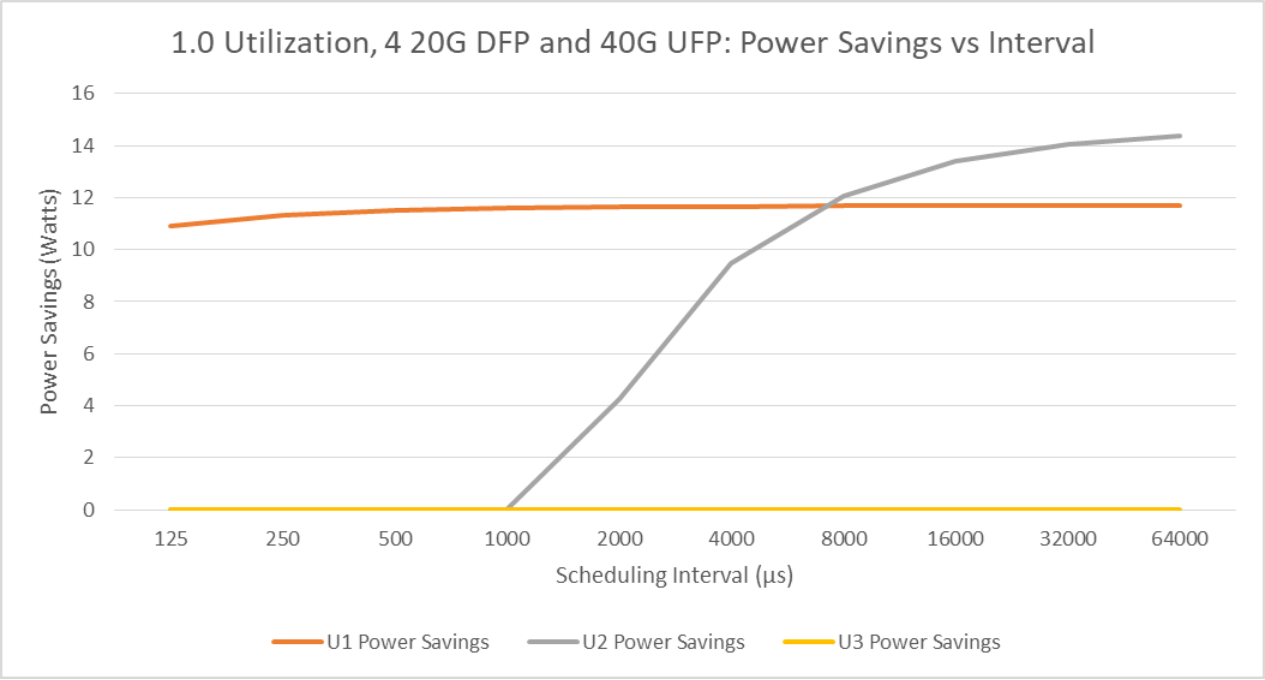


Figure 4 Power savings vs scheduling intervals for a Hub with a UFP to net DFP bandwidth ratio of 2.5:1 (40Gb/s Upstream bandwidth connected to 4 downstream devices each 5Gb/s)

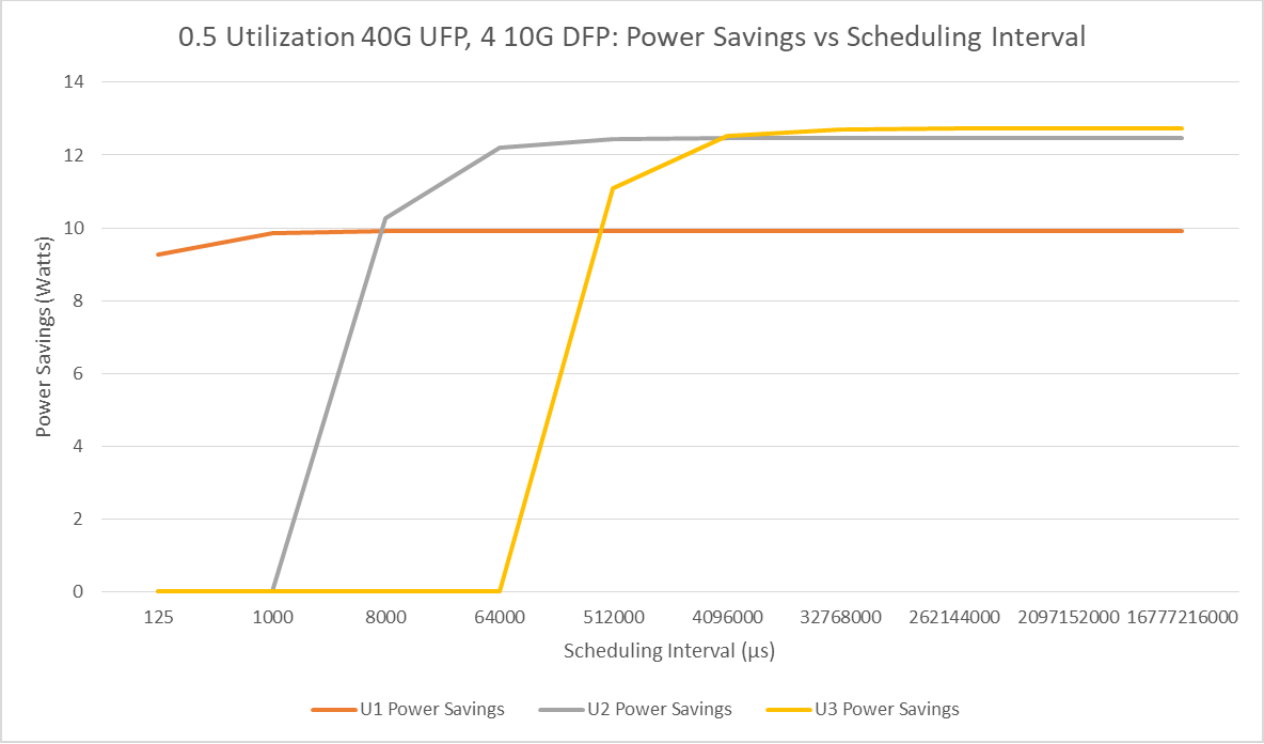


Figure 5: Power savings vs scheduling intervals for a Hub with a UFP to net DFP bandwidth ratio of 1:1 (40Gb/s Upstream bandwidth connected to 4 downstream devices each 10Gb/s)

Figure 6 and *Figure 7* demonstrate how power savings changes with the bandwidth demand (utilization) of each device connected to a DFP. As expected, when utilization increases, each port needs to be in U0 longer and hence the power savings decreases. Since the raw bandwidth of the UFP matches that of the 4 DFPs together, the U1 power savings linearly decreases to 0 at a utilization close to 1. However, we hit 0 power savings in U2 at a utilization of around 0.3. Since the scheduling interval is 1000 µs, that means each DFP needs to be in U0 for 300 µs for a utilization of 0.3. With an entry+exit time of 710 µs for U2 state, that results in the link not entering U2 state. With a long scheduling interval of 64000 µs, power savings is realized even in U3 state in *Figure 7*. As expected, the power savings reduces with utilization. However, since the UFP bandwidth matches the bandwidth of each of the DFP, once each DFP exceeds the utilization of 0.25, the UFP bandwidth becomes the bottleneck and each DFP can deliver no more than 25% of its peak bandwidth, thereby causing a flat-line in the power savings.

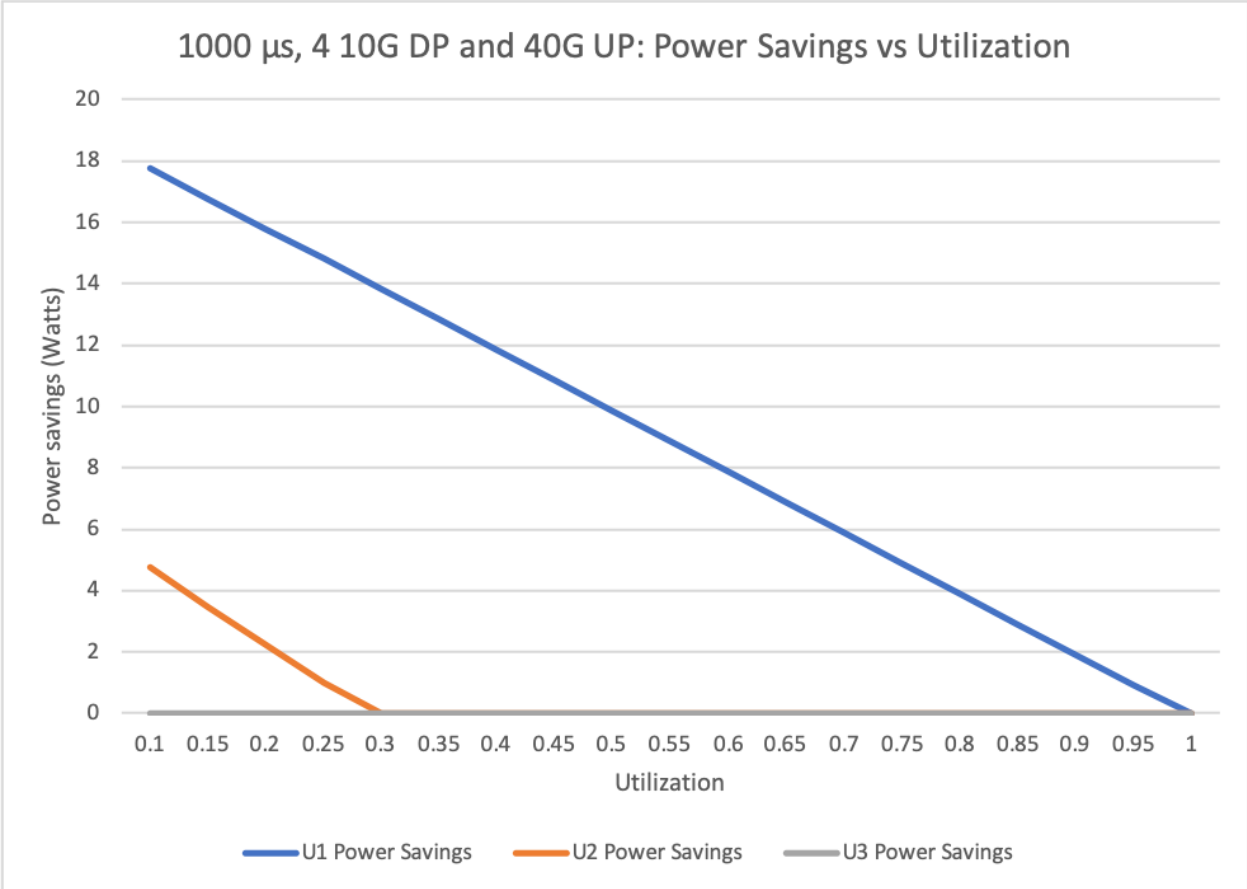


Figure 6: Power savings vs bandwidth demand by each device (utilization) for a Hub with a UFP to net DFP bandwidth ratio of 1:1 (40Gb/s Upstream bandwidth connected to 4 downstream devices each 10Gb/s)

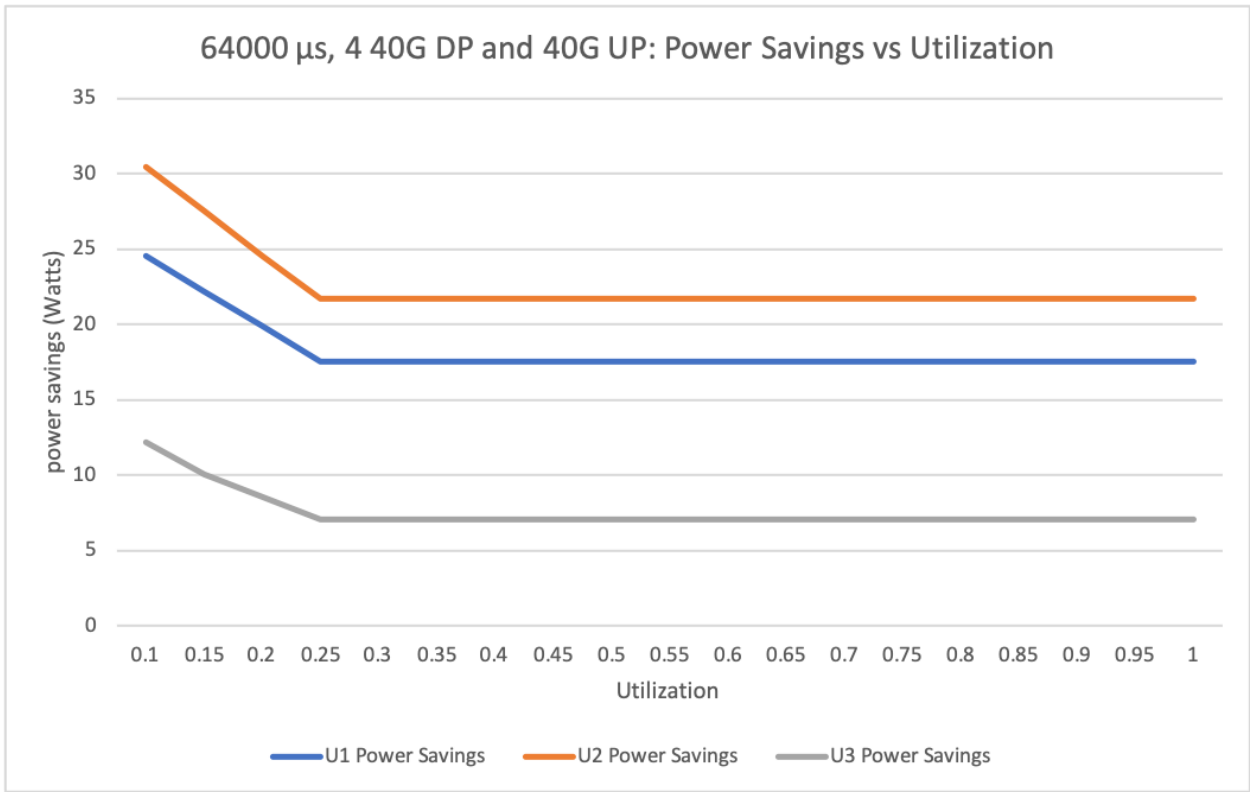


Figure 7: Power savings vs bandwidth demand by each device (utilization) for a Hub with a UFP to net DFP bandwidth ratio of 1:4 (40Gb/s Upstream bandwidth connected to 4 downstream devices each 40Gb/s)

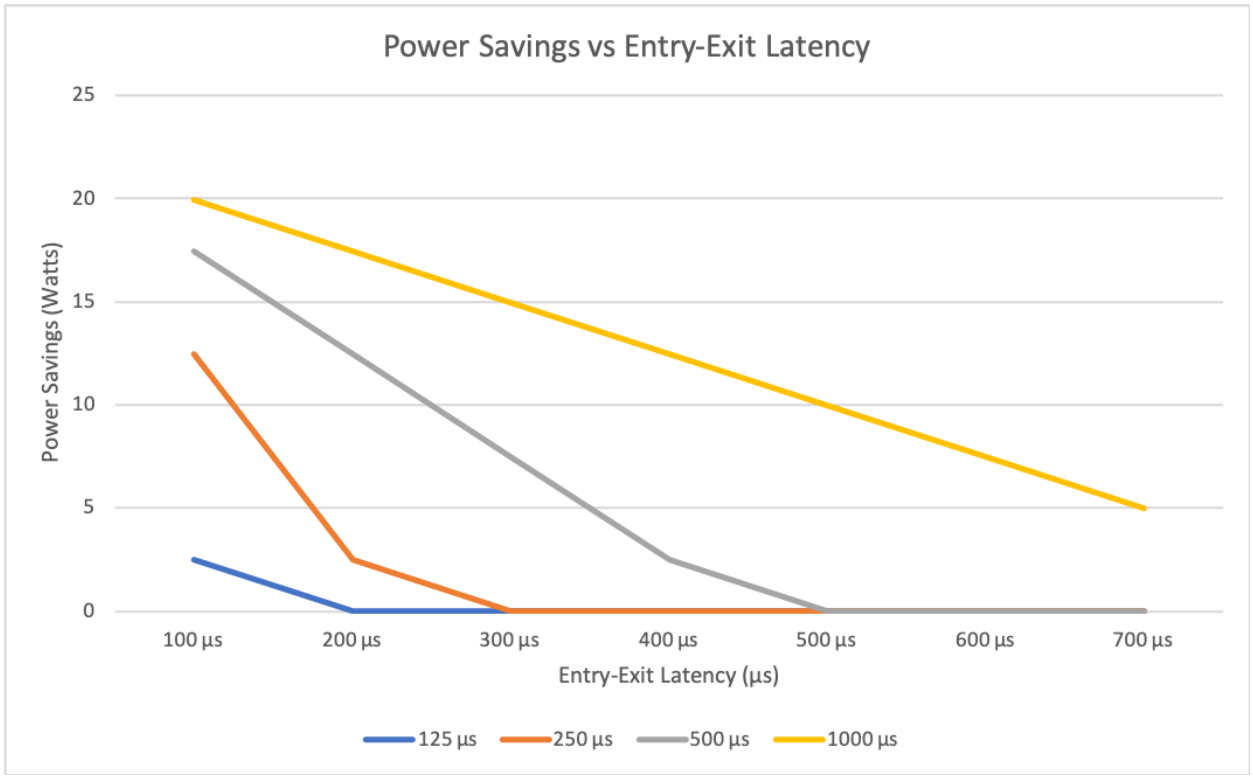


Figure 8: Show the power savings vs U2 entry-exit latency

Figure 8 shows how the power savings decreases with the entry+exit latency increase. As one may expect, the higher the entry+exit latency, the lower will be the sojourn in Ux state for a given scheduling interval and DFP/UFP utilization.

Conclusions and Future Work

We have demonstrated significant power savings in a platform with the proactive power management policy of our proposed approach for hub power savings. We have proposed mechanisms to reduce the entry and exit times to U1 (or CL1) state, resulting in significant power savings even though U1 (or CL1) is a shallow power savings state. Our results also suggest that we save more power by increasing the scheduling interval, assuming the downstream devices can tolerate the latency increase due to the forced lower power state. As USB moves to USB4 and hubs will be used to connect to higher bandwidth devices, these power savings will become even more pronounced.

As future work, we want to extend the power savings mechanisms to a mix of scheduling intervals, depending on the device types. For example, we can create multiple overlapping scheduling intervals for different classes of devices, depending on their latency tolerance requirements. This will be further enhanced to deal with a hierarchy of hubs. We can also address the case where due to design limitations, a device not have enough storage to be able to stream the full bandwidth beyond a certain interval. We also plan to extend the power savings mechanisms to USB2 devices connected below a hub. This can be done by extending a similar power savings scheduling mechanism in the USB2 part of the hub.

References

1. "Universal Serial Bus 3.2 Specification", USB 3.0 Promoter Group, September 2017
2. Abdul Rahman Ismail, "Introduction to USB", USB presentation to Portland State University
3. "USB", Wikipedia, https://en.wikipedia.org/wiki/USB
4. Ganesh TS, "USB Flash Drives - Power Consumption Measurement using Plugables's USBC-TKey", AnandTech, https://www.anandtech.com/show/10163/usb-flash-drives-power-consumption-measurement-using-plugables-usbctkey
5. "Power Consumption", Test USB, http://www.testusb.com/power_issue.htm
6. Mike Michelleti, "USB 3.0: Delivering superspeed with 25% less power", EDN Network, Nov 2010, http://www.testusb.com/power_issue.htm
7. Lecroy, "USB 3.0: Delivering superspeed with 25% less power", Draft 10, App Note, https://www.mindshare.com/files/resources/PLX_USB_PowerManagement_AppNote.pdf
8. Brian Ellis, "Is your USB design suffering from bandwidth overkill", Cypress, https://www.cypress.com/file/88501/download
9. "Understanding USB bandwidth and KVM applications" icron, http://www.icron.com/_resources/USB_BandwidthandApplications.pdf
10. "USB 3.0 Link Power Management States" Cypress Developer Community 3.0, https://community.cypress.com/docs/DOC-14420

11. Rita Horner, “Using PCI Express L1 Sub-States To Minimize Power Consumption In Advanced Process Nodes”, Semiconductor Engineering, July 2014, https://semiengineering.com/using-pci-express-l1-sub-states-to-minimize-power-consumption-in-advanced-process-nodes/