July 2020

A MECHANISM TO DYNAMIC ADJUSTS USB TYPE C PORTS' BANDWIDTH

HP INC

A mechanism to dynamic adjusts USB type C ports' bandwidth

USB 3.2 spec provides 2 sides (flip-flop side) data transfer path called 3.2x2 (10Gx2) and USB 4.0 (20Gx2) following. However, it is challenged to implement the solution into a real system because of 2 side signal request more circuits and PCB/A space. Therefore, it will increase the system design complexity and cost either.

Considering

1. USB ports do not fully occupy always
2. The real USB3.2x2 device might behind host supported 3-5 years
3. Device not always asking full bandwidth for data transformation

Whether we could make the ports as dynamic bandwidth arrange, the answer is correct.

To deliver the idea, it necessary to modify the architecture like Fig. 1, it is based on standard USB 3.2x2 topology and add a "reset engine" and a "cross-point switch" 2 circuits.

Cross-point switch

A cross-point switch is a 2x4 signal switch, it could directive 2 USB date signal to 4 output paths decided by CC signals. EX: 10Gx2 USB3.2x2 bandwidth could fully source to one USB-C port or separate two USB-C ports. Besides, since USB-C supported the flip-flop feature, the cross-points switch could make proper connections whenever the top or bottom side has inserted once side device.

Reset engine:

A reset engine included USB CC logic and could communicate with USB host and UCSI driver via SMBus. It monitors the USB devices' status of ACPI and will remove or connect devices if request. Once the devices connected and doing enumeration, the USB will start handshake and then make proper bandwidth allocation. Further, if port A connected with the device, plug device into port B could trigger reset event once port A device entry U3 status. Then, the UCSI driver will active removes the device to make USB ports re-enumeration and do proper bandwidth arrangement accordingly.

How it's working, you could refer Tab. 1 and Tab. 2 and follow the step below to specify the operation. Of course, the behaves beyond the below 4 cases.

Case A:

While port A/B were empty, each port was reserved USBx1 signal at the top side as default. Once plug USB-C 3.2x2 device at port A, the USBx2 signal will all feed to Port A device.

Case B:

While port A was connected with one USB3.1x1 and topside, the port was source USBx1 signal at the top side. Plug another one USB-C 3.2x1 device at port B but bottom side, the USB Port A's signal swap to bottom side to source Port B device behind Port B device entry U3.

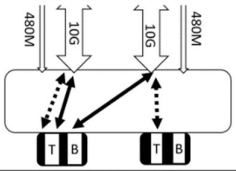
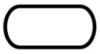
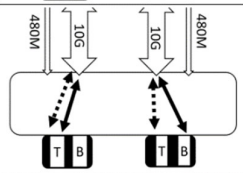
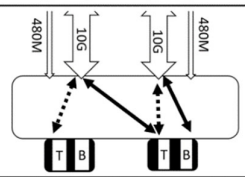
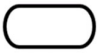
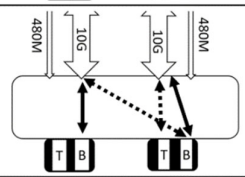
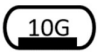
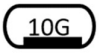
Case C:

While port A/B were connected individual USB 3.2x2 device, each port was only allowed to source USBx1 at the top side like default. Once remove USB-C 3.2x2 device from port A, the USBx2 signal will all feed to Port B device once Port B device entry U3 status.

Case D:

While port B connected with one USB 3.2x2 device in beginning, the port was sourced all USBx2 signals. Once plug anther USB-C 3.2x2 device at port A, the cross-point switch switches one USB signal to Port A and bottom side after device B entry U3.

Tab. 1

Case	change	cross points path acting	USB C_A	USB C_B
A	Before			
	After			
B	Before			
	After			
C	Before			
	After			
D	Before			
	After			

Tab. 2

Symbol	Note
	USB date path Before
	USB date path After
	3.2x2 device loaded (Both)
	3.2x1 device loaded (Top)
	3.2x1 device loaded (Bot)
	No device loaded

Reference

1. ACPI specification v6.3
2. The USB 3.2 specification
3. USB Type-C Connector System Software Interface (UCSI) driver

Figure list

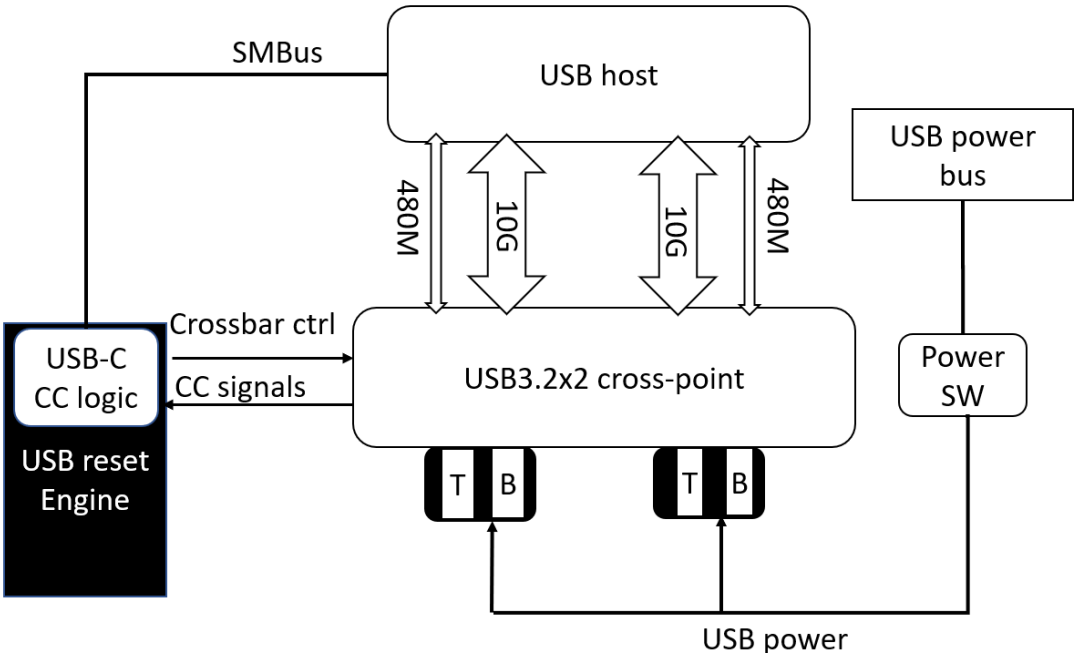


Fig. 1 the architecture to dynamic adjust USB3.2x2 signal for 2 ports

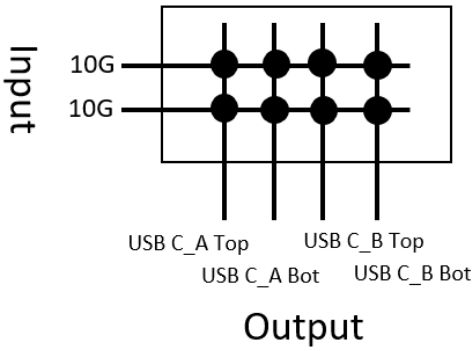


Fig. 2a cross-points

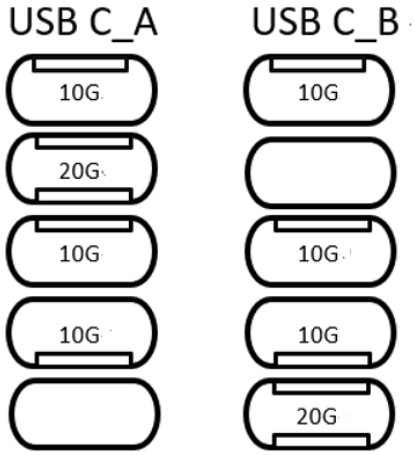


Fig 2b USB 3.2x2 signal share for 2 ports

Disclosed by Chao-Wen Cheng, Jim Chang, Eric Su and Poying Chih, HP Inc.