System on a Package Innovations With Universal Chiplet Interconnect Express (UCIe) Interconnect

Debendra Das Sharma, *Data Center and AI Group, Intel Corporation, Santa Clara, CA, 95054, USA*

Universal Chiplet Interconnect Express (UCIe) is an open industry standard interconnect for developing an open chiplet ecosystem, where chiplets from any supplier can be packaged anywhere in an interoperable manner. This article delves into the architectural and protocol aspects that we developed and have been adopted in the UCIe 1.0 specification. We present our results on these aspects based on our implementation studies.

Gordon Moore predicted the "Day of Reckoning" in his seminal paper where he posited "Moore's law"[1]: "It may prove to be more economical to build large systems out of smaller functions, which are separately packaged and interconnected." Today, we are past that inflection point. On-package integration of multiple dies has been widely deployed across the semiconductor industry for many years, including mainstream volume central processing units (CPUs) and general purpose graphics processor units.[2,3,4]

There are many drivers for on-package chiplets.[3,4] As die sizes increase to meet the growing processing demand, they are exceeding the reticle limit and facing yield challenges in the advanced process nodes. Smaller chiplets connected on package using scale-up protocols help designers overcome these challenges.

Lowering the overall portfolio cost with a time to market advantage is a compelling driver for deploying chiplets.[3,4] For example, the compute cores shown in Figure 1 can be implemented in an advanced process node to deliver leadership power-efficient performance, whereas the fabric functionality comprehending memory and input/output (I/O) controller functions may be reused from a design already deployed in an established process node. Such partitioning results in smaller dies with improved yield. It also mitigates IP porting costs that are increasing significantly for the advanced process nodes.[9]

0272-1732 © 2023 IEEE
Digital Object Identifier 10.1109/MM.2023.3235770
Date of publication 10 January 2023; date of current version 13 March 2023.

Another value of chiplets is the ability to offer bespoke solutions.[3,4] For example, one can choose different numbers of compute, memory and I/O, and accelerator chiplets depending on the need of a particular product segment. As a result, one does not need to do a different die design for different segments, lowering the design, validation, and product costs.[3,4]

UNIVERSAL CHIPLET INTERCONNECT EXPRESS (UCIe) IS AN OPEN INDUSTRY STANDARD INTERCONNECT, OFFERING HIGH-BANDWIDTH, LOW-LATENCY, POWER-EFFICIENT, AND COST-EFFECTIVE ON-PACKAGE CONNECTIVITY BETWEEN HETEROGENEOUS CHIPLETS.

Universal Chiplet Interconnect Express (UCIe)[5] is an open industry standard interconnect, offering high-bandwidth, low-latency, power-efficient, and cost-effective on-package connectivity between heterogeneous chiplets. It addresses the compute, memory, storage, and connectivity needs across the entire compute continuum, spanning cloud, edge, enterprise, 5G, automotive, high-performance computing, and hand-held segments. UCIe 1.0 specification,[5] donated by Intel Corporation, comprehends all the layers of the stack [see Figure 3(a)]. This is the only specification we are aware of with plug-and-play capability and a compliance mechanism, targeting heterogeneous integration of

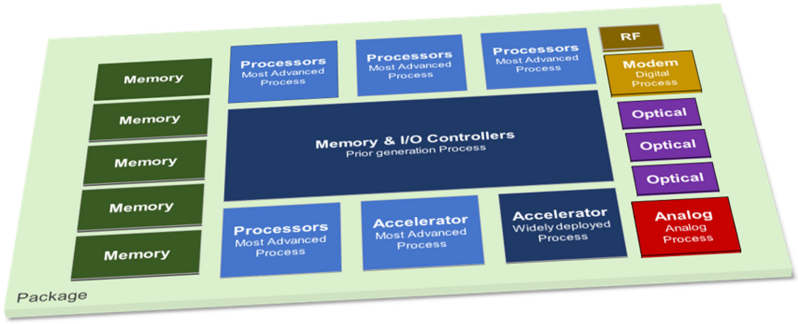


FIGURE 1. UCIe-based open chiplet ecosystem: Platform on a package.

components. The use of well-established PCI-Express (PCIe)[6,7] and Compute Express Link (CXL)[8] protocols and software infrastructure will ensure seamless interoperability. UCIe enables a designer to package chiplets from different designs and fabs, using a wide range of packaging technologies. UCIe is an evolution of our prior proprietary Multi-Die Fabric Interface, implemented in Intel Sapphire Rapids CPU.[2,3,4]

This article delves into the requirements and usage models for UCIe in the "Performance Metrics and Usage Models Supported By UCIe 1.0 Specification" section. Our proposed approach is described in the "Our Proposed Approach for UCIe" section, which has been mostly adopted in the UCIe Specification.[5] We present our results in the "Results" section. Finally, the "Conclusion" section concludes this article.

PERFORMANCE METRICS AND USAGE MODELS SUPPORTED BY UCIe 1.0 SPECIFICATION

UCIe offers compelling key performance indicators (KPIs), as summarized in Figure 2(a).[3,4] To support a wide range of usage models, UCIe deploys two types of packaging, as shown in Figure 2(b),[3,4] each with multiple commercially available options.[10,16,17,18] The standard package (2-D), referred to as UCIe-S, is used for cost-effective performance. The advanced packaging (2.5-D), referred to as UCIe-A, is used for power-efficient performance. Three-dimensional integation of chiplets is outside the scope of this article and UCIe 1.0 specification. UCIe consortium is expected to be address 3-D integration in a future revision of UCIe specification.

A chiplet (or die), designed for standard package or advanced package, is expected to interoperate with any other chiplet designed on the same package type, across the wide range of bump pitch in each type described in Figure 2(a). The KPI table [see Figure 2(a)] conservatively estimates power and performance for the most widely deployed bump pitch today. For example, the bandwidth density will go up by up to 3.24× with 25 μm from 45 μm. Even at 45 μm, the bandwidth density of 1,300+ GB/s/mm or mm^2 (linear, area) is three orders of magnitude better than PCIe/CXL. Similarly, PCIe physical (PHY) power efficiency of 5–10 pJ/b can be lowered by up to 20× with the UCIe due to the shorter channel reach.

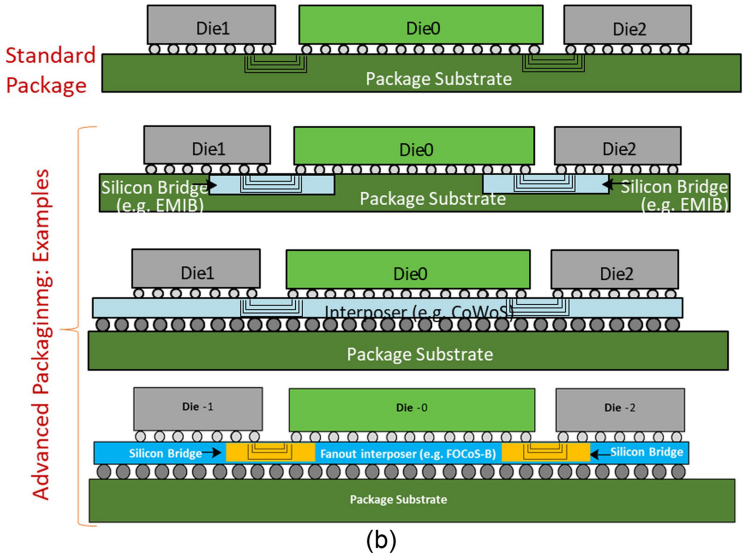
OUR PROPOSED APPROACH FOR UCIe

Our approach is a well specified and layered standard, including protocol layer, die-to-die (D2D) adapter, and PHY layer. A standard configuration register space is defined following the PCIe[6,7] for device discovery, debug, as well as error reporting, logging, and notification services (e.g., interrupts). Packaging and electrical aspects are beyond the scope of this article and is covered by Sharma et al.[4]

The PHY layer is responsible for the electrical signaling, clocking, link training, sideband, circuit architecture, etc. We support different data rates, widths, bump pitches, and channel reach to ensure the widest interoperability feasible, as detailed in Figure 2(a). The unit of construction of the interconnect is a module (also known as cluster) that consists of N single-ended, unidirectional, full-duplex data lanes ($N = 16$ for UCIe-S and $N = 64$ for UCIe-A), one single-ended lane for valid, one lane for tracking, a differential forwarded clock per direction for the main band. The sideband consists of two single-ended lanes (one data and one 800-MHz forwarded clock) per direction. The sideband interface is used for status exchange to facilitate link training, register access, and diagnostics. Multiple modules can be aggregated to deliver more performance per link, as shown in Figure 3(b). At 32 GT/s, a single-module UCIe-S and x16 PCIe 5.0 at 32 GT/s have the same raw bandwidth. A single-

Characteristics / KPIs	Standard Package	Advanced Package	Comments
Characteristics			
Data Rate (GT/s)	4, 8, 12, 16, 24, 32		Lower speeds must be supported -interop (e.g., 4, 8, 12 for 12G device)
Width (each cluster)	16	64	Width degradation in Standard, spare lanes in Advanced
Bump Pitch (um)	100 – 130	25 - 55	Interoperate across bump pitches in each package type across nodes
Channel Reach (mm)	<= 25	<=2	
Target for Key Metrics			
B/W Shoreline (GB/s/mm)	28 – 224	165 – 1317	Conservatively estimated: AP: 45u for AP; Standard: 110u; Proportionate to data rate (4G – 32G)
B/W Density (GB/s/mm^2)	22-125	188-1350	
Power Efficiency target (pJ/b)	0.5	0.25	
Low-power entry/exit	0.5ns <=16G, 0.5-1ns >=24G		Power savings estimated at >= 85%
Latency (Tx + Rx)	< 2ns		Includes D2D Adapter and PHY (FDI to bump and back)
Bit Error Rate (BER)	< 10^{-27} below 16G for UCIe-a; 12G for UCIe-S; else < 10^{-15}		Probability of a bit error
Reliability (FIT)	0 < FIT (Failure In Time) << 1		FIT: #failures in a billion hours (expecting ~1E-10) w/ UCIe Flit Mode

(a)



(b)

FIGURE 2. UCIe: Target performance metrics and supported channels. (a) Key metrics targeted by UCIe 1.0 specification. (b) 2-D and 2.5-D packaging options UCIe 1.0.

module UCIe-A at 32 GT/s has the same raw bandwidth as a x16 PCIe 7.0 at 128 GT/s.

The advanced package has extra wires for repair due to higher susceptibility to faults due to high density. There are two spare wires every 32 data wires, one spare wire for the clock and track, one spare wire for valid, and two spare wires for sideband. The standard package relies on link width degradation to work around failed lanes, consistent with external interconnects with the same bump pitch. The PHY specification provides the bump map for standard and advanced packages[4,5] to ensure seamless interoperability between chiplets.

The D2D adapter provides the link state management and parameter negotiation for the chiplets. It guarantees reliable delivery of data through cyclic redundancy check (CRC) and link level retry mechanism, when enabled.

We map the PCIe and CXL protocols to UCIe natively as those are widely deployed at the board level across all segments of compute. With CXL.io and PCIe, data transfer using direct memory access, software discovery, error handling, virtualization, etc., are addressed. The memory use case is handled through CXL.Mem. Caching requirements of accelerators, smart networking devices, etc., are addressed with CXL.cache. One can easily migrate PCIe/CXL board components on-package, as appropriate. In addition, system-on-chip construction, link management, and security solutions deployed in platforms can be seamlessly deployed on UCIe. We also define a "streaming protocol," which can be used to map any other protocol, such as a proprietary symmetric cache coherency protocol (e.g., ultra path interconnect) on UCIe at the flit-aware die-to-die (D2D) interface (FDI) or raw D2D interface (RDI) level. Our

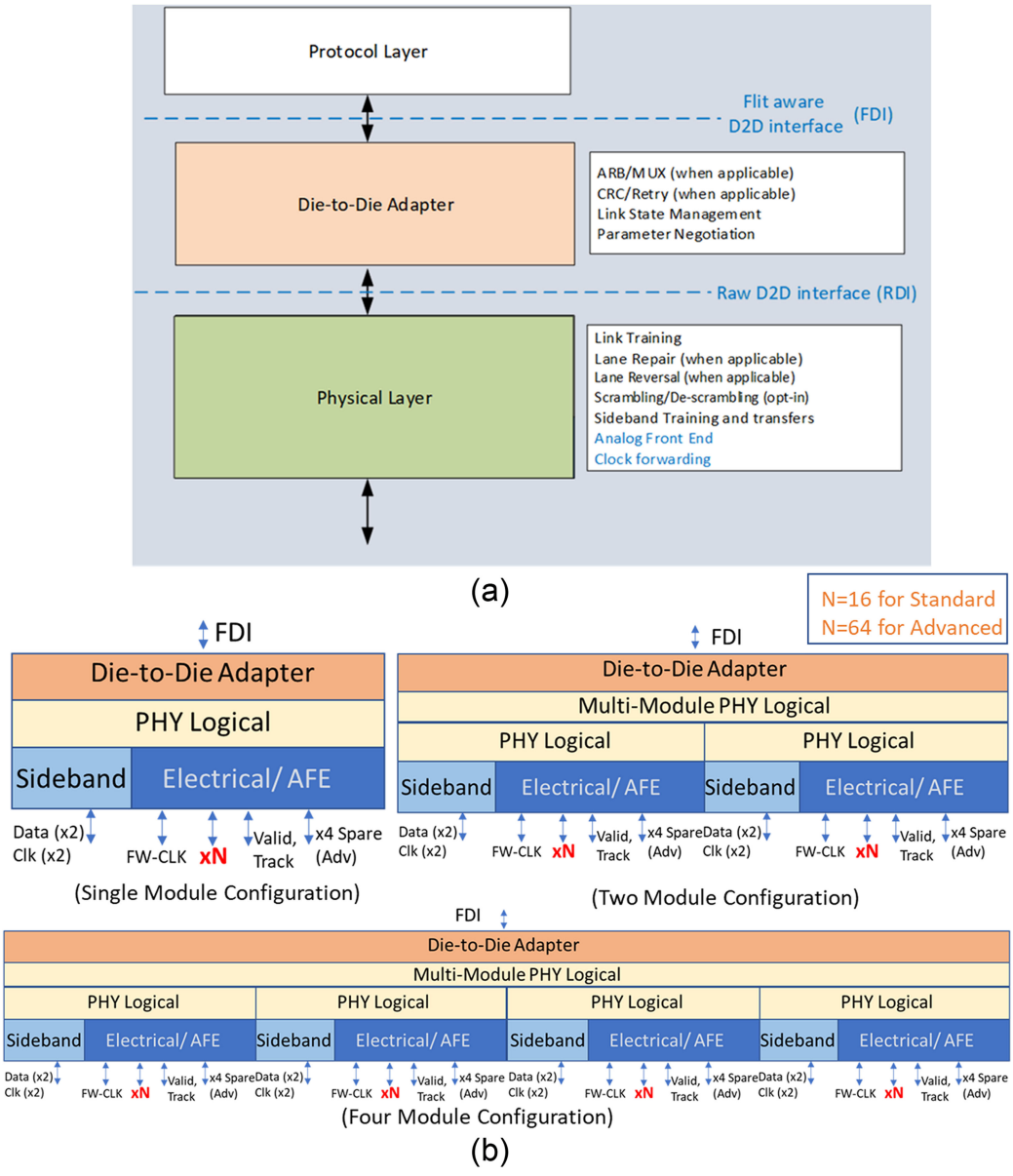


FIGURE 3. Layering with UCIe: Each stack can comprise 1, 2, or 4 PHY building blocks, called modules. (a) Layering with UCIe. (b) Multimodule configuration with UCIe.

approach also enables the UCIe consortium to innovate new protocols to cover new usage models or enhance existing ones going forward.

PCIe and CXL protocols can also use the "raw mode" where the underlying PCIe and CXL forward-error-correction (FEC), CRC, and replay mechanisms[6,7,8] are used. These are useful for applications, such as copackaged optics, where a UCIe retimer chiplet will pass bits mostly unchanged (except for any periodic media management) between two packages in different chassis or rack or pod,[3,11,13] using an alternate media, such as optics. Thus, using UCIe retimers, a CPU package can be connected to a CXL switch package, transporting CXL protocol over an optical connection. This enables us to realize the vision of composable systems at the rack/pod level for pooling and sharing memory and processing resources (CPUs, accelerators, etc.),[8,11,12,13] with high bandwidth and low-latency optical interconnects [as shown in Figure 2(a)], delivering better power-efficient performance with lower total cost of ownership than what is possible today.

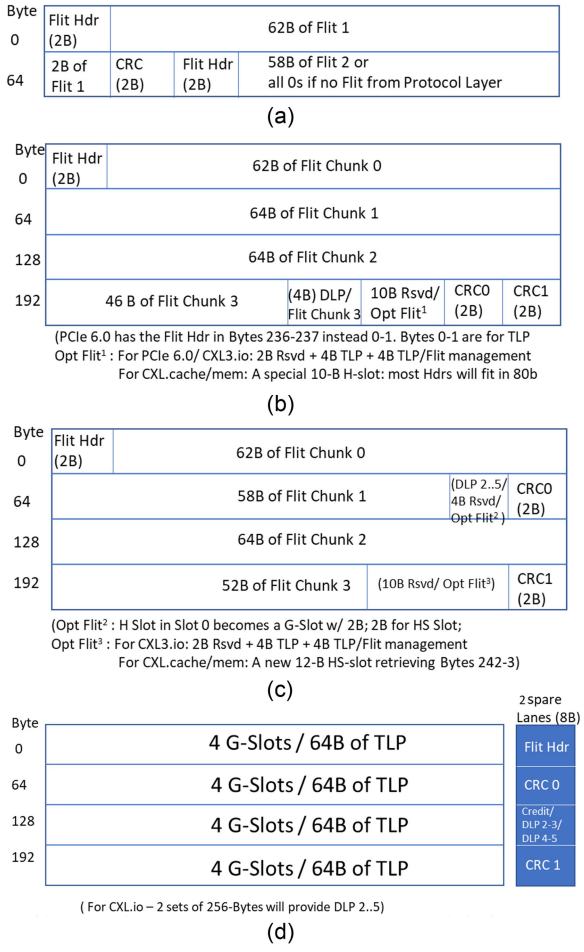


FIGURE 4. Flit layout for various options. (a) 68-byte flit—usage CXL 2.0/PCIe non-flit mode. (b) 256-byte flit—usage CXL 3.0/ PCIe 6.0. (c) 256-byte latency-optimized (LO) flit—usage CXL 3.0. (d) Optional optimized 264-byte LO flit for CXL3 (256 bytes are used for data, 8 bytes from the two spare lanes carry CRC, etc.).

Error Handling Mechanism

We define two sets of bit error rate (BER), depending on data rate: 10^{-27} and 10^{-15} [see Figure 2(a)]. To achieve a target failure in time (FIT, the number of failures in a billion hours) of less than 10^{-3}, we deploy CRC and replay for the higher BER. Parity is used for lower BER, where detected errors are fatal.

We use flow-control unit (flit), protected by CRC or parity, as the basic unit of transfer. The proposed 16-bit CRC, with a generator polynomial of $(x+1)*(x^{15}+x+1)$, ensures up to triple bit flip detection over $<=$ 132-byte transfer, beyond which the aliasing probability is 2^{-16}. For 68-byte flits (e.g., CXL2 and PCIe in non-flit mode), we substitute the protocol CRC with UCIe one. For 256-byte flits (and 264-byte enhanced flits) (such as PCIe 6 and CXL3), we apply two sets of CRC (CRC0 and CRC1) on each 128 byte (or 132 byte) half independently. Given that the circuits are fairly simple without any decision feedback equalizer, no correlated errors are expected.

The valid signal is independently protected against upto triple bit flips (with replay) due to its valid 8-bit encodings having a hamming distance >3.

Flit Layout for UCIe-Supported Protocols

UCIe supports multiple flit layouts (see Figure 4), to support 68-, 256-, and 264-byte flits. A 264-byte flit is defined for PCIe 6.0 and CXL 3.0 with protocol enhancements. Any proprietary protocol can map to one of these flit formats.

Our proposed flit Hdr is a 2-byte field, comprising 3 bits of protocol ID, 1-bit credit/next flit type encoding, 8-bit sequence number, and 2 bits of Ack/Nak/ Sequence number, and 2 reserved bits.[5] In the 68-byte flit arrangement, the 64 bytes of flit payload are identical to CXL 2.0, and the CRC field is the UCIe CRC.

The 256-byte flit [see Figure 4(b)] is similar to the standard 256-byte PCIe 6.0/ CXL 3.0 flit.[7,12,14,15] The 14 bytes of FEC and CRC are replaced with 10 reserved bytes followed by two sets of UCIe CRC; CRC0 (protecting bytes 0–127 and 252–253) and CRC1 (protecting bytes 128–251 and 254–255). The 6 bytes of DLP for PCIe 6.0/ CXL3.io[7,12] are in the flit Hdr and bytes 238–241. We propose an optional bandwidth optimization strategy with an additional 10-byte header slot for CXL3.cache/mem,[12] as shown in Figure 5(a), and an additional 8-byte TLP payload for CXL.io/PCIe instead of the reserved bytes. An alternative would be both slot 0 and slot 15 to be 12B each (HS slots) is shown in Figure 5(a).

Figure 4(c) depicts the 256-byte LO flit, which follows a similar layout as the LO 256-byte CXL3 flit,[12] with the 18 bytes of FEC and CRC substituted by reserved bytes, UCIe CRC or other payload. CRC0 is placed in bytes 126–127, protecting the first 128 bytes. The 6 bytes of DLP for PCIe 6.0/ CXL3.io[7,12] are in the flit Hdr and bytes 122–125. CRC1 is placed in bytes 254–255, protecting bytes 128–255. Figure 5(b) shows the layout of CXL.cache/mem slots without bandwidth optimization. The optional bandwidth optimization layout is shown in Figure 5(c). Slot 0 is converted to a G-slot, by adding bytes 124–125. Slot 7 becomes the HS slot, and slot 8 becomes a G-slot based on the availability of bytes. We create a new HS-slot in slot 15. Thus, we increased the number of G-slots from 13 to 14.

We define a new 264-byte flit for advanced package by taking advantage of the spare lanes, if available after repair, as shown in Figure 4(d). The spare lanes

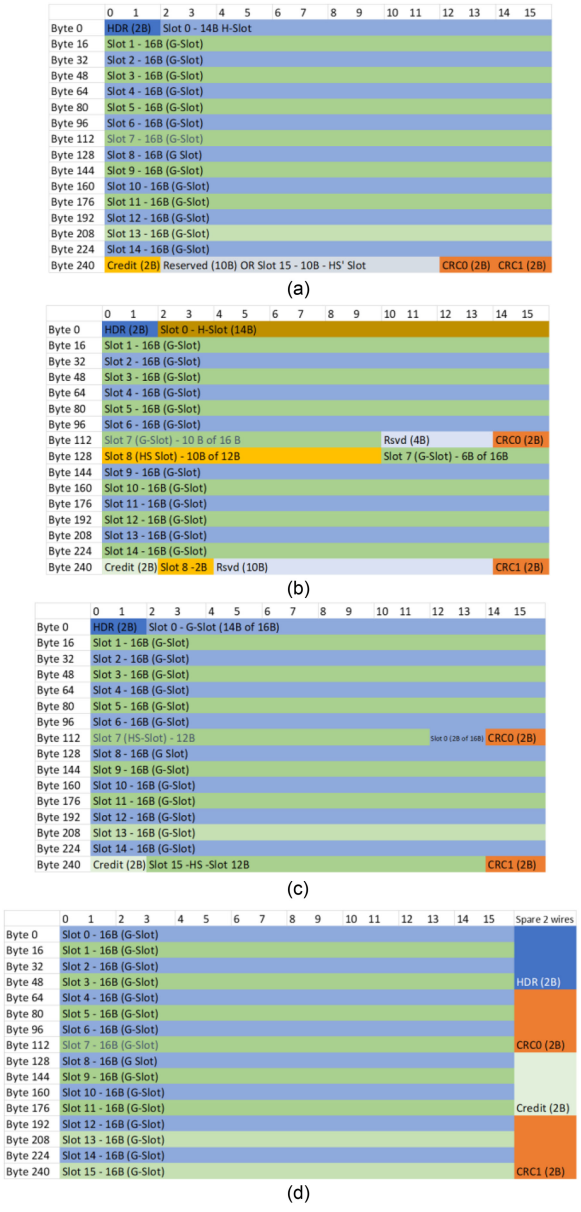


FIGURE 5. Proposed UCIe-optimized flit layout for CXL. cache/mem. (a) UCIe 256-byte flit for CXL3: unoptimized 15 slots or optimized 16 slots. (b) UCIe 256-byte LO flit for CXL3. (c) UCIe 256-byte LO flit with extra slots. UCIe 264-byte LO flit with 16 G-slots.

carry the 2 bytes of Hdr and 2 bytes of credit/DLP. CRC0 covers bytes 0–127 plus Hdr 2 bytes. CRC1 covers bytes 128–255 plus the credit/ DLP. For CXL3.io or PCIe 6.0 flits, the second half of the flit is the same length as the first half with 128-byte TLP-payload. For CXL.cache/mem, this results in 16 G-slots, as shown in Figure 5(d).

When the raw mode for transfer is used, it occurs in 64-byte units. The protocol layer directly sends bits through the RDI interface, bypassing the D2D adapter.

Low-Power States in UCIe

UCIe supports deep low-power states (L1 and L2), similar to PCIe,[6,7] following an explicit software-initiated handshake between the two components. During L1, everything in the main data path is clock gated, all main-band wires are tristated, and clock-generating circuits, such as phased-locked loop (PLL) are turned off. The state is maintained since the voltage is on, only leakage power is consumed. During L2, the main data path can be power gated, resulting in even deeper power savings. However, the state is lost. The sideband is an independent voltage domain and is used in both cases to bring the main-band link back up again.

We propose a new hardware-controlled, asymmetric, and dynamic low-power state for UCIe. When there is no information (flit) to be sent, the transmitter does not toggle the forwarded clock (FWDCLK) wires and keeps the valid low. Since the forwarded clock is used to capture data on the receive side, the receiving die can clock gate the receive logic while the transmitting die clock gates the transmit path. Thus, the two directions can be clock gated independently. The clock generation logic must be active so that the entry and exit times are low [$<=1$ ns, Figure 2(a)]. The FWDCLK is active for 16 unit interval (UI) with valid low and data tristated, after FWDCLK is turned off (parked high/low). We can turn ON the clock and data after that in <8 UI time. The expected power savings are 85% with this mechanism, based on our sapphire rapids (SPR) experience. Given the low entry and exit latency ($\sim$1 ns), we expect this to result in a linear power curve load-line starting at the 15% of peak power.

RESULTS

Microarchitecture and Latency

Figure 6 shows the microarchitectural diagram with the target latency numbers of a UCIe implementation transporting CXL (PCIe) protocol, delivering 128 GB/s/direction of raw bandwidth with an internal clock frequency of 2 GHz. This means it can be an advanced package at 16 Gigabits/secor a two-module standard package at 32 G. Other frequency and bandwidth will have similar latency characteristics by adjusting the data path width and frequency.

The analog PHY has the Tx/Rx drivers along with other circuits for clocking, tracking, and the sideband. There is a first-in–first-out (FIFO) in each direction to adjust the frequency of the logic domain (here 2 GHz,

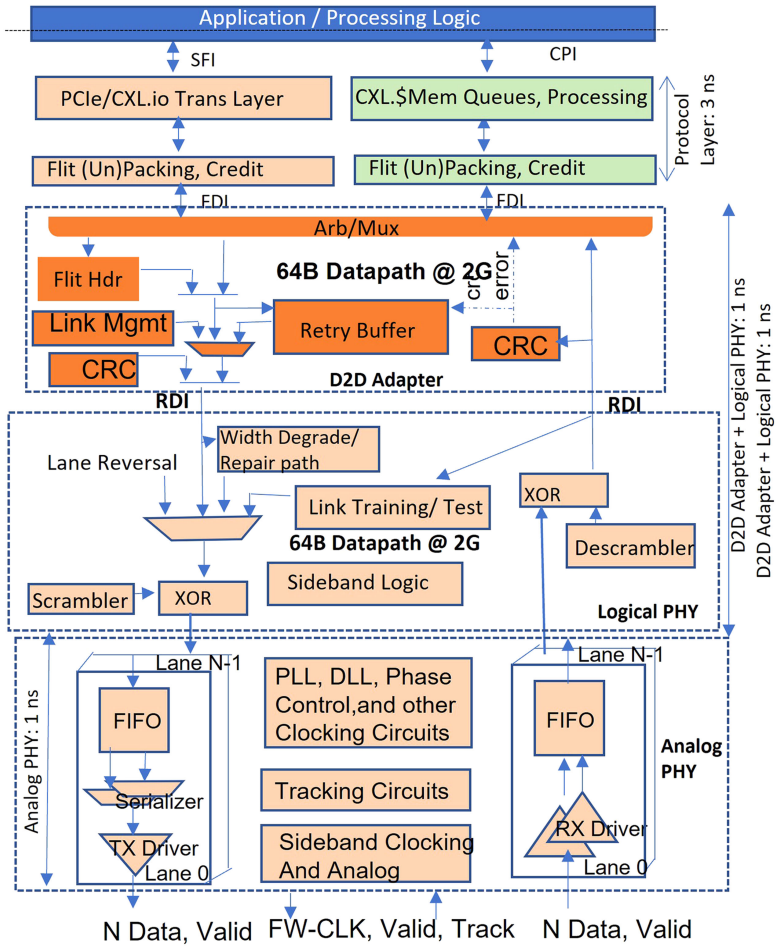


FIGURE 6. Microarchitecture with target latency.

typically a fraction of the forwarded clock frequency) and to adjust for any drift. The transmit path has multiplexers to serialize. On the receive path, deserialization is done through the FIFO. With a 2-GHz logic clock, analog PHY transmit and receive path is expected to consume 1/2 ns each, resulting in a 1-ns roundtrip.

The logical PHY is expected to have a single level of ex-or for each received/transmitted data bit for (de) scrambling. The (de)scrambler logic precomputes and registers this bit from the previous cycle for latency optimization, as shown in Figure 6. We coded the CRC logic (transmit and receive) with five-levels of gates. The flit Hdr generation can happen in parallel before the data arrives since that is tied to the arb/mux decision. The rest of the data path is mux/demux, as shown in Figure 6. With these, the expected latency of roundtrip from FDI to the analog PHY boundary is 1 ns, as shown in Figure 6. Thus, the FDI to bump round-trip latency is expected to be 2 ns through individual implementations and may have different numbers depending on process technology, placement of blocks, logic frequency, etc. Since the CXL.$Mem path does not have the link layer part, the latency between FDI and CPI will be 3 ns.[13]

Bandwidth Efficiency

Bandwidth efficiency is defined as the number of data payload bytes transferred over the total number of bytes transferred. Thus, transaction header, flit header, CRC, and link layer payload (e.g., credit, sequence number, etc.) are all considered overhead. We evaluate the relative bandwidth efficiency with the 256-byte flit [the unoptimized version in Figure 4(b) with 15 slots of CXL.cache/mem, Figure 5(a)] as the baseline. We evaluate the 256 byte with optimization [see Figure 4(b) with 16 slots of CXL.cache/mem, Figure 5(a)], 256-byte LO without optimization [see Figure 4(c) with 15 slots of CXL.cache/mem, Figure 5 (b)], 256-byte LO with optimization [see Figure 4(c) with 16 slots of CXL.cache/mem, Figure 5(c)], and the

TABLE 1. Bandwidth efficiency for different flit modes.

	256-byte flit: unopt (1H, 14 G-slots–15 slots): Baseline	256-byte flit: opt (1H, 1HS', 14 G-slots–16 slots)	256-byte LO flit: unopt (1H, 1HS, 13 G-slots–15 slots)	256-byte LO flit: Opt (2HS, 14 G-slots–16 slots)	264-byte flit 16 G-slots
CXL.io/PCIe	236B TLP	244B TLP	236B TLP	244B	256B
B/W improvement	1.0	1.03	1.0	1.03	1.08
CXL.cache/mem slot efficiency	15/16 = 0.94	1	0.94	1	1
CXL.Mem: 1R 0W (M2S: 1 Req for Read; S2M: 1 DRS and 4 Data)					
M2S slots	1 [no data – any slot]				
S2M slots*	4.43[a, b]	4.57[c, d]	4.62[d]	4.57[d]	4+1/3 = 4.33[e]
M2S B/W Eff	0	0	0	0	0
S2M B/W Eff	0.85	0.88	0.81	0.88	0.92
B/W improvement	1.0	1.04	0.95	1.04	1.08
CXL.Mem 1R1W (M2S: 2 Req, 4 Data; S2M: 1 DRS, 1 NDR, 4 Data)					
M2S slots*	6[e]				
S2M slots	4.78[b]	4.88[b]	4.89[b]	4.88[b]	4.67[b]
M2S B/W Eff	0.625	0.67	0.625	0.67	0.67
S2M B/W Eff	0.625	0.67	0.625	0.67	0.67
B/W improvement	1.0	1.07	1.0	1.07	1.07
CXL.Mem 2R1W (M2S: 3 Req, 4 Data; S2M: 2 DRS, 1 NDR, 8 Data)					
M2S slots	7[e]				
S2M slots*	9.20[b]	9.39[b]	9.42[b]	9.39[b]	9
M2S B/W Eff	0.41	0.43	0.40	0.43	0.44
S2M B/W Eff	0.82	0.85	0.80	0.85	0.89
B/W improvement	1.0	M2S 1.05/S2M 1.04	0.98	M2S 1.05/S2M 1.04	M2S 1.07/S2M 1.09
CXL.Cache 100% Read_Current: D2H: Rd_Curr–1 slot; H2D: 1 DH + 4 Data					
D2H slots	1				
H2D slots*	4.32[b]	4.57[d]	4.62[d]	4.57[d]	4.25[e]
D2H B/W Eff	0	0	0	0	0
H2D B/W Eff	0.87	0.88	0.81	0.88	0.94
B/W improvement	1.0	1.01	0.93	1.01	1.08
CXL.Cache 100% Wr: D2H: 2 Req (Rd_Own, Dirty_Evict) + DH (Data Hdr) + 4 Data; H2D: 2 Rsp (GO, Wr Pull) + DH + 4 Data					
D2H slots*	6.25[e] (2 Req, 0.25G DH, and 4 data)				
H2D slots	5.25[e]	4.84[b]	4.83[b]	5.13[b]	4.92[b]
D2H B/W Eff	0.6	0.64	0.6	0.64	0.64
H2D B/W Eff	0.6	0.64	0.6	0.64	0.64
B/W improvement	1.0	1.07	1.0	1.07	1.07

Notes: For CXL.Memory, xRyW indicates *x* Read and *y* Read w/Data (RwD) requests, each request occupying one slot (any type) plus 4*y* data slots (G-slots only) in the M2S (e.g., CPU to memory) direction. For the S2M (e.g., memory to CPU) direction, reach read results in one DRS and 4 Data and each write results in one NDR transaction. Thus, this results in 4*x* data slots, each occupying only one G-slot. In addition, S2M will have *x*+*y* DRS/NDR response headers; two response can be fit in one H/HS slot and three response can fit one G-slot. The bandwidth efficiency in each direction is calculated by taking the max of the M2S and S2M slots, indicated by an* in the table, and using that to divide the slots that carry data in that direction, multiplied by the slot efficiency (number of H/HS/G-slots /16).

For CXL.cache, the same rules apply. D2H represents device to host CPU, and H2D represents the host CPU to device. For these calculations, a D2H or an H2D Req will occupy one slot (any type). A G-slot can have 3 H2D Rsp, 4 D2H Rsp, 4 D2H DH, or 4 H2D DH. 1 H-slot can have 2 H2D Rsp, 4 D2H Rsp, 4 D2H DH, or 3 H2D DH. An HS slot can have 2 H2D Rsp, 3 D2H Rsp, 3 D2H DH, or 3 H2D DH. Each data can only be in a G-slot.

[a]2 DRS in 1 H-slot can populate 8 G-slots with data, based on the 1R0W mix that is 4.5 slots for 4 data ratio. Remaining 6 G-slots will use 3 DRS in 1 G-slot to populate data that is 4.33 slots for 4 data ratio. Thus, we get (4.5*9 + 4.33*6)/15 = 4.43.

[b]All those with a superscript[b] have the same mechanism as[a] where the H/HS slots are not enough to populate the remaining G-slots with data in a flit. So we use the averaging mechanism similar to[a] to derive the number of slots.

[c]With 2 H/HS slots per flit, we can have 4 DRS, which is 4 cache lines that need 16 G-slots based on the 1R0W mix. Given that we have 14 G-slots, the H/HS slots remain underutilized. Thus, the average slots is 4 data x 16 slots/14 G-slots = 4.57.

[d]All those with a superscript[d] have the same mechanism as[c] where the H/HS slots are underutilized, and the number of G-slots divided by 16 should be used as the multiplier.

[e]For the G-slots only case or the case where each header takes a slot and there is enough percentage of headers to occupy all the H/HS/HS* slots, we just average the number of slots required. For example, 1 DRS needs 1/3 slot as 3 DRS can fit in 1 G-slot.

264-byte [see Figure 4(d) with 16 slots of CXL.cache/mem, Figure 5(d)] against the baseline of 256-byte unoptimized flit. The results with the methodology described are given in Table 1. As expected, the 256-byte LO has lower effective bandwidth, and the 264-byte flit outperforms all with a bandwidth improvement of 7%–9% over baseline. The two bandwidth-optimized versions of 256 byte as well as 256-byte LO perform identically with a 1%–7% improvement over the baseline. The two bandwidth-optimized versions offer bandwidth improvement over the baseline for the same power, whereas the 264-byte flit has a 3% power penalty due to the use of two spare wires.

Performance Impact of Retry

Let p_{ber} denote the probability of a bit error, n is the number of bits in a flit ($n = 1,024$), w is the link width ($w = 16$ or 64 per module), f *is* the frequency (4/8/12/16/24/32 GT/s), p_e is the prob of e errors in a flit, p_{retry} is the probability of retry of a flit, and T_R is the end-to-end retry latency in nanoseconds (expected to be <5 ns given the 2-ns latency of each die) covering N_R flits

$$p_{\text{retry}} = 1 - (1 - p_{\text{ber}})^n \approx n \times p_{\text{ber}} \quad (1)$$

$$N_R = \frac{T_R w f}{n} \quad (2)$$

$$\text{B/W loss w/retry} = 1 - (1 - p_{\text{retry}})^{N_R} \approx N_R \times p_{\text{retry}}. \quad (3)$$

Thus, the bandwidth loss due to retry is negligible. For example, a 64-lane link at 32G has the bandwidth loss of 10^{-9}% when $p_{\text{ber}} = 10^{-15}$.

Reliability

We use FIT (number of failures, i.e., CRC aliasing, when CRC is used, in 1^9 h) as the measure of reliability. Since the CRC is guaranteed to detect up to three errors and any odd number of errors, aliasing happens with a probability of 2^{-16} when we have even number of errors $>=4$ in a flit. The FIT can be derived as follows:

$$p_{\text{flit\_alias}} = (p_4 + p_6 + p_8 + \ldots) \times 2^{-16} \approx p_4 \times 2^{-16}$$
$$= \binom{n}{4}(1 - p_{\text{ber}})^{n-4} p_{\text{ber}}^4 2^{-16} \approx \frac{n^4}{4!} \times p_{\text{ber}}^4 \times 2^{-16} \quad (4)$$

$$\text{No of flits in 109 h:}\ N = \frac{10^9 \times 10^9 \times 3600 \times w \times f}{n} \quad (5)$$

$$\text{FIT} = 1 - (1 - p_{\text{flit\_alias}})^N \approx N \times p_{\text{flit\_alias}}. \quad (6)$$

For a 64-lane link at 32 GT/s and $p_{\text{ber}} = 10^{-15}$, FIT $= 5 \times 10^{-33}$, which is practically 0.

When parity is used ($p_{\text{ber}} = 10^{-27}$), the FIT has two components. An odd number of errors cause a fatal error (detected, uncorrectable error), whereas with non-0 even number of errors, causes aliasing with potential silent data corruption (SDC). As derived in the following, both are acceptable numbers:

$$\text{FIT, SDC: } N \times (p_2 + p_4 + ..) \approx N \times \frac{n^2}{2} \times p_{\text{ber}}^2 = 3.77 \times 10^{-36} \quad (7)$$

$$\text{FIT, DUE: } N \times (p_1 + p_3 + ..) \approx N \times n \times p_{\text{ber}} = 7.37 \times 10^{-3}. \quad (8)$$

TABLE 2. Realized power savings for various ratios for 64-lane and 16-lane link with t_{lp} of 0.5 and 0.125, respectively.

x	y	B/W Utilization	64-Lane Pc/Pmax	64-Lane Best Pc/ achieved Pc	16-Lane Pc/Pmax	16-Lane Best Pc/ achieved Pc
0	1	0	0.15	1	0.15	1
1	9	10%	0.28	0.85	0.25	0.96
2	18	10%	0.26	0.92	0.24	0.98
4	36	10%	0.25	0.96	0.24	0.99
8	72	10%	0.24	0.98	0.24	0.99
1	3	25%	0.47	0.77	0.39	0.93
2	6	25%	0.42	0.87	0.38	0.96
4	12	25%	0.39	0.93	0.37	0.98
8	24	25%	0.38	0.96	0.37	0.99
1	1	50%	0.79	0.73	0.63	0.92
2	2	50%	0.68	0.84	0.60	0.96
4	4	50%	0.63	0.92	0.59	0.98
8	8	50%	0.60	0.96	0.58	0.99
1	0.33	75%	1.00	0.79	0.87	0.91
2	0.66	75%	0.95	0.83	0.83	0.95
4	1.32	75%	0.87	0.91	0.81	0.98
8	2.64	75%	0.83	0.95	0.80	0.99
1	0	100	1	1	0.15	1

Power Savings With Dynamic Clock Gating Across Workloads

The dynamic, hardware-controlled, low-power state can save power. The extent of power savings depends on how long the link is idle minus the entry and exit times to/from the low power state. If we assume an average pattern of x flits ($x > 1$) followed by y flits worth of link idle, and t_{lp} the low-power entry and exit time, as measured by flits, with a 0.15 peak power consumption when the stack is clock gated, the power consumed (P_c) can be expressed as a fraction of peak power ($P_{\max}$) as follows (x, y, and f can be fraction)

$$P_c = P_{\max}\left(\frac{x + t_{\text{lp}}}{x + y} + \frac{y - t_{\text{lp}}}{x + y} \times 0.15\right)$$
$$= \frac{P_{\max}}{x + y}\left(x + 0.15y + 0.85t_{\text{lp}}\right). \quad (9)$$

Table 2 summarizes the power savings. As expected, the more we can queue up flits and keep the link idle longer (i.e., larger x and y for the same utilization), P_c value drops and gets closer to the theoretical number. However, a larger value of x or y indicates a larger latency due to potential delay in sending the transaction out. A value of $x = 4$ for advanced package is the sweet spot, resulting in a delay of <3 ns in each hop at 32 GT/s. For the standard package, a greedy scheme with $x = 1$ seems to

yield reasonable power savings since it takes longer to transmit the flit. Our approach offers the designer the choice to make the power and latency tradeoff dynamically depending on the system needs at the point.

OUR APPROACH FOR UCIE 1.0 OFFERS COMPELLING POWER-EFFICIENT, RELIABLE, AND COST-EFFECTIVE PERFORMANCE WITH PLUG-AND-PLAY AND COMPLIANCE ASPECTS ADDRESSED UPFRONT.

CONCLUSION

UCIe will drive an open chiplet ecosystem that will unleash innovations across the compute continuum. Our approach for UCIe 1.0 offers compelling power-efficient, reliable, and cost-effective performance with plug-and-play and compliance aspects addressed upfront. We foresee several generations of UCIe with new innovations allowing an ensemble of chiplets offering different capabilities for the customer to choose from that best addresses their application requirements.

REFERENCES

1. G. Moore, "Cramming more components onto integrated circuits," *Electronics*, vol. 38, no. 8, pp. 114–117, Apr. 1965.
2. N. Nassif et al., "Sapphire rapids: The next-generation Intel Xeon scalable processor," in *Proc. IEEE Int. Solid-State Circuits Conf.*, 2022, pp. 44–46, doi: 10.1109/ISSCC42614.2022.9731107.
3. D. D. Sharma, "Universal Chiplet Interconnect Express (UCIe): Building an open chiplet ecosystem," UCIe, Beaverton, OR, USA, White Paper, 2022. [Online]. Available: https://www.uciexpress.org/general-8
4. D. D. Sharma, G. Pasdast, Z. Qian, and K. Aygun, "Universal Chiplet Interconnect Express (UCIe): An open industry standard for innovations with chiplets at package level," *IEEE Trans. Compon., Packag. Manuf. Technol.*, vol. 12, no. 9, pp. 1423–1431, Sep. 2022.
5. "Universal Chiplet Interconnect Express (UCIe) specification rev 1.0," Feb. 17, 2022. [Online]. Available: www.uciexpress.org
6. PCI-SIG, "PCI Express base specification revision 5.0, version 1.0,"May 28, 2019. [Online]. Available: https://picture.iczhiku.com/resource/eetop/SYkDTqhOLhpUTnMx.pdf
7. PCI-SIG, "PCI Express base specification revision 6.0, version 1.0," Jan. 11, 2022.
8. CXL Consortium, "Compute eXpress Link 2.0 specification," Sep. 9, 2020. [Online]. Available: www.computeexpresslink.org
9. IEEE Electronics Packaging Society, "Heterogeneous integration roadmap, 2021 edition," [Online]. Available: https://eps.ieee.org/hir
10. R. Mahajan et al., "Embedded multi-die interconnect bridge (EMIB) – A localized, high density multi-chip packaging (MCP) interconnect," *IEEE Trans. Compon. Packag. Manuf. Technol.*, vol. 9, no. 10, pp. 1952–1962, Oct. 2019.
11. D. D. Sharma, "Universal Chiplet Interconnect Express (UCIe): An open standard for developing a successful chiplet ecosystem," *IEEE Electron. Packag. Soc. Newslett.*, Jul. 2022. [Online]. Available: https://eps.ieee.org/images/files/TC_article_Universal_Chiplet_Interconnect_Express_UCIe.pdf
12. CXL Consortium, "Compute eXpress Link 3.0 specification," Aug. 2022. [Online]. Available: www.computeexpresslink.org
13. D. D. Sharma, "Compute eXpress Link: An open-industry standard enabling heterogeneous data-centric computing," in *Proc. IEEE Symp. High-Perform. Interconnects*, 2022, pp. 5–12.
14. D. D. Sharma, "PCI Express 6.0 specification: A low-latency, high-bandwidth, high-reliability, and cost-effective interconnect with 64.0 GT/s PAM-4 signaling," *IEEE Micro*, vol. 41, no. 1, pp. 23–29, Jan./Feb. 2021.
15. D. D. Sharma, "PCI Express 6.0 at 64.0 GT/s with PAM-4 signaling: A low latency, high bandwidth, high reliability, and cost-effective interconnect," in *Proc. IEEE Symp. High-Perform. Interconnects*, 2020, pp. 1–8.
16. L. C. T. Lee, Y. Chang, S. Huang, J. On, E. Lin, and O. Yang, "Advanced HDFO packaging solutions for chiplets integration in HPC application," in *Proc. IEEE 71st Electron. Compon. Technol. Conf.*, 2021, pp. 8–13.
17. S. Y. Hou et al., "Integrated deep trench capacitor in Si interposer for CoWoS heterogeneous integration," in *Proc. IEEE Int. Electron Devices Meeting*, 2019, pp. 19.5.1–19.5.4.
18. P. K. Huang et al., "Wafer level system integration of the fifth generation CoWoS-S with high performance Si interposer at 2500 mm^2," in *Proc. IEEE 71st Electron. Compon. Technol. Conf.*, 2021, pp. 101–104, doi: 10.1109/ECTC32696.2021.00028.

DEBENDRA DAS SHARMA is an Intel Senior Fellow, Santa Clara, CA, 95054, USA. Sharma received a Ph.D. degree in computer systems engineering from the University of Massachusetts, Amherst, MA, USA. He is a Senior Member of IEEE. Contact him at debendra.das.sharma@intel.com.