BurstLink: Techniques for Energy-Efficient Video Display for Conventional and Virtual Reality Systems

Jawad Haj-Yahya
ETH Zürich

Jisung Park
ETH Zürich

Rahul Bera
ETH Zürich

Juan Gómez Luna
ETH Zürich

Taha Shahroodi
ETH Zürich, TU Delft

Jeremie S. Kim
ETH Zürich

Efraim Rotem
Intel Corporation

Onur Mutlu
ETH Zürich

ABSTRACT

Conventional planar video streaming is the most popular application in mobile systems. The rapid growth of 360° video content and virtual reality (VR) devices is accelerating the adoption of VR video streaming. Unfortunately, video streaming consumes significant system energy due to high power consumption of major system components (e.g., DRAM, display interfaces, and display panel) involved in the video streaming process. For example, in conventional planar video streaming, the video decoder (in the processor) decodes video frames and stores them in the DRAM main memory before the display controller (in the processor) transfers decoded frames from DRAM to the display panel. This system architecture causes large amount of data movement to/from DRAM as well as high DRAM bandwidth usage. As a result, DRAM by itself consumes more than 30% of the video streaming energy.

We propose BurstLink, a novel system-level technique that improves the energy efficiency of planar and VR video streaming. BurstLink is based on two key ideas. First, BurstLink *directly* transfers a decoded video frame from the video decoder or the GPU to the display panel, completely bypassing the host DRAM. To this end, we extend the display panel with a *double remote frame buffer (DRFB)* instead of DRAM's double frame buffer so that the system can directly update the DRFB with a new frame while updating the display panel's pixels with the current frame stored in the DRFB. Second, BurstLink transfers a *complete decoded frame* to the display panel *in a single burst*, using the maximum bandwidth of modern display interfaces.

Unlike conventional systems where the frame transfer rate is limited by the pixel-update throughput of the display panel, BurstLink can always take full advantage of the high bandwidth of modern display interfaces by decoupling the frame transfer from the pixel update as enabled by the DRFB. This direct and burst frame transfer of capability BurstLink significantly reduces energy consumption of video display by 1) reducing accesses to DRAM, 2) increasing system's residency at idle power states, and 3) enabling temporal power gating of several system components after quickly transferring each frame into the DRFB.

BurstLink can be easily implemented in modern mobile systems with minimal changes to the video display pipeline. We evaluate BurstLink using an analytical power model that we rigorously validate on an Intel Skylake mobile system. Our evaluation shows that BurstLink reduces system energy consumption for 4K planar and VR video streaming by 41% and 33%, respectively. BurstLink provides an even higher energy reduction in future video streaming systems with higher display resolutions and/or display refresh rates.

CCS CONCEPTS

• **Hardware → Platform power issues**; **Displays and imagers**.

KEYWORDS

video streaming, video display, display panels, energy efficiency, data movement, mobile systems, memory, DRAM

ACM Reference Format:
Jawad Haj-Yahya, Jisung Park, Rahul Bera, Juan Gómez Luna, Taha Shahroodi, Jeremie S. Kim, Efraim Rotem, and Onur Mutlu. 2021. BurstLink: Techniques for Energy-Efficient Video Display for Conventional and Virtual Reality Systems. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO '21), October 18–22, 2021, Virtual Event, Greece.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3466752.3480085

1 INTRODUCTION

Conventional planar (i.e., 2-dimensional) video streaming is the most prevalent application in mobile devices [18]. Virtual reality (VR) video streaming is emerging as one of the most important applications in the entertainment market [117]. Cisco predicts that video streaming will generate more than 79% of mobile data traffic by 2022 [22], and Goldman Sachs predicts that around 79 million users will use VR video streaming by 2025 [49]. To provide users with an immersive experience, video formats and mobile display panels support increasingly high resolutions (e.g., 4K [6, 88]) and refresh rates (e.g., 120Hz [89]). These trends come at the cost of significantly higher energy consumption of video display, which negatively impacts the battery life of a mobile device [100]. Mobile systems need an efficient planar/VR video display architecture that provides high energy efficiency while enabling high video/display resolutions and refresh rates.

In conventional mobile systems, DRAM main memory and the display panel consume the majority of planar video streaming energy. Fig. 1 shows the measured energy consumption breakdown of a real Intel Skylake [27, 96] mobile system while streaming 30 frames-per-second (FPS) videos of full-high-definition (FHD, 1920×1080), quad-high-definition (QHD, 2560×1440), and

MICRO '21, October 18–22, 2021, Virtual Event, Greece

ACM ISBN 978-1-4503-8557-2/21/10…$15.00
https://doi.org/10.1145/3466752.3480085

4K (3840×2160) resolutions.[1] We break down the system energy consumption into three major components: DRAM (main memory), Display (all components in an LCD [21] display panel), and Others, which includes three main components: the processor (including the video decoder and display controller), network (WiFi), and storage (eMMC). As a single decoded frame's size becomes as large as tens of megabytes for a very high-resolution video (e.g., 24MB for a 4K video), DRAM alone contributes more than 30% of the total system energy consumption. State-of-the-art VR streaming schemes [68, 116], which significantly reduce the energy for Others (especially the processor) compared to unoptimized VR video streaming [37, 48], do not affect DRAM & Display and show similar trends in the energy consumption breakdown (not shown here).

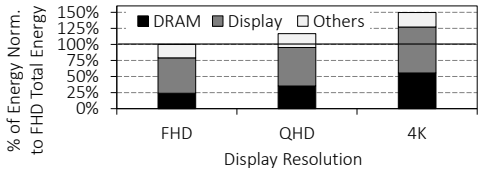

Figure 1: Energy consumption of a modern mobile computing system while streaming videos at various resolutions, normalized to FHD.

In this paper, we find two inefficiencies of conventional video display schemes that result in the underutilization of advanced architectural features widely available in modern mobile computing systems. First, the video decoder or GPU stores video frames in the DRAM main memory before the display controller (in the processor) transfers these frames from DRAM to the display panel. This system architecture causes large amount of data movement to/from DRAM as well as high DRAM bandwidth usage. Modern mobile systems commonly employ a *remote frame buffer (RFB) inside the display panel* to improve energy efficiency in a static-image display (see Section 2.3 for more details). Unfortunately, current video streaming architectures do not utilize the RFB due to a few limitations. For example, when other *planes* (e.g., graphical user interface (GUI) and cursor planes) must be displayed in addition to the video plane, the display controller (DC) must first merge them with the video plane to generate a single frame that the DC sends to the display panel. However, when there is no interaction between the video plane and other planes, we could significantly reduce energy consumption if we send the decoded video frames directly to the display and bypass host DRAM. Unfortunately, current video streaming architectures do not utilize this opportunity due to limitations in datapaths of both the processor and the display panel.

Second, conventional video processing and display schemes underutilize the high bandwidth provided by modern display interfaces. The state-of-the-art embedded-DisplayPort (eDP) [103, 105] interface supports a peak bandwidth of 25.92 Gbps [105] to cope with high-quality videos with high resolutions and frame rates. However, even when displaying a 4K 60FPS video, conventional mobile systems send each frame at a transfer rate of about 11.3 Gbps. This underutilization is because the display panel's pixel-update bandwidth dictates the frame transfer rate between the host system and the display panel. The eDP-bandwidth underutilization negatively affects energy efficiency since a long time spent for actively transferring frames reduces the time that the system can spend in low-power states. This inefficiency is expected to remain for the coming years, considering the large quality gap between displays and video contents [32, 50, 98]: while modern displays increasingly support higher resolutions and refresh rates, a significant majority of video streaming content is still in high definition (HD) or standard definition (SD) resolution [32, 98]. For example, a recent study [32] reports that 4K TVs account for about 55% of the TV market, while only less than 10% of video content provided by major streaming platforms (e.g., Netflix, Amazon Prime Video, and YouTube) is in 4K resolution with frame rates higher than 30FPS.

Based on these two major observations, we propose BurstLink, a new system-level technique that improves both planar and VR video streaming energy efficiency. BurstLink exploits advanced architectural features that are available yet underutilized in modern mobile systems. BurstLink is based on two **key ideas**. First, we introduce *Frame Buffer Bypassing*, a novel mechanism that enables *direct* transfer of a decoded video frame from the host system directly to the display panel, completely bypassing the host DRAM. To allow the system to transfer a new frame while the display panel updates pixels with the current frame in the remote frame buffer, we extend the display panel with a *double remote frame buffer (DRFB)*. Using the DRFB, the system can directly update one of the buffers of the DRFB with a new frame while updating the panel's pixels with the current frame stored in the other buffer of the DRFB. Second, we introduce *Frame Bursting*, a novel mechanism that enables transferring a *whole decoded frame* to the display panel *in a single burst* using the maximum bandwidth of modern display interfaces. Unlike conventional systems where the pixel-update throughput of the display panel limits the frame transfer rate, BurstLink can always take full advantage of the high bandwidth of modern display interfaces by decoupling the frame transfer from the pixel update using the DRFB.

The direct and bulk frame transfer enables BurstLink to significantly reduce energy consumption over a conventional video display scheme in two ways. First, by directly transferring processed video frames from the video decoder or GPU to the display panel *without buffering* them into the host DRAM (i.e., Frame Buffer Bypassing), BurstLink saves a significant fraction of the DRAM energy consumption. Second, by transferring an entire frame at the maximum display interface bandwidth (i.e., Frame Bursting), BurstLink reduces the usage of the processor and the display subsystem since they are active only during the burst period, thereby allowing the system to enter deep low-power states more frequently by turning off unused resources (e.g., the display controller, display interface, and host DRAM). In addition to planar/VR video display, Frame Bursting is also applicable to other important mobile workloads like casual gaming and office productivity [7, 12].

We evaluate BurstLink using an analytical power model that we rigorously validate with a real modern Intel Skylake [96] mobile system. Our evaluation shows that BurstLink 1) reduces system energy consumption for 4K 60FPS planar and 360° VR video streaming by 41% and 33%, respectively, and 2) provides an even higher

[1] Recent works also show similar trends on ARM system-on-chips (SoCs) [74, 113].

reduction as display resolution and/or display refresh rate increases. BurstLink also reduces system energy consumption for video conferencing, MobileMark [12], and casual gaming [7] workloads by 30%, 28%, and 27%, respectively, mainly by utilizing the Frame Bursting technique of BurstLink.

BurstLink aims to improve the energy efficiency of video streaming, one of the most important application scenarios in modern mobile computing systems. However, BurstLink can also be used in more general frame-based applications such as video capture (recording), audio streaming, video chat, social networking, and interactive games. A general takeaway from BurstLink is that using main memory (DRAM) as a *communication hub* between system components is energy-inefficient. Instead, BurstLink uses small *remote* memory near the data consumer (e.g., a display panel) to significantly reduce the number of costly main memory accesses in frame-based applications.

We make the following **key contributions** in this work:

- We provide the first study that identifies the main energy inefficiencies in traditional video display schemes of mobile systems and proposes novel techniques in both the processor and display panel to address the inefficiencies.
- We propose BurstLink, a new energy-efficient video display scheme based on two key new ideas: 1) *Frame Buffer Bypassing*, which transfers a decoded video frame directly to the display panel without buffering it in the host DRAM, and 2) *Frame Bursting*, which burst-transfers each decoded frame to the display panel as quickly as possible and thus increases opportunities for system idleness.
- We evaluate BurstLink using a thoroughly-validated analytical power model, which we open-source online [1]. Our evaluation shows that BurstLink reduces system energy consumption for 4K 60FPS planar and VR video streaming by 41% and 33%, respectively. BurstLink's energy reduction increases with higher display resolutions, making BurstLink an even better fit for next-generation high-resolution displays.

2 BACKGROUND

2.1 Mobile SoC Microarchitecture

The microarchitecture of a mobile system-on-chip (SoC) typically consists of the following components.

Main SoC Domains. A high-end mobile processor (e.g., Intel Skylake [27], AMD Kabini [16], Samsung Exynos [47]) is commonly implemented as an SoC that integrates three main domains into a single chip: 1) *compute domain*, such as CPU cores and graphics engines, 2) *IO domain*, which includes several *intellectual properties* (IPs) sharing the *IO interconnect* (e.g., display controller (DC), image signal processing engine (ISP), video decoders (VDs), video encoders (VEs)), and 3) *memory domain*, which includes the memory controller and the DRAM interface.

IO Interconnect. IO interconnects, e.g., Intel On-chip System Fabric (IOSF) [65] and ARM Advanced Microcontroller Bus Architecture (AMBA) [10, 77], are on-chip communication technologies. IO Interconnect allows multiple IPs to 1) perform peer-to-peer (P2P) communication [76] and 2) access main memory (DRAM) using direct memory access (DMA) [67].

P2P and DMA Engines. IO IPs are typically equipped with DMA and P2P engines [63]. The DMA engine enables the IP to access the main memory directly, while the P2P engine enables direct communication between two IPs without copying data to main memory. P2P reduces the data transmission delay and increases the overall available system bandwidth. DMA and P2P engines each have *control registers (CRs)* that the *IP driver* configures.

Traditional Display Subsystem. Fig. 2 shows an overview of a conventional display subsystem, which consists of five main components: two on the processor side (i.e., the *Video Decoder (VD)* and *Display Controller (DC)*) and three on the display panel side (i.e., the *embedded-DisplayPort (eDP) Receiver*, *Pixel Formatter (PF)*, and *Remote Frame Buffer (RFB)*, all inside the timing-controller (T-con)[2]). We explain video processing steps and power states in more detail in Sections 2.4 and 2.5, respectively, using Fig. 2 as a basis.

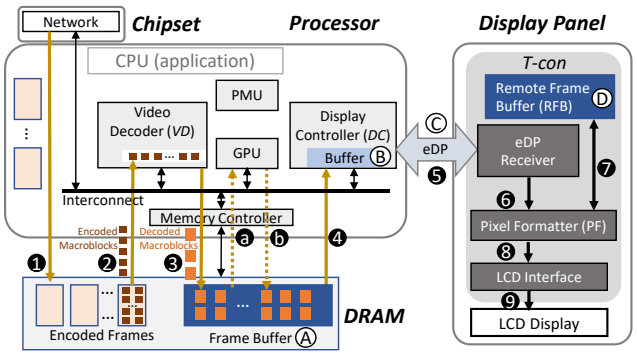

Figure 2: Overview of a conventional display subsystem.

2.2 System Idle Power States (C-states)

The Advanced Configuration and Power Interface (ACPI [102])[3] defines a processor's *idle power states*, commonly called *C-states* [102]. C-states are defined for two primary levels: 1) component level, such as thread (*TCi*), core (*CCi*), and graphics (*RCi*) C-states, and 2) SoC level, known as *package C-states* (*PCi* or *Ci*) [42, 55].

A package C-state defines an idle power state of the system (consisting of the processor, chipset, external memory devices). A system enters a specific package C-state depending on each system component's idle power state (*component C-state*). Various levels of package C-states exist to provide a range of power consumption levels with various techniques such as clock gating at the uncore level [42, 55] or a nearly complete shutdown of the system [45]. The ACPI standard includes recommendations on the C-states, but manufacturers are free to define their C-states and the SoC's behavior at each C-state. In this work, we focus on the package C-states of the Intel Skylake architecture [55], but similar idle power state definitions exist in other architectures (e.g., AMD [4] and ARM [82]). Table 1 shows all package C-states of the Intel Skylake architecture and the major conditions under which the power management unit (PMU) places the SoC into each package C-state (a similar table exists in the Intel manual [55]).

[2] A timing controller (T-con) is a circuit that processes and coordinates the coloration of the pixels in a display panel [92].

[3] ACPI [102] is an industry standard that is widely used for OS-directed configuration, power management, and thermal management of computing systems.

Table 1: Package C-states in the Intel Skylake mobile SoC.

Package C-state	Major conditions to enter the package C-state
C0	One or more cores or graphics engine **executing instructions**
C2	All cores in **CC3** (clocks off) **or deeper** and graphics engine in **RC6** (power-gated). DRAM is **active.**
C3	All cores in **CC3 or deeper** and graphics engine in **RC6**. Last-Level-Cache (LLC) may be flushed and turned off, DRAM in **self-refresh (SR)**, most IO and memory domain clocks are gated, some IPs and IOs can be **active** (e.g., **DC and Display IO**).
C6	All cores in **CC6** (power-gated) or **deeper** and graphics engine in **RC6**. LLC may be flushed and turned off, DRAM in **self-refresh**, IO and memory domain clock generators are turned off. Some IPs and IOs can be **active (e.g., VD, DC)**.
C7	Same as Package C6 while some of the IO and memory domains are **power-gated**.
C8	Same as Package C7 with additional **power-gating** in the IO and memory domains. **Only DC and Display IO are ON**.
C9	Same as Package C8 while all IPs must be off. Most VRs voltage are reduced. **The display panel can be in PSR**.
C10	**Same as Package C9** while **all SoC VRs** (except always-on VR) **are off**. **The display panel is off**.

2.3 Display Panel Refresh

Current display technologies require the host SoC to refresh the display panel several tens of times every second [69]. For example, a display panel with a *refresh rate* of 60Hz is refreshed 60 times per second. During each frame refresh window (i.e., 1/refresh_rate), the display controller (DC) inside the host SoC transfers a full-frame to the display panel by repeatedly performing three steps: the DC 1) fetches a portion of the image data from the DRAM frame buffer (Ⓐ in DRAM in Fig. 2), 2) stores the fetched image data into the DC's local buffer (Ⓑ in DC in Fig. 2), and 3) sends the buffered data to the display panel via the display interface (Ⓒ in Fig. 2).

Panel Self-Refresh (PSR). To reduce system energy consumption when displaying *static images*, VESA (Video Electronics Standards Association [104]) introduced the Panel-Self-Refresh (PSR) standard [47, 64, 92]. PSR 1) adds a local frame buffer, called a *remote frame buffer (RFB, Ⓓ in Display in Fig. 2*), into the panel T-con to store *one* frame, and 2) defines a protocol in which the DC notifies the display panel of an unchanged image. These changes enable the PSR mode, where the panel performs self-refresh using the static image stored in the RFB *without* accessing DRAM main memory. Doing so allows many host-side components, including DRAM, display interface, and DC to be powered down, reducing system energy consumption.

PSR Selective Updates (PSR2). While a mobile system is in PSR mode, the host SoC can make selective frame updates to the RFB, also known as PSR2 [54, 92, 103, 105]. This optimization can be used, for example, to turn on/off a blinking display cursor. PSR2 is supported by the newest embedded-Display Port (eDP) 1.4 [105].

2.4 Planar and VR Video Processing

Planar (i.e., 2-dimensional) video processing consists of three main stages: 1) *buffering* of encoded frames, 2) *decoding* of the buffered frames, and 3) *displaying* of the decoded frames. VR video processing requires an additional stage, called *projection*, which is performed immediately before the *displaying* stage. The video application and the device (e.g., GPU, display) drivers are responsible for *orchestrating* the different system components (e.g., programming the DMA engines and handling interrupts) during these stages.

Buffering. For video *streaming*, the network IP receives encoded video frames. Similarly, for video *playback*, the application reads the frames from storage devices. These encoded frames, each of which is hundreds of KBytes in size, are buffered in DRAM (❶ in Fig. 2). The buffering process enables the system to tolerate network bandwidth fluctuation [5] and reduce the number of storage accesses, which enables smoother and more efficient video processing.

Decoding. The video decoder (VD) reads an encoded frame from DRAM (❷) and starts decoding it. An encoded frame consists of many macroblocks, each of which stores the pixel information of a small exclusive region of the frame [106]. An encoded macroblock is the basic processing element in video decoding and typically includes 16×16, 32×32, or 64×64 pixels [71, 95, 106]. VD reads an encoded frame at macroblock granularity and buffers several encoded macroblocks (e.g., tens of KBytes [106]). Each encoded macroblock first passes through a series of stages, including entropy-decoding, inverse-DCT, and inverse quantization [71, 95, 106, 115]. Next, each macroblock is reconstructed in various ways depending on its type. VD reconstructs an *I-Type* macroblock from its neighboring macroblocks of the *same* encoded frame. In contrast, the VD reconstructs a *P-Type* and *B-Type* macroblock from the macroblocks in the *previous* and *previous/later* encoded frames, respectively, as indicated by the extra information stored in macroblock metadata (i.e., motion vectors). Finally, a decoded macroblock is written to the frame buffer in DRAM (❸) [68, 115] in preparation for the next stage of video processing.

Projection. In planar video processing, each frame can be directly displayed once decoded. However, in VR video processing, each frame must go through a set of *projective transformation* (PT) operations before being displayed.[4] Therefore, each decoded VR video frame is forwarded to the GPU (ⓐ), which performs PT operations and writes the processed frame back to the DRAM frame buffer (ⓑ) [68, 116].

Displaying. The display controller (DC) reads a decoded frame from the DRAM frame buffer (❹) at chunk granularity (e.g., 512 KB) [11, 54] and stores the frame data within its limited-size internal buffer before sending it to the display panel. During the frame transfer from the DC internal buffer to the display panel, host-side components, including the CPU, network interface, and VD, enter low-power states (e.g., DRAM is placed into the self-refresh (SR) mode [54]). DC sends the frame chunks to the display panel over the eDP interface (❺) according to the display refresh rate. For instance, if the refresh rate is 60 Hz, the DC sends the frame chunks to the display panel within a window of ~16 ms (i.e., 1/60 sec), which we refer to as the *frame window*. On the display panel side, the eDP receiver forwards the chunks to the *Pixel Formatter* (PF) (❻) [105]. PF 1) stores the decoded frame into the RFB (❼) [13, 16, 47, 105], 2) converts the decoded frame into a pixel data array, and 3) sends it to the LCD interface (❽). The LCD interface uses row and column drivers to display (❾) each frame's pixels on the LCD display.

[4] Projective transformation (PT) is the process of mapping points in the 3D space that fall within the user's viewing area to pixels on a 2D plane. In this way, the VR video can be directly displayed in the same way as a conventional planar video [68, 116].

2.5 System Power States in Video Processing

During the video processing flow, the system switches between different power states. Fig. 3 shows how the package C-state changes while an Intel Skylake mobile processor [42, 87] renders (a) a 30FPS (frames per second) video and (b) a 60FPS video, respectively, on a 60Hz display panel (i.e., the frame window is ~16 ms). At the beginning of each 16 ms window, the system resides in the *C*0 power state [42], an active state where all the system components (i.e., CPU cores, GPU, video decoder (VD), display controller (DC), eDP interface, and display panel) are running. This *C*0 state corresponds to 1) orchestration tasks that are run by the application and the drivers on the CPU cores, 2) buffering new encoded frames (❶ in Fig. 2), and 3) frame decoding by the VD (❷ and ❸). In VR video processing, the GPU performs PT operations while in *C*0 state (ⓐ and ⓑ). Once frame decoding state is complete, all the cores in the processor are powered off in the remaining frame window, while the DC 1) *periodically* fetches a chunk of the decoded frame from DRAM to the DC buffer (❹) and 2) *continuously* transfers the fetched data to the display panel (❺). The system resides in the *C*2 state while the DC fills its buffer. Once the DC buffer is full, the path to the host DRAM is closed, and the system enters the *C*8 power state [42] where only the DC, eDP interface, and display panel are active. When the DC buffer is almost empty, the DC forces the system to return to the *C*2 state such that it can open the path to the host DRAM and fetch the next chunk of the decoded frame. Note that the DC keeps transferring pixel data from its buffer to display panel at a *constant* rate (which is determined by the display's pixel update rate) regardless of system power state transitions. This power state sequence repeats until the DC transfers a complete decoded frame to the display panel.

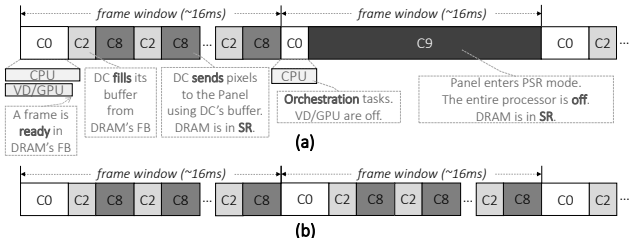

Figure 3: Package C-state timeline while displaying (a) a 30FPS video and (b) a 60FPS video on a 60Hz display panel.

A display panel with a 60Hz refresh rate can support up to 60 FPS. When the video frame rate is 30 FPS, each decoded frame is updated on the panel twice (back to back), as shown in Fig. 3(a). During *every other* frame window for a 30FPS video (e.g., second frame window in Fig. 3(a)), the RFB provides the buffered frame data to the PF (❼) to refresh the display (PSR). During this entire frame window, the whole processor, DC, and eDP interface can be disabled, enabling the system to enter the deep low-power state *C*9 [42]. The PSR technology significantly reduces the energy consumption of the entire display subsystem. Therefore, we use PSR as the baseline for evaluation in the rest of the paper. Fig. 3(b) shows the power C-state timeline while the same system plays a 60FPS video. Since the video frame rate matches the panel refresh rate, the system needs to decode and update a new frame for every frame window, which leaves no opportunity to use the PSR mode between frame updates.

Fig. 4 shows the system power consumption and package C-state residency distribution when running a web-browsing workload followed by an FHD 60FPS video streaming workload on a 60Hz display. The figure shows that, while video streaming, the system primarily resides at *C*8 (~75%), *C*2 (~15%), and *C*0 (~8%) states, while infrequently entering other package C-states for short times (e.g., the total residency of *C*3/*C*6/*C*7 states is less than 2%).

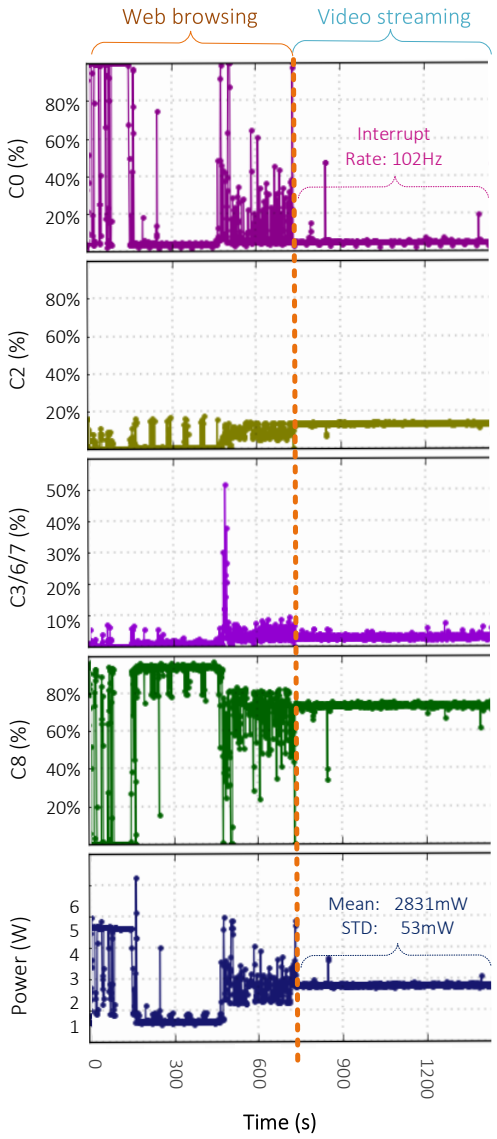

Figure 4: System power consumption and package C-state residency distribution when running a web-browsing workload followed by an FHD 60FPS video streaming workload on a 60Hz display.

3 MOTIVATION

We present our key observations that motivate a new energy-efficient video display scheme in modern mobile systems.

Observation 1: Unnecessary data movement between the display subsystem and host DRAM. In current video processing schemes, the video decoder (or GPU for virtual reality (VR) videos) stores each decoded frame into the frame buffer in the host DRAM (❸ and ❺ in Fig. 2), and the display controller (DC) fetches the decoded frame from the host DRAM to send it to the display panel (❹). Doing so is necessary when there exist other planes in addition to the video plane.[5] For example, suppose that there are four planes to display: 1) the *background* plane that is typically a static image, 2) the *video* plane that contains the video stream, 3) the *application-graphic* plane for the graphical user interface (GUI), and 4) the *cursor* plane to display the cursor. In such a case, each plane has its frame buffer in the host DRAM. DC reads data chunks from each plane's frame buffer, generates one composite chunk out of them, and sends the composite chunk to the display panel. However, when the user plays a video in full-screen mode (which is typical for planar videos and is the default for VR videos), *storing the decoded frame into the host DRAM first and then reading it again from the host DRAM is unnecessary* since there is no other plane for the DC to merge with the video plane. Bypassing the DRAM in such (common) cases would significantly reduce unnecessary data movement over the power-hungry off-chip interconnects [26, 39, 72, 91] and thus improve energy efficiency. Our goal is to enable *frame buffer bypassing* whenever it is possible, with minimal changes to current mobile SoC microarchitectures.

Observation 2: Underutilization of the eDP interface bandwidth. As explained in Section 2.5, the system alternates between power states C2 (when reading a chunk to the DC buffer) and C8 (when the buffer is full) while the DC continuously sends a *full* decoded frame (e.g., 24 MB for a 4K resolution) to the display panel, which keeps both the DC and eDP receiver active during the *entire* frame window (e.g., ~16 ms in a 60Hz refresh rate). However, the newest eDP interface [105] supports a maximum bandwidth of 25.92 Gbps, where it takes only 7.2 ms to transfer an entire 4K decoded frame. This means that the DC and eDP receiver can potentially switch to a power-saving mode for 55% of the 16ms frame window after decoding and sending the entire frame in *one burst*.

The root cause for this inefficiency in conventional systems is that the display controller, eDP receiver, and pixel-formatter (PF) are *tightly coupled*. For example, in a 4K display with a 60Hz refresh rate, the pixel update rate must be fixed to about 11.3 Gbps (i.e., 60 frames, each of which is 24 MB in size for 4K resolution, needs to be updated every second),which dictates the DC's transfer rate through the eDP interface. The PF's pixel update rate is determined by the display panel's resolution and refresh rate, in order to be aligned with the update rate of the LCD panel. Note that increasing the PF's pixel update rate without proper changes to the LCD panel would cause image *flickering* and *distortion* [64].

Our goal is to eliminate the bottleneck in the display panel so that the system directly transfers a full decoded frame from the video-decoder (or GPU) to the display panel in a burst, exploiting the display interface's maximum bandwidth. Doing so would (1) reduce the energy consumption of the host DRAM by eliminating data movement to/from the DRAM frame buffer, and (2) increase the system's idle-power state residency by reducing the usage of the processor and the display subsystem since they are active only during the burst period.

[5] A plane is a window of content to be displayed on the screen that defines an independent data stream. The final image is a composition (overlay) of different planes in a pre-defined order of superposition [62].

4 BURSTLINK DESIGN

To address the two inefficiencies discussed in Section 3, we propose BurstLink, a novel system-level technique that improves the energy efficiency of planar and VR video streaming. BurstLink is based on two key mechanisms: *Frame Buffer Bypass* and *Frame Bursting*. *Frame Buffer Bypass directly* transfers a decoded video frame from the video decoder or the GPU to the display panel, completely bypassing the host DRAM. *Frame Bursting* transfers a *complete decoded frame* to the display panel *in a single burst*, using the maximum bandwidth of modern display interfaces. This section describes how the two mechanisms work to make video display and other frame-based mobile workloads more energy-efficient in modern mobile systems.

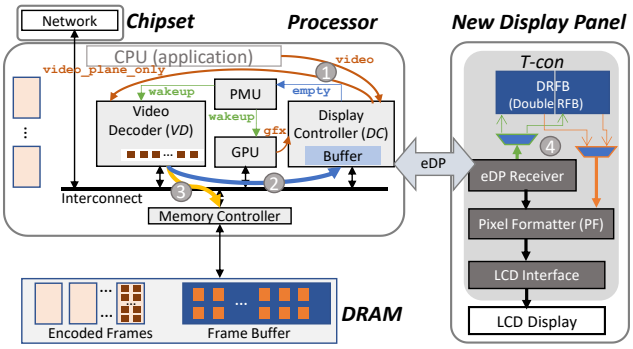

Figure 5: BurstLink video processing and display.

4.1 Frame Buffer Bypass

The Frame Buffer Bypass technique redirects the processed frame from the video decoder (VD) (or the GPU) to the display controller (DC) via the on-chip interconnect (❷ in Fig. 5) if two conditions are satisfied: 1) an asserted signal from the DC (`video_plane_only` ❶ in Fig. 5) indicating that only the video plane needs to be displayed and thus the frame should not be merged with any other frames and 2) a set flag in the VD (`single_video`) indicating that only a single video application is running. Fig. 6 depicts the package C-state timeline throughout the Frame Buffer Bypass procedure. Once the CPU completes the orchestration tasks, the VD sends the processed frame directly to the DC buffer[6] instead of first buffering it in DRAM. Bypassing DRAM allows the processor to perform this process while the system is in the low-power state *C7* (described in Table 1) instead of the higher-power *C0* power state required

[6] The DC buffer is implemented as a *double-buffer* [54], which allows the DC to send one frame's data to the display panel while it is simultaneously receiving data for another frame.

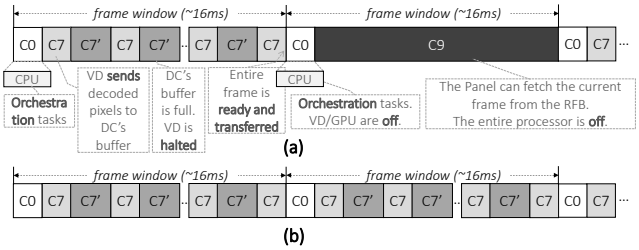

Figure 6: Package C-state timeline of a mobile processor using the Frame Buffer Bypass technique while displaying (a) a 30FPS video and (b) a 60FPS video on a 60Hz display panel.

by conventional systems. When the DC buffer is full, the VD is halted until the DC transmits the data to the DRFB in the display panel over the eDP interface. The system power state reduces even further to $C7'$, i.e., $C7$ with VD clock-gated. Once the buffer is almost empty, the DC notifies the VD via the power management unit (PMU) `empty` and `wakeup` signals (in Fig. 5) so that the VD can continue transferring the frame to the DC buffer. The display panel receives the data over the eDP interface and stores it directly into the DRFB inside the display. The pixel formatter (PF) pulls the data from the DRFB at its own IO rate and renders the pixels on the panel.

The above process of BurstLink provides two key advantages over conventional systems. First, BurstLink effectively bypasses DRAM in most of the video processing stages. Doing so reduces both DRAM bandwidth and energy consumption. Second, BurstLink reduces the orchestration overhead that requires application and driver involvement (e.g., programming the DMA engines and handling interrupts). Doing so decreases active power state residency (i.e., $C0$) and increases deep idle power state residency (e.g., $C7$ or $C7'$), thereby reducing system energy consumption.

Effectively, the Frame Buffer Bypass technique interleaves the decoding (projection) and display stages in both planar and VR video processing. Unlike conventional systems that decode (or project) the entire frame at the beginning of the frame window in power state $C0$, our technique distributes the frame decoding process across the frame window while keeping intermediate data inside the DC buffer and the DRFB. Doing so allows the decoding process to be performed in the $C7$ low-power state without any performance penalty compared to conventional systems.

Windowed Video Support. In addition to full-screen planar and VR videos, BurstLink uses the Frame Buffer Bypassing technique for *windowed planar video*,[7] such as a video clip in a window inside the browser (e.g., YouTube [18]). This feature is enabled by the selective update capability of PSR (i.e., PSR2 described in Section 2.3) that is supported by the eDP 1.4 protocol [105].

BurstLink performs windowed video streaming in two stages. In the first stage (which is the same as in conventional systems), the system prepares an initial frame with the traditional components. Suppose that a user watches a streaming video played on a web browser. In this case, the GPU prepares the graphical parts of the browser, and the VD decodes the video frames downloaded from the content server (e.g., YouTube) via the network. The frames from the GPU and the VD are stored separately in different DRAM frame buffers. Then the DC reads, scales (i.e., resizes the video frame to fit the browser window), and overlays the frames to generate the final integrated frame that is sent to the panel and stored in the DRFB (or the regular RFB in conventional systems).

The second stage starts once the host processor detects that the graphical frames from the GPU are not changing (i.e., only the video window is being updated). Then the host processor informs the DC and VD so that the display subsystem operates in PSR2 mode. In this stage, the VD continues decoding the downloaded frames and sends the decoded frames directly to the DC. The DC then directly sends only the decoded video frames (after scaling them) to the eDP receiver with the offsets of the video frames that define the updated regions in the video frames. The eDP receiver selectively updates the video only at the corresponding offsets in the DRFB, and the PF renders the entire frame into the LCD display panel.

Falling Back to the Conventional Display Mode. For all cases that BurstLink does *not* support, the system falls back to the conventional display mode. For example, whenever multiple planes are required, the system operates using the conventional video display scheme where the processor transfers all the decoded frames through the DRAM frame buffer. To enable this, BurstLink *dynamically* selects the destination of the VD/GPU output (i.e., the decoded frame) using the *destination selector* inside the VD/GPU. As shown in Fig. 5, the destination selector directs the decoded frame to the DC (❷) when 1) displaying only the video plane (`video_plane_only` signal from DC) and 2) a single video application is running (`single_video` flag in the VD). Otherwise, it stores the decoded frame into the DRAM frame buffer (❸) as in conventional schemes.

Examples of cases that BurstLink does not support include 1) when there is a graphics interrupt (to the DC [54]) indicating that a graphics plane is available (e.g., when the application's GUI appears), 2) when the system exits the PSR2 mode (e.g., in case of a windowed video display) due to a user-input interrupt (e.g., from the touch screen or keyboard), and 3) when using multiple display panels. Note that additional content like a *closed-caption* (CC) [59, 108] arrives with the video stream and does *not* require multi-plane support as VD handles such content.

4.2 Frame Bursting

In conventional display subsystems, the system sets the transfer rate between the display controller (DC) and the display panel (i.e., the eDP interface transfer rate) depending on the display resolution (i.e., the number of pixels per frame), panel refresh rate (i.e., the number of displayed frames per second), and color depth (i.e., the number of bits per pixel (bpp)) [54]. Since the PF's throughput is dictated by the pixel-update speed of the LCD panel, conventional systems align the eDP transfer rate with the PF frequency, which leads to a far lower eDP transfer rate than the maximum bandwidth of the eDP interface (e.g., up to 25.92Gbps in eDP 1.4 [105]). This far-from-optimal transfer rate bottlenecks BurstLink's entire video processing pipeline, limiting the system power states to $C7$ and $C7'$ (while decoding a frame and transferring its data to the display panel, as shown in Fig. 6).

[7] We assume windowed video only for planar video since VR video is typically streamed in full-screen mode with a Head-Mounted Display (HMD) [68].

To leverage the maximum eDP interface bandwidth, we propose *Frame Bursting*, a technique to burst transfer the decoded frame from the processor to the display panel. With the Frame Bursting technique, the display panel receives a full-frame over the eDP interface and stores it *directly* into the DRFB (❹ in Fig. 5) before transferring the frame to the PF, which removes the slow update process between the PF and the LCD panel from the critical path. The PF can fetch the frame data from the DRFB at the rate required by a given configuration (i.e., the display resolution, refresh rate, and color depth) to generate pixels and send them into the LCD. This technique reduces the utilization of the processor and the display subsystem, enabling the system to enter deep low-power states (e.g., *C*9) after quickly transferring the decoded frame to the DRFB. The processor turns off all unused resources (e.g., the DC, eDP interface, and DRAM) in such deep low-power states.

4.3 System Power States in BurstLink

Fig. 7 shows the power state (i.e., package C-state) timeline of a system that supports BurstLink (i.e., both Frame Buffer Bypass and Frame Bursting) while rendering a 30FPS (frames per second) and 60FPS planar video on a 60Hz display panel. The system resides in power state *C*0 at the beginning of a frame window, where the display driver sends the encoded video frame to the VD and then remains idle for the rest of the frame. Once the driver is idle, the system then alternate between *C*7 and *C*7′ power states, where the VD decodes the video frame in multiple chunks and sends them to the DC, which transfers the chunks to the DRFB over the eDP interface at maximum bandwidth. Once the entire decoded frame is sent to the DRFB, the system enters the deep low-power state *C*9, in which most components, including the VD, DC, and eDP interfaces (both on processor and panel sides), are power-gated. For every other frame of a 30FPS video, the system enters directly into *C*9 at the beginning of the frame (after a short driver orchestration task in *C*0) state since the entire frame already exists in the DRFB.

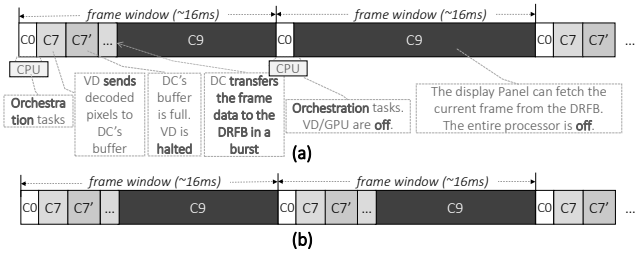

Figure 7: Package C-state timeline of a mobile processor with complete BurstLink for (a) 30FPS and (b) 60FPS videos on a 60Hz display panel.

Table 2 compares BurstLink to conventional video processing (Baseline) in terms of the average power consumption in each power state *Ci* and the *power-state residency (Residency)*, i.e., the percentage of a frame window that the system resides in state *Ci*. The last column shows the average power (AvgP) of each scheme. As shown in Table 2, BurstLink significantly reduces the average power consumption over conventional video processing by more than 40% by 1) enabling the system to reside in lower power states (e.g., *C*9) and 2) reducing the time that the system resides in higher power states (e.g., *C*0). We discuss BurstLink's power/energy savings in more detail in Sections 5 and 6.

Table 2: Power consumption comparison of BurstLink and conventional video processing on an Intel Skylake SoC when playing an FHD video at 30FPS on a 60Hz display.

	Package C-states	*C*0	*C*2	*C*7	*C*8	*C*9	AvgP (mW)
Baseline (measured)	**Power (mW)**	5940	5445	1385	1285	1090	**2162**
	Residency (%)	9%	11%	-	80%	-	
BurstLink (estimated)	**Power (mW)**	6090	5740	1530	1435	1090	**1274**
	Residency (%)	2%	-	19%	-	79%	

4.4 Implementation and Hardware Cost

Frame Buffer Bypass and Frame Bursting can be implemented independently on top of a conventional display subsystem. BurstLink incorporates both mechanisms to maximize energy efficiency. BurstLink requires mainly three changes to a conventional system.

DRFB. While using the *double* remote frame buffer (DRFB) introduces additional cost, power, and area overheads, it would not be a severe obstacle to wide adoption of BurstLink due to two reasons. First, according to the bill-of-materials (BOM[8][57]) cost estimation of Microsoft Surface Pro [97] (our evaluated system in Section 6), DRAM's BOM cost is $13.9/GB while the full HD display panel costs $100.4 [97]. Based on this estimation, doubling the existing Tcon's RFB from 24MB to 48MB will increase its BOM cost by only 32.5 cents (i.e., $13.9 \times \frac{24}{1024}$), which is only a 0.3% increase in the total BOM cost of the display panel (corresponding to 0.05% of the mobile device BOM cost). Second, according to our estimation based on Samsung's recent proposal for a cost-effective RFB implementation [52], the additional power overhead from doubling the size of the RFB is 58 mW (6% of the display panel power), which is significantly lower than BurstLink's overall power savings. Note that the DRFB does not increase the panel size since the DRAM of the DRFB is mounted on a flexible printed circuit board (FPCB) rather the panel [52].

Destination Selector. As shown in Fig. 5, BurstLink dynamically selects the destination of the video decoder's (VD) output depending on the 1) `single_video` flag in the VD and 2) `video_plane_only` (❶ in Fig. 5) signal from the display controller (DC) to the destination selector. The flag and signal can be determined using two data elements that are already stored in configuration registers in the VD and DC. First, since each video application injects its requests into the VD using the driver API [35, 74, 113, 115], the VD already keeps track of *the number of concurrently running video applications* (and their requirements) in its control and status registers (CSRs). Second, since each application also sends its requirements to the DC [54, 70], the number of used planes and each plane's type (e.g., video, graphics, or cursor) are already available in the DC CSRs (e.g., SR02 and GRX registers in Intel DC [54]). Therefore, implementing the destination selector is straightforward.

PMU Firmware Changes. BurstLink requires changes to the PMU firmware to 1) enable the processor to enter power state *C*9 when

[8]Given a product, a BOM is a list of product's immediate components with which it is built and the components' relationships.

Frame Buffer Bypassing is enabled, 2) wake up the VD (i.e., switch to power state *C*7) to resume frame data decoding once the DC buffer is empty, and 3) enable the DC to transfer the decoded frame data using the maximum eDP bandwidth when Frame Bursting is activated. We estimate that these changes increase the power-management firmware code (e.g., Pcode [36] in the Intel Skylake SoC) by only a few tens of lines,[9] which leads to only a 0.004% increase in the processor's die area for the Intel Skylake SoC [27].

4.5 Generalization of BurstLink Techniques

BurstLink aims to improve the energy efficiency of video streaming (which is one of the most important application scenarios in modern mobile systems). However, the proposed techniques can also be used in more general frame-based applications such as video capture (recording), audio streaming, video chat, social networking, and interactive games. A general takeaway from BurstLink is that using main memory (DRAM) as a *communication hub* between system components is energy-inefficient. Instead, BurstLink uses small *remote* memory (e.g., 48MB DRFB, which is only 0.3% of 16GB DRAM) near the data consumer (e.g., a display panel) or the data producer (e.g., a camera sensor) to significantly reduce the number of costly main memory accesses in frame-based applications.

5 EXPERIMENTAL METHODOLOGY

We outline our methodology for evaluating BurstLink. First, we describe our workloads. Second, we introduce our new industry-grade analytical power model for evaluating the baseline system and BurstLink. We open-source our model online [1]. Third, we discuss our process for validating our model against *power measurements* from a real modern mobile device that is based on the Intel Skylake system.

5.1 Workloads

We evaluate BurstLink with planar and VR video-streaming workloads [7, 24], which are used in standard industrial benchmarks for battery-life [2, 7, 78–80] and academic evaluations of video-streaming optimizations [15, 28, 33, 68, 73, 74, 85, 113, 115, 116]. These workloads typically assume that only a single application (e.g., video streaming in our evaluation) runs on the system. In smartphones and tablets, the currently used application typically runs in full-screen mode, while the other opened applications are normally sent to the background and moved to a suspended state. This assumption is also true of our evaluated system, the Microsoft Surface [109] when running in Tablet-Mode [99].

5.2 Analytical Power Model

Modeling the Baseline System. We develop a new analytical power model that estimates the average system power, $Power_{avg}$, within a frame window, as follows:

$$Power_{avg} = \sum_{i=0}^{10} P_{C_i} \cdot R_{C_i} + P\_en_{C_i} \cdot Lat\_en_{C_i} + P\_ex_{C_i} \cdot Lat\_ex_{C_i}$$

P_{C_i} denotes the average system power consumption in power state *Ci* (e.g., package C-states in Table 1). $P\_en_{C_i}$ and $P\_ex_{C_i}$ denote the average power consumption while entering and exiting state *Ci*, respectively. R_{C_i} denotes the residency at power state *Ci*, i.e., the percentage of the total time the system spends at power state *Ci*. $Lat\_en_{C_i}$ ($Lat\_ex_{C_i}$) denotes the latency for entering (exiting) power state *Ci*. We use a synthetic benchmark to 1) place the system in different power states, and 2) measure entry and exit latencies [90]. We obtain power state residency using processor's residency reporting counters [55]. Our power model inherently accounts for major system parameters such as 1) DC buffer size, 2) DRAM capacity, 3) DRAM bandwidth, and 4) eDP bandwidth. These parameters can directly affect each state's residency and power consumption, and the frequency with which the system switches between power states.

For power states in which the DRAM is active (i.e., not in self-refresh, such as in package *C*0 and *C*2 states as shown in Table 1), the power state's average system power consumption (i.e., P_{C_i}) depends also on DRAM power, which is correlated to DRAM bandwidth. We model DRAM power consumption in two parts: 1) *background power*, which is consumed regardless of memory access characteristics and only depends on DRAM power states (i.e., self-refresh, CKE-High (active), CKE-Low (fast power-down) [26]), and 2) *operating power*, which highly depends on DRAM read/write bandwidth [19, 26, 34]. First, to model background power, we record the time spent in each DRAM power state. We weight the power consumption in each state by the measured time values to obtain the average background power. Note that the DRAM power states in our processor are correlated to the package C-states. For example, DRAM is in the active (CKE-High) state only in *C*0 and *C*2 states while it is in the self-refresh state in all other package C-states. Second, we model the operating power by multiplying the average power per unit read/write bandwidth (e.g., 1 GB/s) by the actual read/write bandwidth consumed. To determine the average power per unit read (write) bandwidth, we 1) run a memory benchmark that generates reads (writes) at different bandwidth values (similar to [94]), 2) measure DRAM power consumption, and 3) extrapolate the power consumption per 1GB/s reads (writes).

Modeling the BurstLink System. We model the power consumption of the system enhanced with the two techniques of BurstLink using measured data from our baseline power model. The two techniques of BurstLink affect both the residency and power level at each package C-state. For example, *C*9 residency increases as the system finishes transferring each decoded frame more quickly.

We carefully model the *estimated* average power consumption (i.e., P_{C_i}) and the residency (i.e., R_{C_i}) at each power state *Ci* with BurstLink, taking into account two essential factors. 1) *Inactive* system components (e.g., power-gated DC or DRAM in the self-refresh state) in each power state. 2) Changes in each SoC component's operating frequency (e.g., the DC and eDP interface consume more power when using the maximum eDP bandwidth for Frame Bursting). We plug in the new values in our analytical model to estimate the average system power consumption when applying each of BurstLink's techniques.

5.3 Measurements and Power Model Validation

Baseline System. We use an *Intel reference design* for high-end tablet devices [114], such as the Microsoft Surface Pro [109]. Our

[9] This estimation is based on the number of lines of code for Pcode in Intel Skylake systems to fill the DC buffer [36].

baseline system is equipped with an Intel Skylake [27, 31] processor (whose specifications are summarized in Table 3), and multiple debug/configuration capabilities.

Table 3: Baseline system.

Processor	Intel i5-6300U Skylake [96], 14 nm, TDP: 15 W Frequencies: 800-2400 MHz, L3: 3 MB
Memory	LPDDR3-1866MHz [56], 8 GB, dual-channel

To validate our power model for the baseline and BurstLink systems, we carry out the following steps: 1) we measure the average power and residency at each power state in the baseline system, 2) we break down the measured power into system components, and 3) we measure the effect of frequency/bandwidth changes on the average power and residency at each power-state. We compare all our power measurements to the estimations provided by our power model.

Measurement Setup. For the system power measurements, we use a Keysight N6705B DC power analyzer [60] equipped with an N6781A source measurement unit (SMU) [61]. The N6705B is normally used for high-accuracy (around 99.975% [61]) power measurement of low-power devices (e.g., smartphones and tablets). The power analyzer measures and logs the instantaneous power consumption of different device components. Control/analysis software (14585A [60]) for data visualization and management runs on a separate laptop connected to the power analyzer.

Fig. 8 shows the power measurement setup of the Intel Skylake mobile system [27, 31, 44, 112] under study. The system has a battery and multiple power supplies (i.e., voltage regulators [40, 41]) for the mobile system components. We refer the reader to the Keysight manual [60] for more detail on the actual connections of measurement wires to the N6705B power analyzer, the design under test (DUT), DUT's battery, and control/analysis software. The power analyzer can measure the power consumption of the different power states ($C0$, $C2$, and $C7$–9) in a single experiment. We measure the residency of each C-state using the Intel VTune profiler [53] on our evaluated workloads.

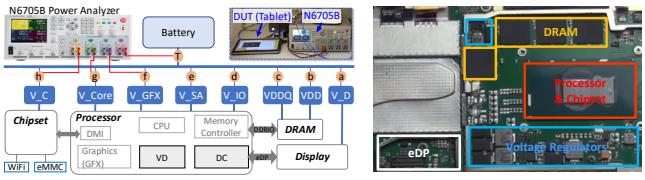

Figure 8: Power measurement of the Intel Skylake mobile system using the Keysight N6705B DC power analyzer [60] (left). Illustration of Microsoft Surface Pro tablet's system components [109] (right).

Baseline Power Measurements. We carry out multiple measurements for different system components, including the processor, DRAM, chipset, and display. As illustrated in Fig. 8, we connect the power analyzer's four channels to measurement points a-f for individual power domains, and measurement point T for the total system power drained out of the battery. For the processor power, we measure four voltage domains: 1) V_Core, the voltage supply for cores and last-level-cache (LLC), 2) V_GFX, the voltage supply for the graphics engine and the VD, 3) V_IO, the voltage supply for the IOs including the eDP DRAM, DDRIO (digital part), and the interface to the chipset, and 4) V_SA, the voltage supply for the *system agent*[10] that contains several controllers, including the memory controller and DC. Each measurement uses four analog channels with a 50-μs sampling interval.

Power Breakdown into System Components. We further break down the measured processor power consumption into processor's sub-components (e.g., VD, DC, eDP, memory-controller) using power estimation techniques [38]. Using the design characteristics of these components (such as capacitance, leakage, operational frequency, and voltage), we estimate their relative power consumption. Next, using the measured power consumption of the system, we estimate the power of each component. Note that other power estimation techniques can also be used to determine this component-level power breakdown [14, 17, 66].

Power Model Accuracy. To validate our model, we run four representative battery life workloads [7]: web browsing, light gaming, video conferencing, and video playback with multiple display resolution setups. We measure average system power (as explained above) and collect package C-states residencies along with each run. We use our analytical power model to estimate the average power consumption of these workloads. Then, we compare the measured vs. estimated average power consumption. We find that the accuracy of our analytical power model is approximately 96% for the evaluated workloads. The accuracy for each of the four mainly-used power states in our battery-life workloads, $C0$, $C2$, $C7$, and $C8$, is 97.4%, 96.2%, 95.1%, and 94.7%, respectively.

6 EVALUATION

We evaluate BurstLink against the baseline video display system (described in Fig. 2) with *five* studies. 1) We study energy reduction for four different display resolutions for planar video streaming. 2) We show energy reduction of different workloads and resolutions for VR video streaming. 3) We show the effect of frame rate on BurstLink energy reduction. 4) We compare BurstLink to state-of-the-art techniques that reduce the energy consumption of video processing. 5) We evaluate the benefits of BurstLink for other mobile workloads than video display.

6.1 Planar Video Streaming Energy Reduction

Fig. 9 shows the energy consumption of each technique of BurstLink (i.e., Frame Bursting and Frame Buffer Bypassing) and the full BurstLink, normalized to the baseline system, averaged across frame windows of 30 FPS videos displaying on a 60 Hz panel.

We make two major observations. First, BurstLink reduces the overall system energy consumption by 37% for an *FHD* display.

[10] SA stands for System Agent which houses the traditional Northbridge chip [23, 83]. SA contains several functionalities, such as the memory controller and the IO controllers/engines [39, 40, 45, 107].

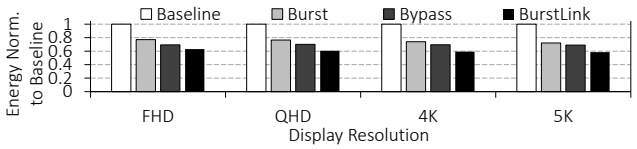

Figure 9: Total system energy reduction of Frame Bursting, Frame Buffer Bypassing, and BurstLink for 30 FPS HD video.

Frame Bursting and Frame Buffer Bypassing reduce overall energy by 23% and 31% compared to the baseline, respectively. Second, BurstLink's energy reduction increases as display resolution increases. For a 5*K* display, BurstLink reduces the overall system energy by ∼42%.

Fig. 10 compares BurstLink's system energy consumption breakdown across three major system components, i.e., DRAM, display, and others (which includes the processor, WiFi network card, and eMMC storage) with that of the baseline system. BurstLink reduces the total dissipated energy of DRAM by 3.8× and 5.7× for *FHD* and 5*K* resolutions, respectively. The higher the video resolution, the higher the DRAM's relative energy consumption out of the entire system energy consumption, and therefore, the higher the energy reduction of BurstLink. BurstLink reduces the total dissipated energy of *others* by 13.1× and 2.1× for *FHD* and 5*K* resolutions, respectively. The higher the video resolution, the lower the processor's relative energy consumption out of the entire system energy consumption, and therefore, the lower the energy reduction of BurstLink.

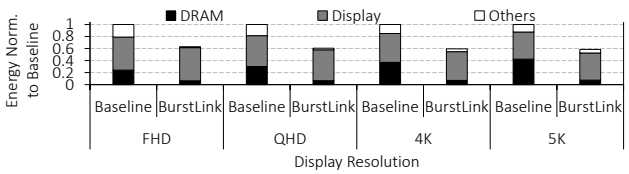

Figure 10: Energy breakdown into system components.

6.2 VR Video Streaming Energy Reduction

Fig. 11 shows the energy reduction of five 360° VR streaming workloads [24] when running with BurstLink. We assume an optimized state-of-the-art VR streaming scheme [68, 116] in the baseline, which significantly reduces the compute energy compared to traditional schemes.

We make two major observations. First, BurstLink reduces the overall system energy consumption by up to 33%. Compute-energy-dominant (mainly GPU) workloads have lower benefits compared to memory-energy-dominant workloads (as would be expected), since BurstLink greatly reduces memory energy. Second, BurstLink's benefits decrease as VR display resolution increases. This is mainly because compute energy becomes more dominant in VR workloads as display resolution increases [68, 116], which leaves less relative potential for BurstLink to save in memory energy.

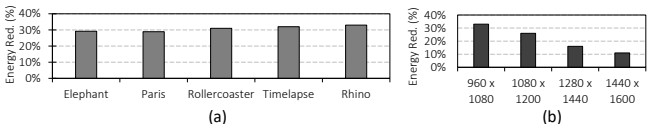

Figure 11: (a) VR video streaming energy reduction for five VR workloads. (b) Energy reduction for different VR display resolutions of the Rhino workload (other workloads are similar).

6.3 Effect of Video Frame Rate

We evaluate BurstLink with 60 FPS HD videos. BurstLink's energy reduction increases as video frame rate increases from 30 FPS (in Fig. 9) to 60 FPS. As shown in Fig. 12, BurstLink reduces overall energy consumption by 46% and 47% for *FHD* and 5*K* display resolutions, respectively.

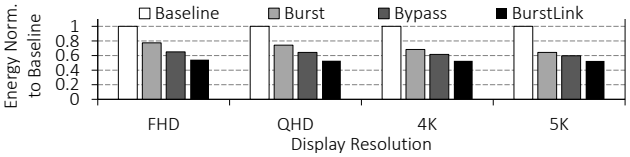

Figure 12: Total system energy reduction of Frame Bursting, Frame Buffer Bypassing, and BurstLink for 60 FPS HD videos.

Based on these results, we make two key observations. First, workloads with 60 FPS obtain higher energy benefits compared to those with 30 FPS. This result is expected since, workloads with 30 FPS have reduced baseline bandwidth and DRAM energy consumption as they utilize the optimization described in Fig. 3. Therefore, the relative DRAM energy saving of BurstLink is lower than that for workloads with 60 FPS. Second, DRAM and eDP bandwidth consumption increase as display resolution increases and/or as refresh rate increases. As a result, BurstLink's energy reduction also increases with increased display resolution and/or refresh rate. This observation makes BurstLink even more critical for future display technology, given the trends of increasing resolutions and refresh rates.

6.4 Benefits over Existing Techniques

We study the benefits of using BurstLink with state-of-the-art techniques such as 1) video compression schemes [3, 8, 9, 29, 54, 75, 86, 93], 2) a recent technique by Zhang *et al.* [115] that combines race-to-sleep [25, 30], content caching [71, 95], and display caching [111] techniques, and 3) Virtualizing IP chains (VIP) [74] technique.

Effect of Frame Buffer Compression (FBC). The evaluated Intel Skylake system supports FBC [54]. FBC compresses a decoded video frame before storing it in the frame buffer region in host DRAM to reduce the data movement overhead and DRAM bandwidth consumption. Modern FBC techniques can compress video frames with up to 50% compression rate. FBC leads to high computational overheads [8, 93] and significant storage overhead. For example,

Intel's FBC uses a region of memory reserved for graphics data to store the compressed frame buffer [54].

Moreover, FBC techniques are error-prone due to the design complexity of compression blocks and algorithms [81]. Therefore, several systems allow the display driver to enable or disable this feature [3, 8, 54].

Fig. 13 compares the energy consumption of BurstLink to the baseline system with FBC enabled. As the effect of FBC is more prominent in higher resolution videos, we only show the energy consumption for 4*K* and 5*K* resolution displays with a 60Hz refresh rate. As shown in Fig. 13, FBC with 50% compression rate can reduce overall system energy consumption by 9% for a 4*K* resolution display.

On the other hand, BurstLink eliminates the DRAM storage overhead by bypassing DRAM and directly transferring the decoded frame to PSR in burst. Besides DRAM bandwidth reduction, BurstLink enables the system to spend more time in deep idle power states where several components are power-gated (e.g., DC and eDP interface in the processor and the display panel). As shown in Fig. 13, BurstLink reduces overall energy consumption by 40.6% for 4*K* displays.

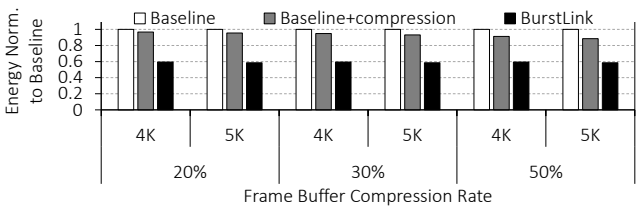

Figure 13: Energy reduction of BurstLink over a baseline with frame buffer compression (refresh rate: 60Hz).

Race-to-Sleep, Content Caching, and Display Caching. Zhang *et al.* [115] propose a mechanism to save video processing energy consumption by applying three techniques: 1) batching several encoded frames and boosting the VD frequency to decode multiple frames at once to increase system idle periods, allowing the system to enter deep idle power states, 2) caching decoded macroblocks in VD to reduce the data written to DRAM, and 3) using a cache in DC to reduce the amount of data that the system reads from DRAM. This mechanism is an extension of a prior mechanism, *short-circuiting* [113], which proposes techniques 2) and 3).

Zhang *et al.* [115]'s mechanism has significant design complexity because it requires implementing two special caching schemes, frame batching and concurrent decoding of several frames. The average DRAM bandwidth reduction of this mechanism is 34% [115] (for the three techniques combined), reducing the overall system energy consumption of 4*K* video streaming by 6%. On the other hand, BurstLink eliminates the DRAM overhead completely by transferring the decoded frame directly to PSR and reduces the overall system energy consumption by 40.6% for 4*K* displays. We conclude that BurstLink has higher energy reduction than [115]'s three techniques combined (including short-circuiting [113]).

Virtualizing IP Chains (VIP). VIP [74] proposes two main mechanisms. First, it reduces the CPU core orchestration overhead of invoking several IPs when running frame-based applications (e.g., video playback) by 1) chaining of IPs (i.e., the output of an IP in the chain is an input to the next IP in the chain) to avoid data transfer through the DRAM and 2) initiating the handling of multiple frames at the same time. BurstLink also reduces the CPU core orchestration overhead (from approximately 10% to less than 5% of the frame time) by offloading part of the orchestration task to the PMU firmware (as we discuss in Sec 4.1). One of the main advantages of BurstLink over VIP is that unlike VIP, which requires substantial changes to the software stack, BurstLink is transparent to the application and requires only a few changes to the drivers (e.g., video decoder driver).

Second, VIP reduces the traffic to DRAM by enabling direct communication between IPs (IP-to-IP chaining) instead of using DRAM. Compared to BurstLink, VIP is limited due to two main reasons. First, the traffic of frame-based applications is not always a chain. One IP waits for traffic from two (or more) other IPs/sources to complete its task in many cases. For example, in windowed video playback (discussed in Section 4.1), the DC needs the data of *multiple* frame buffers to generate the final image. Second, VIP does not solve the key bottleneck in the display data flow, where the decoding and displaying processes occupy the entire frame window (e.g., within 16*ms*) because the display panel consumes the frame data within the entire frame time. This bottleneck forces the VD, DC, and the eDP interface (at both SoC and display panel) to remain active across the entire frame. BurstLink avoids this bottleneck with the Frame Bursting technique.

In addition to the above advantages of BurstLink over VIP, our results (when modeling VIP using our power model) show that BurstLink has higher energy reduction over VIP for 4K video streaming workloads due to the ability of BurstLink to turn off the VD, DC, and the eDP interface during the majority of the frame window.

6.5 Benefits on Other Mobile Workloads

Besides video streaming, other popular mobile computing workloads can benefit from BurstLink's two techniques (i.e., Frame Buffer Bypassing and Frame Bursting). We demonstrate this with the following two examples.

First, high resolution local (i.e., not streaming) video-playback (e.g., 4K with 120/144Hz or 5K with 60Hz) can exhibit lower DRAM energy consumption when using the DRAM Frame Buffer Bypassing technique. Fig. 14(a) shows the large (more than 40%) energy reduction of our Frame Buffer Bypassing technique when playing three local video files with different resolutions and/or frame rates.

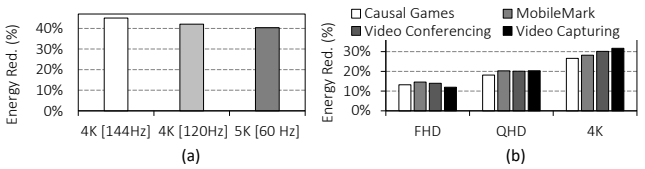

Figure 14: (a) Energy reduction of Frame Buffer Bypassing for three high video and display resolutions. (b) Energy reduction of Frame Bursting for four mobile computing workloads.

Second, we demonstrate BurstLink's benefits under four mobile workloads: video capturing, video conferencing, casual gaming [7], and MobileMark [12].[11] In a mobile device (e.g., a tablet), these workloads render images on display using one plane (usually the graphics plane). In these applications, the DC transfers the frame data from the DRAM frame buffer into the display panel. When the DC detects that a single plane exists,[12] it activates the Frame Bursting technique of BurstLink, which transfers the frame buffer from DRAM into the DRFB in a burst. Frame Bursting reduces energy consumption by 1) increasing the idle time of these workloads, as the workloads spend less time transferring the frame buffer and more time in low-power states, and 2) power-gating the DC and the eDP interface at both the processor and the display panel. Fig. 14(b) shows the large energy reduction of Frame Bursting for four workloads on a high-end tablet.

7 RELATED WORK

To our knowledge, this is the first work to leverage Panel-Self-Refresh memory to improve the energy efficiency of video processing in modern mobile computing systems. BurstLink significantly reduces costly data movement between DRAM and display subsystem components and enables frame transfer with the maximum display bandwidth, which allows the system to reside in lower power states. We briefly discuss related prior work that we classify into five categories.

Buffer Compression. Many prior works [8, 9, 29, 51, 86, 93, 110] propose various frame buffer compression techniques, such as run-length encoding (RLE) [93] and differential pulse code modulation (DPCM) [93], which increase the effective DRAM bandwidth (by reducing the amount of transferred frame data) and thus improve display processing performance. We already show that BurstLink outperforms frame buffer compression (Section 6.4).

Batching of Decoded Frames. Zhang *et al.* [115] propose three optimizations to increase the idle time (race to halt) and reduce bandwidth from VD to DRAM and DRAM to DC in video processing. Their proposals lie across batch processing, content and display caching and frequency/voltage optimization. We already demonstrate the clear advantages of BurstLink over these optimizations (Section 6.4).

Other prior works [20, 58, 101] propose different batching and pipelining techniques at a macro-block level. As BurstLink offers to achieve energy efficiency through leveraging PSR memory directly, similar techniques for batch processing can be easily combined with our proposal to increase energy improvement.

Optimizing Frequency/Voltage. Several prior works improve energy efficiency by reducing the voltage and frequency of 1) the logic used for decoding/encoding [73, 84, 85] or 2) multiple system domains [39]. BurstLink, on the other hand, does not depend on frequency/voltage scaling for its energy improvement.

Other PSR-based proposals. Prior works [13, 16, 46, 47, 64, 92, 103] use the PSR frame buffer mainly for *static images* rather than bypassing DRAM. BurstLink uses the PSR for static and dynamic images to skip costly data movement and significantly reduce DRAM energy consumption.

8 CONCLUSION

We introduce BurstLink, a new energy-efficient planar and VR video display scheme that uses display panel local memory to eliminate buffering frames in main memory. BurstLink *directly transfers* a full decoded frame from the video-decoder (or GPU) to the display panel *in a burst*, exploiting the display interface's maximum bandwidth. Doing so (1) reduces the energy consumption of the host DRAM by eliminating data movement to/from the DRAM frame buffer, and (2) increases the system's idle-power state residency by reducing the usage of the processor and the display subsystem since they are active only during the burst period. BurstLink reduces system energy consumption for 4K planar and VR video streaming by 41% and 33%, respectively, and its benefits increase as display resolution and/or refresh rate increases. As video consumption in mobile devices continues to increase sharply, along with an increase in display resolution to meet user satisfaction, BurstLink is poised to provide high energy efficiency in current and next-generation mobile processors. BurstLink's techniques can also be used in more general frame-based applications such as video capture (recording), audio streaming, video chat, social networking, and interactive games.

A general takeaway from BurstLink is that using main memory (DRAM) as a *communication hub* between system components is energy-inefficient. Instead, BurstLink uses small *remote* memory near the data consumer (e.g., a display panel) to significantly reduce the number of costly main memory accesses in frame-based applications.

ACKNOWLEDGMENTS

We thank the anonymous reviewers of ASPLOS 2021, ISCA 2021, and MICRO 2021 for feedback. We thank the SAFARI Research Group members for valuable feedback and the stimulating intellectual environment they provide. We acknowledge the generous gifts provided by our industrial partners: Google, Huawei, Intel, Microsoft, and VMware. An earlier version of this paper was placed on arxiv.org on 11 April 2021 [43].

REFERENCES

[1] BurstLink Source Code. https://github.com/CMU-SAFARI/BurstLink. 2021.
[2] 9to5mac. iPhone 11 and iPhone 11 Pro Improves on iPhone XR Stellar Battery Life. https://bit.ly/34Rlx5A. 2020.
[3] AMD. RADEON: Dissecting the Polaris Architecture. AMD Whitepaper. 2016.
[4] AMD. AMD Processor Specifications. https://bit.ly/3zQV46u. 2020.
[5] Pablo Ameigeiras, Juan J Ramos-Munoz, Jorge Navarro-Ortiz, and Juan M Lopez-Soler. Analysis and Modeling of YouTube Traffic. *ETT*. 2012.
[6] Anandtech. Lenovo Launches Legion Y740S Ultra-Thin 4K 15.6-Inch Gaming Notebook. https://bit.ly/39kxz7J. 2020.
[7] Anandtech. The Microsoft Surface Pro (2017) Review: Evaluation. https://bit.ly/2WCB3yZ. 2020.
[8] ARM. ARM Frame Buffer Compression. https://bit.ly/2WsvG5X. 2016.
[9] ARM. Transaction Elimination. https://bit.ly/2qI3rmy. 2016.
[10] ARM. AMBA: The Standard for On-Chip Communication. https://bit.ly/2UEzQ8l. 2020.
[11] ARM. ARM Mali Display Processors. https://bit.ly/33b01Yk. 2021.
[12] BAPCo. MobileMark 2014. https://bapco.com/products/mobilemark-2018. 2019.
[13] Omri Barkay, WLAN Engineer, IT Dave Buchholz, IT Dan Codorean, IT Doug DeVetter, and Baz Stevens. Evaluating Intel® Pro Wireless Display for Enterprise

[11] MobileMark is an application-based benchmark that reflects usage patterns of business mobile users in the areas of office productivity, creativity and web browsing.

[12] The number of active planes is known to the DC. The DC uses this information in multi-plane cases, where it reads from multiple frame buffers to overlay the planes [54].

Use. *Intel White Paper*. 2013.
[14] Ramon Bertran, Marc Gonzalez, Xavier Martorell, Nacho Navarro, and Eduard Ayguade. A Systematic Methodology to Generate Decomposable and Responsive Power Models for CMPs. *TC*. 2013.
[15] Amirali Boroumand, Saugata Ghose, Youngsok Kim, Rachata Ausavarungnirun, Eric Shiu, Rahul Thakur, et al. Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks. In *ASPLOS*. 2018.
[16] Dan Bouvier, Brad Cohen, Walter Fry, Sreekanth Godey, and Michael Mantor. AMD Kabini APU SoC. In *Hot Chips*. 2013.
[17] David Brooks, Pradip Bose, Viji Srinivasan, Michael K Gschwind, Philip G Emma, and Michael G Rosenfield. New Methodology for Early-stage, Microarchitecture-level Power-Performance Analysis of Microprocessors. *IBM J. Res. Dev.* 2003.
[18] Jean Burgess and Joshua Green. *YouTube: Online Video and Participatory Culture*. John Wiley & Sons. 2018.
[19] Kevin K Chang, Abdullah Giray Yaglıkçı, Saugata Ghose, Aditya Agrawal, Niladrish Chatterjee, Abhijith Kashyap, et al. Voltron: Understanding and Exploiting the Voltage-Latency-Reliability Trade-Offs in Modern DRAM Chips to Improve Energy Efficiency. *SIGMETRICS*. 2017.
[20] Tung-Chien Chen, Yu-Wen Huang, and Liang-Gee Chen. Analysis and Design of Macroblock Pipelining for H.264/AVC VLSI Architecture. In *ISCAS*. 2004.
[21] Inseok Choi, Hojun Shim, and Naehyuck Chang. Low-power Color TFT LCD Display for Hand-held Embedded Systems. In *ISLPED*. 2002.
[22] Cisco. Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update. White Paper. https://bit.ly/2Y2wqhb. 2019.
[23] Pat Conway and Bill Hughes. The AMD Opteron northbridge architecture. *IEEE Micro*. 2007.
[24] Xavier Corbillon, Francesca De Simone, and Gwendal Simon. 360-Degree Video Head Movement Dataset. In *MMSys*. 2017.
[25] Anup Das, Geoff V Merrett, and Bashir M Al-Hashimi. The slowdown or race-to-idle question: Workload-aware energy optimization of SMT multicore platforms under process variation. In *DATE*. 2016.
[26] Howard David, Chris Fallin, Eugene Gorbatov, Ulf R Hanebutte, and Onur Mutlu. Memory Power Management via Dynamic Voltage/Frequency Scaling. In *ICAC*. 2011.
[27] Jack Doweck, Wen-Fu Kao, Allen Kuan-yu Lu, Julius Mandelblat, Anirudha Rahatekar, Lihu Rappoport, et al. Inside 6th-Generation Intel Core: New Microarchitecture Code-Named Skylake. *MICRO*. 2017.
[28] Fanyi Duanmu, Eymen Kurdoglu, Yong Liu, and Yao Wang. View Direction and Bandwidth Adaptive 360 Degree Video Streaming Using a Two-tier System. In *ISCAS*. 2017.
[29] Kenneth C Dyke. Method for Reducing Frame Buffer Memory Accesses. US Patent 8,358,314. 2013.
[30] Rotem Efraim, Ran Ginosar, C Weiser, and Avi Mendelson. Energy Aware Race to Halt: A Down to EARtH Approach for Platform energy management. *CAL*. 2012.
[31] Eyal Fayneh, Marcelo Yuffe, Ernest Knoll, Michael Zelikson, Muhammad Abozaed, Yair Talker, et al. 4.1 14nm 6th-Generation Core Processor SoC with Low Power Consumption and Improved Performance. In *ISSCC*. IEEE. 2016.
[32] FlatpanelsHD. The Resolution Gap. https://bit.ly/3djTHm1. 2020.
[33] Benjamin Gaudette, Carole-Jean Wu, and Sarma Vrudhula. Improving Smartphone User Experience by Balancing Performance and Energy with Probabilistic QoS Guarantee. In *HPCA*. 2016.
[34] Saugata Ghose, Abdullah Giray Yaglikçi, Raghav Gupta, Donghyuk Lee, Kais Kudrolli, William X Liu, et al. What Your DRAM Power Models are not Telling You: Lessons from a Detailed Experimental Study. *SIGMETRICS*. 2018.
[35] Google. Grafika Video Application. https://github.com/google/grafika. 2020.
[36] Corey Gough, Ian Steiner, and Winston Saunders. CPU Power Management. In *Energy Efficient Servers: Blueprints for Data Center Optimization*. 2015.
[37] Mario Graf, Christian Timmerer, and Christopher Mueller. Towards Bandwidth Efficient Adaptive Streaming of Omnidirectional Video over http: Design, Implementation, and Evaluation. In *MMSys*. 2017.
[38] Stephen H Gunther. Managing the Impact of Increasing Microprocessor Power Consumption. *Intel Technology Journal*. 2001.
[39] Jawad Haj-Yahya, Mohammed Alser, Jeremie Kim, A Giray Yağlıkçı, Nandita Vijaykumar, Efraim Rotem, et al. SysScale: Exploiting Multi-domain Dynamic Voltage and Frequency Scaling for Energy Efficient Mobile Processors. ISCA. 2020.
[40] Jawad Haj-Yahya, Mohammed Alser, Jeremie S Kim, Lois Orosa, Efraim Rotem, Avi Mendelson, et al. FlexWatts: A Power-and Workload-Aware Hybrid Power Delivery Network for Energy-Efficient Microprocessors. In *MICRO*. 2020.
[41] Jawad Haj-Yahya, Jeremie S Kim, A Giray Yaglikci, Ivan Puddu, Lois Orosa, Juan Gómez Luna, et al. IChannels: Exploiting Current Management Mechanisms to Create Covert Channels in Modern Processors. *ISCA*. 2021.
[42] Jawad Haj-Yahya, Avi Mendelson, Yosi Ben Asher, and Anupam Chattopadhyay. Power Management of Modern Processors. In *EEHPC*. 2018.
[43] Jawad Haj-Yahya, Jisung Park, Rahul Bera, Juan Gómez Luna, Efraim Rotem, Taha Shahroodi, et al. BurstLink: Techniques for Energy-Efficient Conventional and Virtual Reality Video Display. *arXiv preprint arXiv:2104.05119*. 2021.
[44] Jawad Haj-Yahya, Efraim Rotem, Avi Mendelson, and Anupam Chattopadhyay. A Comprehensive Evaluation of Power Delivery Schemes for Modern Microprocessors. In *ISQED*. 2019.
[45] Jawad Haj-Yahya, Yanos Sazeides, Mohammed Alser, Efraim Rotem, and Onur Mutlu. Techniques for Reducing the Connected-Standby Energy Consumption of Mobile Devices. In *HPCA*. 2020.
[46] Per Hammarlund, Alberto J Martinez, Atiq Bajwa, David L Hill, Erik Hallnor, Hong Jiang, et al. 4th Generation Intel Core Processor, Codenamed Haswell. In *Hot chips*. 2013.
[47] Sodam Han, Yonghee Yun, Eunju Hwang, Wook Kim, and Young Hwan Kim. Samsung Exynos 5410 Processor-Experience the Ultimate Performance and Versatility. *White Paper*. 2013.
[48] Jian He, Mubashir Adnan Qureshi, Lili Qiu, Jin Li, Feng Li, and Lei Han. Rubiks: Practical 360-Degree Streaming for Smartphones. In *MobiSys*. 2018.
[49] C. Heather, Bellini, W. Chen amd M. Sugiyama, M. Shin, S. Alam, , et al. Virtual and Augmented Reality. https://bit.ly/3nULNE0. 2020.
[50] HowtoGeek. The Resolution Gap. https://bit.ly/3dhCGJc. 2020.
[51] Hung-ju Huang, Jo-tan Yao, and Chung-heng Chen. Method and System for Segment Encoded Graphic Data Compression. US Patent 5,748,904. 1998.
[52] Moon-Sang Hwang, Deok-Jun Choe, Do-Wan Kim, Joon-Bae Park, Jun-Woo Bae, Won-jun Choe, et al. 16-2: Cost-effective Driver IC Architecture using Low-power Memory Interface for Mobile Display Application. In *SID Symposium Digest of Technical Papers*. Wiley Online Library. 2017.
[53] Intel. Intel VTune Profiler User Guide. online, accessed March 2020. https://software.intel.com/en-us/vtune-help-window-cpu-c-p-states-platform-power-analysis.
[54] Intel. Open Source Graphics. https://bit.ly/2qQgHp7. 2016.
[55] Intel. 6th Generation Intel® Processor for U/Y-Platforms Datasheet. https://intel.ly/37rtnU7. 2020.
[56] JEDEC. JESD79-3-1.35 V DDR3L-800, DDR3L-1066, DDR3L-1333, DDR3L-1600, and DDR3L-1866. *JEDEC Std., JESD79-3-1A* 1. 2013.
[57] Jianxin Jiao, Mitchell M Tseng, Qinhai Ma, and Yi Zou. Generic Bill-of-Materials-and-Operations for High-Variety Production Management. *Concurrent Engineering*. 2000.
[58] Genhua Jin, Jin-Su Jung, and Hyuk-Jae Lee. An Efficient Pipelined Architecture for H.264/AVC Intra Frame Processing. In *ISCAS*. 2007.
[59] Scott Kellicker. Closed Captions for Live Streams. US Patent 8,782,721. 2014.
[60] Keysight. IntKeysight N6705B DC Power Analyzer. https://bit.ly/2MZ9Hhv. 2019.
[61] Keysight. IntKeysight Source Measure Units Power Modules. https://bit.ly/38kkStt. 2019.
[62] Moinul H Khan and Srikanth Rengarajan. Bandwidth-efficient Display Controller for Low Power Devices in Presence of Occlusion. In *International Conference on Consumer Electronics*. 2007.
[63] Tim Kogel, Malte Doerper, Andreas Wieferink, Rainer Leupers, Gerd Ascheid, Heinrich Meyr, et al. A Modular Simulation Framework for Architectural Exploration of On-chip Interconnection Networks. In *CODES/ISSS*. 2003.
[64] Seh Kwa, George R Hayek, Kamal R Shah, and Achintya K Bhowmik. Panel Self-Refresh Technology: Decoupling Image Update from LCD Panel Refresh in Mobile Computing Systems. In *SID Symposium Digest of Technical Papers*. 2012.
[65] Hasnain Lakdawala, Mark Schaecher, Chang-Tsung Fu, Rahul Limaye, Jon Duster, Yulin Tan, et al. A 32 nm SoC with Dual Core ATOM Processor and RF WiFi Transceiver. *JSSC*. 2012.
[66] Paul Landman. High-level Power Estimation. In *ISLPED*. 1996.
[67] Donghyuk Lee, Lavanya Subramanian, Rachata Ausavarungnirun, Jongmoo Choi, and Onur Mutlu. Decoupled Direct Memory Access: Isolating CPU and IO Traffic by Leveraging a Dual-Data-Port DRAM. In *PACT*. 2015.
[68] Yue Leng, Chi-Chun Chen, Qiuyue Sun, Jian Huang, and Yuhao Zhu. Energy-Efficient Video Processing for Virtual Reality. In *ISCA*. 2019.
[69] M Menozzi, F Lang, U Naepflin, C Zeller, and H Krueger. CRT Versus LCD: Effects of Refresh Rate, Display Technology and Background Luminance in Visual Performance. *Displays*. 2001.
[70] Microsoft. Programming Reference for Windows Driver Kit - Display. https://bit.ly/33T8D7L. 2020.
[71] Debargha Mukherjee, Jim Bankoski, Adrian Grange, Jingning Han, John Koleszar, Paul Wilkins, et al. The Latest Open-source Video Codec VP9-an Overview and Preliminary Results. In *PCS*. 2013.
[72] Onur Mutlu. Memory Scaling: A Systems Architecture Perspective. In *IMW*. 2013.
[73] Nachiappan Chidambaram Nachiappan, Praveen Yedlapalli, Niranjan Soundararajan, Anand Sivasubramaniam, Mahmut T Kandemir, Ravi Iyer, et al. Domain Knowledge Based Energy Management in Handhelds. In *HPCA*. 2015.
[74] Nachiappan Chidambaram Nachiappan, Haibo Zhang, Jihyun Ryoo, Niranjan Soundararajan, Anand Sivasubramaniam, Mahmut T Kandemir, et al. VIP: Virtualizing IP chains on Handheld Platforms. In *ISCA*. 2015.
[75] Nvidia. NVIDIA GeDorce GTX 1080. Nvidia Whitepaper. 2016.
[76] Umit Y Ogras, Paul Bogdan, and Radu Marculescu. An Analytical Approach for Network-on-Chip Performance Analysis. *TCAD*. 2010.

[77] Srinivas Patil, Abhijit Jas, Peter Lisherness, and Enrico Carrieri. Functional Fabric-based Test Controller for Functional and Structural Test and Debug. US Patent 8,793,095. 2014.

[78] Pcmag. How We Test Laptops. https://bit.ly/3lJZZ1C. 2020.

[79] Pcmag. Samsung Galaxy S20 Review. https://bit.ly/3dsF14q. 2020.

[80] Phonearena. PhoneArena Battery Test Results. https://bit.ly/36Z3Ljr. 2020.

[81] Phoronix. Linux 4.11 To Enable Frame-Buffer Compression By Default For Skylake+. online, accessed March 2020. https://bit.ly/3a59yCP. 2019.

[82] Qualcomm Technologies. Qualcomm Snapdragon 410E (APQ8016E) Processor Device Specification. online. https://developer.qualcomm.com/qfile/28813/lm80-p0436-7_f_410e_proc_apq8016e_device_spec.pdf. 2018.

[83] Sivakumar Radhakrishnan, Sundaram Chinthamani, and Kai Cheng. The Blackford Northbridge Chipset for the Intel 5000. *IEEE Micro*. 2007.

[84] Erwan Raffin, Erwan Nogues, Wassim Hamidouche, Seppo Tomperi, Maxime Pelcat, and Daniel Menard. Low Power HEVC Software Decoder for Mobile Devices. *JRTIP*. 2016.

[85] Karthik Rao, Jun Wang, Sudhakar Yalamanchili, Yorai Wardi, and Ye Handong. Application-specific Performance-aware Energy Optimization on Android Mobile Devices. In *HPCA*. 2017.

[86] Jim Rasmusson, Tomas Akenine-Möller, Jon Hasselgren, and Jacob Munkberg. Frame Buffer Compression and Decompression Method for Graphics Rendering. US Patent 8,031,937. 2011.

[87] Efraim Rotem. Intel Architecture, Code Name Skylake Deep Dive: A New Architecture to Manage Power Performance and Energy Efficiency. In *IDF*. 2015.

[88] 4K SAMPLES. Puppies Bath in 4K. online, accessed Jan 2020. https://bit.ly/39jHIS2. 2012.

[89] Samsung. Samsung Galaxy S20 Specification . https://www.samsung.com/global/galaxy/galaxy-s20/specs. 2020.

[90] Robert Schöne, Daniel Molka, and Michael Werner. Wake-up Latencies for Processor Idle States on Current x86 Processors. *Computer Science-Research and Development*. 2015.

[91] Vivek Seshadri, Yoongu Kim, Chris Fallin, Donghyuk Lee, Rachata Ausavarungnirun, Gennady Pekhimenko, et al. RowClone: Fast and Energy-efficient in-DRAM Bulk Data Copy and Initialization. In *MICRO*. 2013.

[92] Kamal Shah, Seh Kwa, and George Hayek. Extending Battery Life of Ultrabook™ Through use of Panel Self Refresh Technology (PSR). In *SID Symposium Digest of Technical Papers*. 2013.

[93] Hojun Shim, Nachyuck Chang, and Massoud Pedram. A Compressed Frame Buffer to Reduce Display Power Consumption in Mobile Systems. In *ASPDAC*. 2004.

[94] Lavanya Subramanian, Vivek Seshadri, Yoongu Kim, Ben Jaiyen, and Onur Mutlu. MISE: Providing Performance Predictability and Improving Fairness in Shared Main Memory Systems. In *HPCA*. 2013.

[95] Gary J Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. Overview of the High Efficiency Video Coding (HEVC) Standard. *TCSVT*. 2012.

[96] Simon M Tam, Harry Muljono, Min Huang, Sitaraman Iyer, Kalapi Royneogi, Nagmohan Satti, et al. SkyLake-SP: A 14nm 28-Core Xeon® Processor. In *ISSCC*. 2018.

[97] Tech Thoughts. Microsoft Surface RT & Pro: BOM & Price Estimate. https://bit.ly/3rspuY3. 2021.

[98] TheVerge. YouTube Joins Netflix in Reducing Video Quality in Europe. https://bit.ly/3jQEewe. 2020.

[99] Rob Tidrow, Jim Boyce, and Jeffrey R Shapiro. *Windows 10 Bible*. John Wiley & Sons. 2015.

[100] Toms Guide. 2020. Samsung Galaxy S20 Ultra Battery Life: 120Hz Display Takes 3 Hours Off. https://www.tomsguide.com/news/samsung-galaxy-s20-ultra-battery-life-120hz-display-takes-3-hours-off

[101] Tang-Hsun Tu and Chih-Wen Hsueh. Batch-Pipelining for H. 264 Decoding on Multicore Systems. In *DCC*. 2010.

[102] UEFI.org. Advanced Configuration and Power Interface (ACPI) specification . https://bit.ly/2ToagrG. 2021.

[103] VESA. eDP Embedded DisplayPort: The New Generation Digital Display Interface for Embedded Applications . https://bit.ly/2TE2Sps.

[104] VESA. VESA publishes Embedded DisplayPort (eDP) Standard Version 1.4a . https://bit.ly/2sglj8f. 2021.

[105] VESA. Vesa Rolls Out Production-ready Embedded Displayport Standard 1.4 For Mobile Personal Computing Devices. https://bit.ly/2KYrk03. 2021.

[106] Thomas Wiegand, Gary J Sullivan, Gisle Bjontegaard, and Ajay Luthra. Overview of the H. 264/AVC Video Coding Standard. *TCSVT*. 2003.

[107] Wikipedia. Skylake Microarchitecture. https://bit.ly/2NG8Hz2. 2019.

[108] Wikipedia. Closed captioning - Online Video Streaming. https://bit.ly/30VB96C. 2020.

[109] Wikipedia. Microsoft Surface. https://bit.ly/38N8vrS. 2020.

[110] Y Yabe, Y Aimoto, M Motomura, T Takizawa, T Miyamoto, T Iwasaki, et al. Compression/Decompression DRAM for Unified Memory Systems: a 16Mb, 200MHz, 90% to 50% Graphics-Bandwidth Reduction Prototype. In *ISSCC*. 1998.

[111] S Jae Yang, Jason Nieh, Matt Selsky, and Nikhil Tiwari. The Performance of Remote Display Mechanisms for Thin-Client Computing.. In *USENIX ATC*. 2002.

[112] Ahmad Yasin, Jawad Haj-Yahya, Yosi Ben-Asher, and Avi Mendelson. A Metric-guided Method for Discovering Impactful Features and Architectural Insights for Skylake-based Processors. *TACO*. 2019.

[113] Praveen Yedlapalli, Nachiappan Chidambaram Nachiappan, Niranjan Soundararajan, Anand Sivasubramaniam, Mahmut T Kandemir, and Chita R Das. Short-Circuiting Memory Traffic in Handheld Platforms. In *MICRO*. 2014.

[114] Rumi Zahir, Mark Ewert, and Hari Seshadri. The Medfield Smartphone: Intel Architecture in a Handheld Form Factor. *IEEE Micro*. 2013.

[115] Haibo Zhang, Prasanna Venkatesh Rengasamy, Shulin Zhao, Nachiappan Chidambaram Nachiappan, Anand Sivasubramaniam, Mahmut T Kandemir, et al. Race-to-sleep+ Content Caching+ Display Caching: A Recipe for Energy-efficient Video Streaming on Handhelds. In *MICRO*. 2017.

[116] Shulin Zhao, Haibo Zhang, Sandeepa Bhuyan, Cyan Subhra Mishra, Ziyu Ying, Mahmut T Kandemir, et al. Déjà View: Spatio-Temporal Compute Reuse for 'Energy-Efficient 360° VR Video Streaming. In *ISCA*. 2020.

[117] Michael Zyda. From Visual Simulation to Virtual Reality to Games. *IEEE Computer Society*. 2005.