BurstLink

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

Jawad Haj-Yahya *Jisung Park* *Rahul Bera* *Juan Gómez Luna*
Taha Shahroodi *Jeremie S. Kim* *Efraim Rotem* *Onur Mutlu*

SAFARI
SAFARI Research Group
safari.ethz.ch

ETH zürich

intel

TU Delft

Executive Summary

Problem: planar and virtual reality (VR) video streaming consumes significant system energy due to the high power consumption of major system components (e.g., DRAM, display interfaces, and display panel)

Goal: improve the energy efficiency of planar and VR video streaming by leveraging display panel local memory to eliminate buffering frames in main memory

Mechanism: BurstLink, a new planar and VR video streaming scheme that

- Directly transfers a full decoded video frame from the video-decoder (or GPU) to the display panel, completely bypassing the host DRAM
- Transfers a complete decoded frame to the display panel in a burst, exploiting the display interface's maximum bandwidth

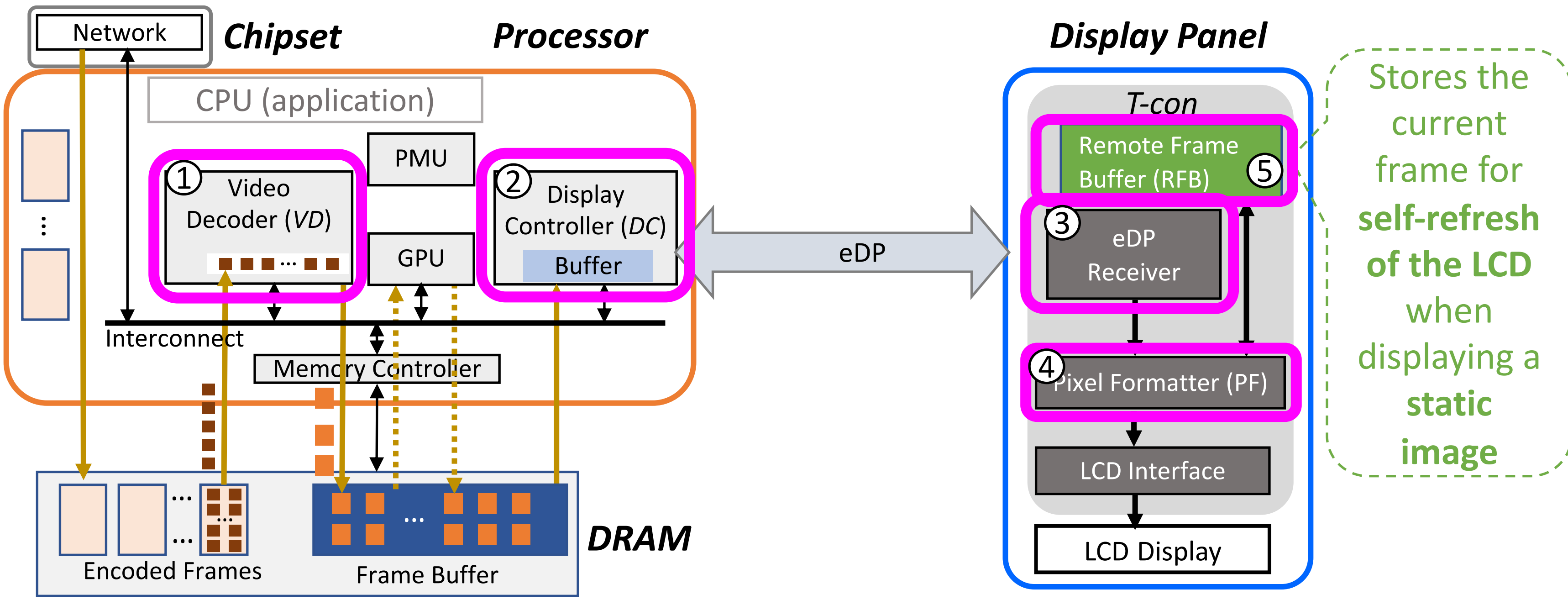
Evaluation: we evaluate BurstLink using our open-sourced analytical power model that we rigorously validate on an Intel Skylake mobile system. BurstLink:

- Reduces system energy consumption for 4K planar/VR video streaming by 41%/33%
- Provides an even higher energy reduction in future planar video streaming systems with higher display resolutions and/or display refresh rates

Presentation Outline

1. **Overview of Mobile SoC Microarchitecture**
2. Motivation and Goal
3. BurstLink
   1. Frame Buffer Bypassing
   2. Frame Bursting
4. Evaluation
5. Conclusion

Overview of a Traditional Display Subsystem



A conventional display subsystem consists of five main components:

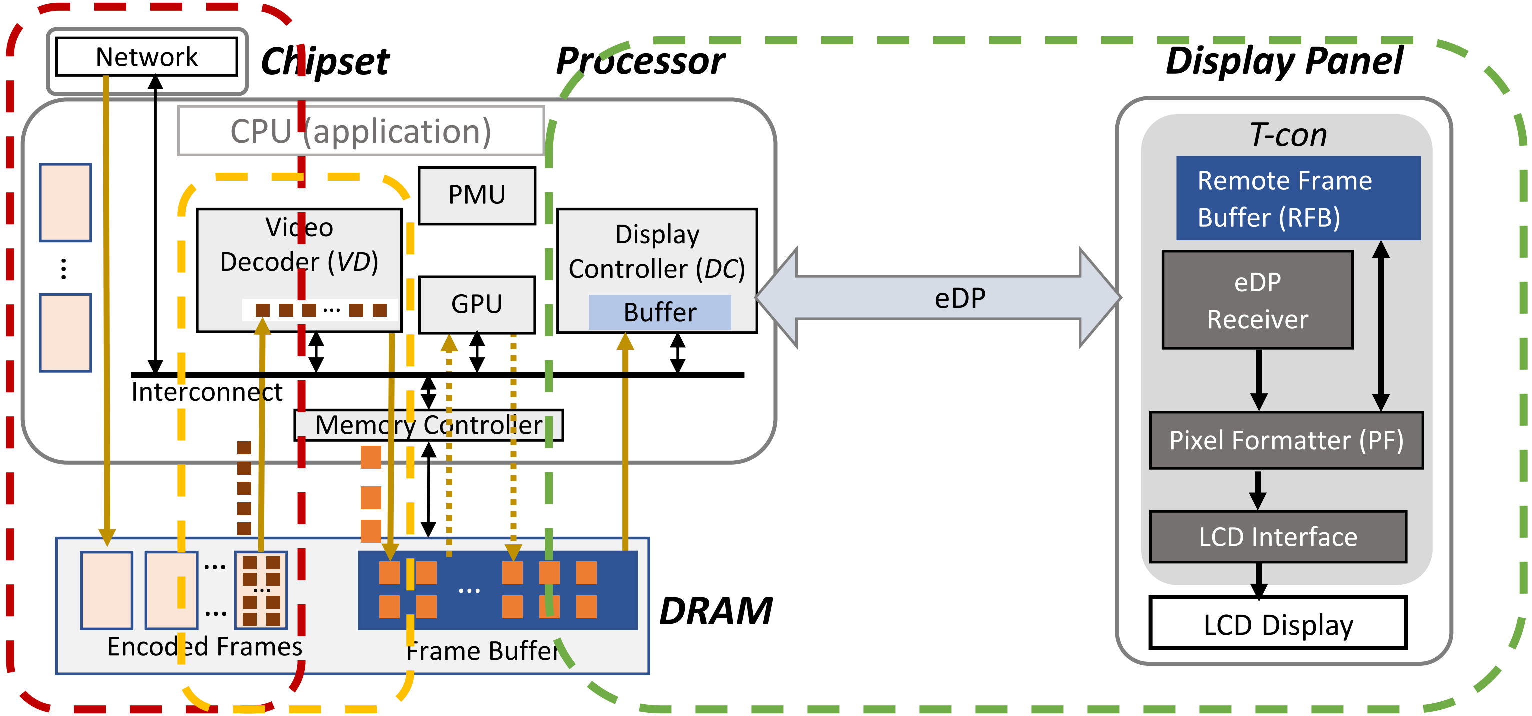
In the **processor**:

1. Video Decoder (VD)
2. Display Controller (DC)

In the **display panel**:

3. embedded-DisplayPort (eDP) Receiver
4. Pixel Formatter (PF)
5. Remote Frame Buffer (RFB)

Planar Video Processing Stages



Planar video processing consists of three main stages:

Buffering the encoded frames → **Decoding** the buffered frames → **Displaying** the decoded frames

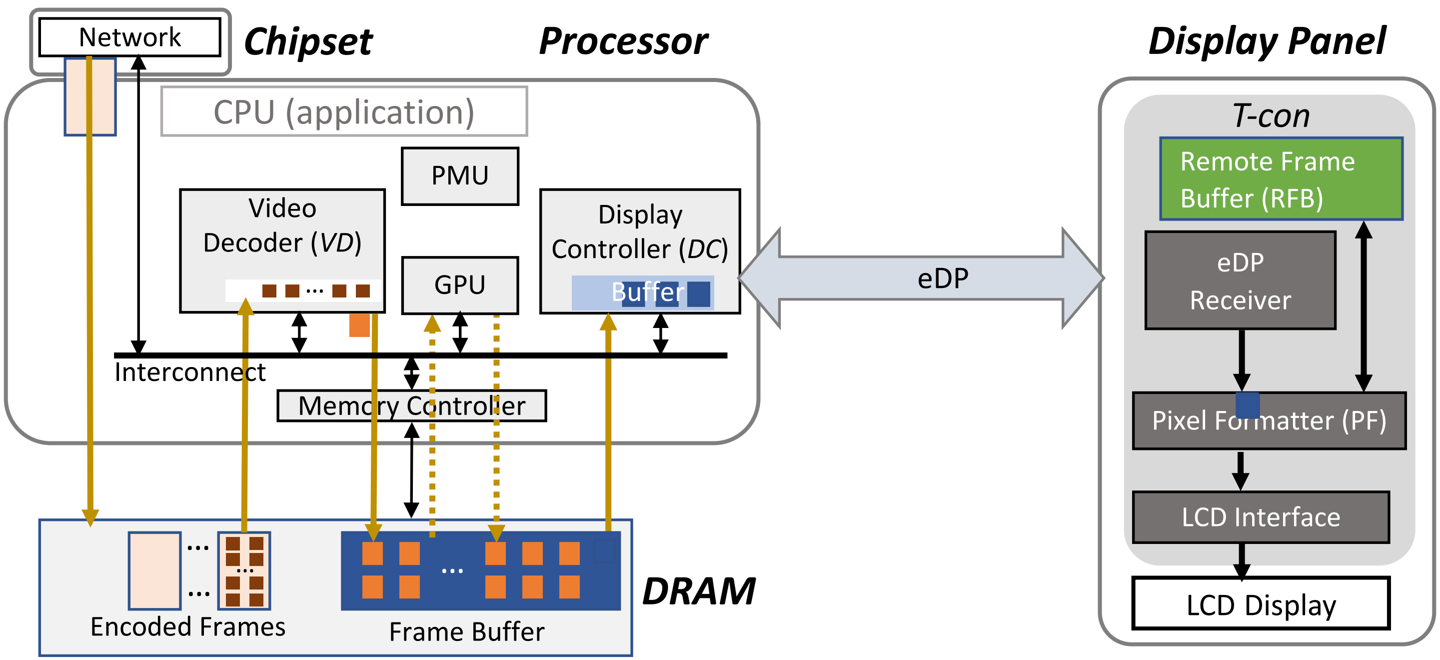
Presentation Outline

Two Problems in Video Processing

1. Unnecessary Data Movement to/from Host Memory

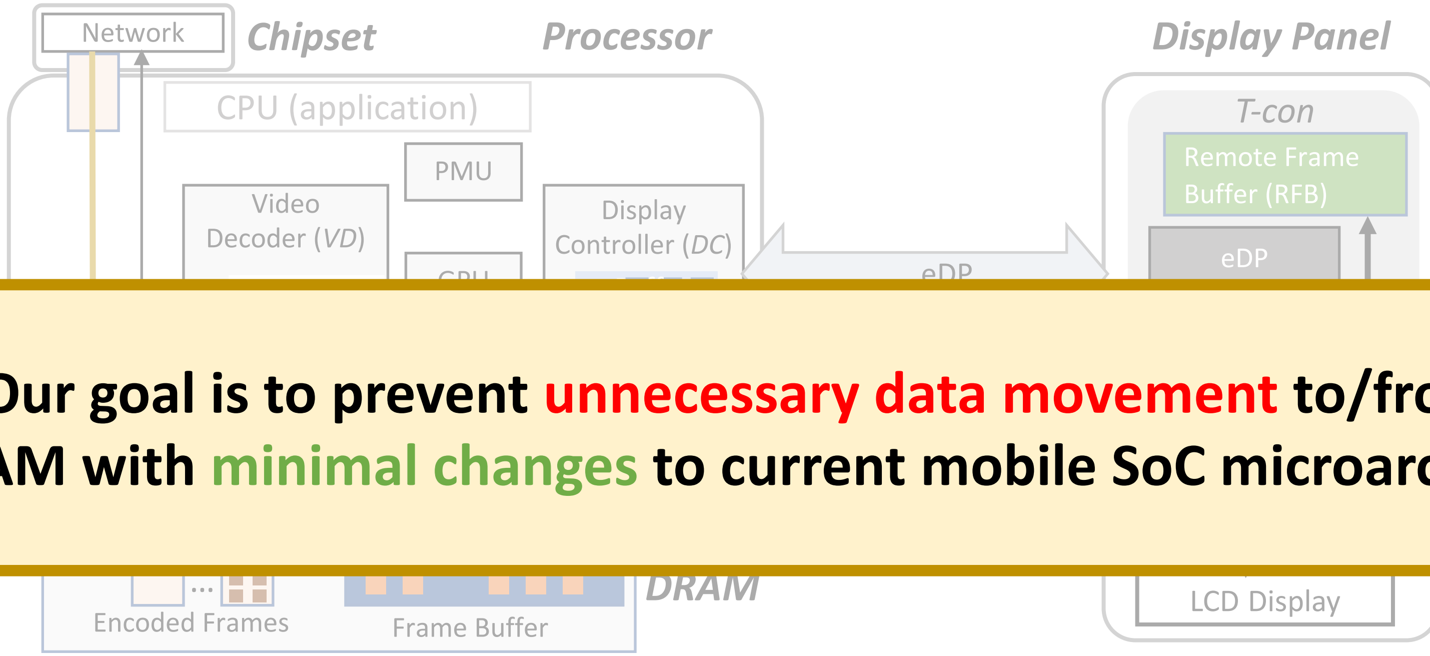
2. Underutilization of Display Interface (eDP) Bandwidth

1. Unnecessary Data Movement



- In current video processing schemes, the video decoder stores each decoded frame into the frame buffer in the host DRAM
  - This is necessary only when other planes exist in addition to the video plane (e.g., background, application-graphic plane, and cursor)
  - DC reads the data chunk from each plane's frame buffer, generates one composite chunk out of them, and sends the composite chunk to the display

1. Unnecessary Data Movement



Our goal is to prevent unnecessary data movement to/from host DRAM with minimal changes to current mobile SoC microarchitectures

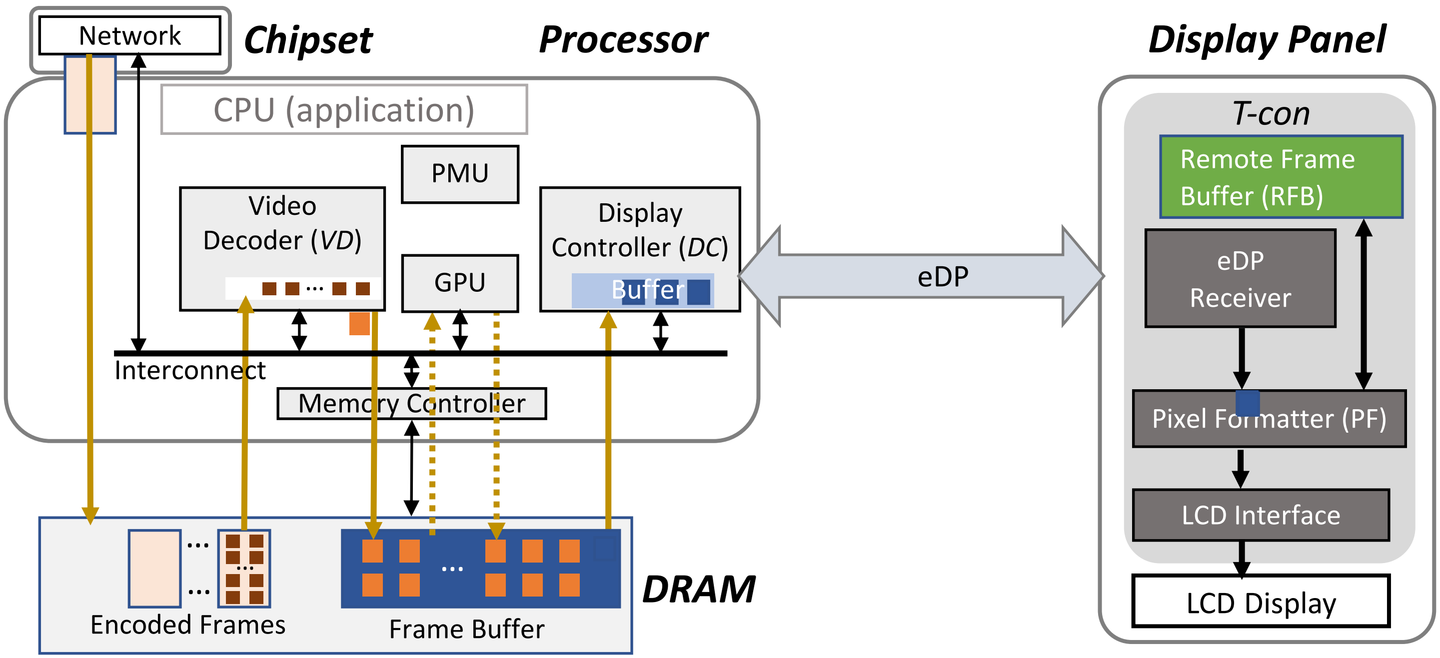
We prevent unnecessary data movement via "Frame Buffer Bypassing" (described later)

plane (e.g., background, application-graphic plane, and cursor)

- DC reads the data chunk from each plane's frame buffer, generates one composite chunk out of them, and sends the composite chunk to the display

SAFARI

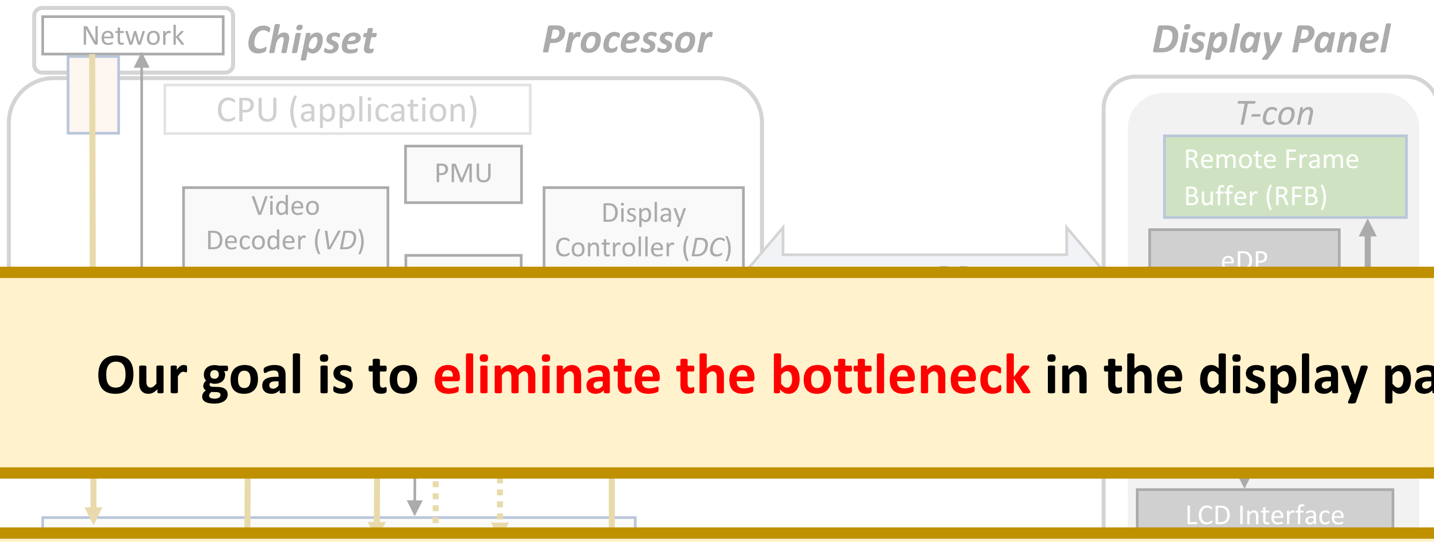
2. Underutilization of eDP Bandwidth



- The DC sends decoded frame data to the display panel in a constant rate during the entire frame window, keeping the DC and the eDP receiver continuously active
  - The transfer rates of the DC, eDP receiver, and pixel-formatter (PF) are tightly coupled and bottlenecked by the PF
  - The eDP interface bandwidth is underutilized during video streaming
    - For example, only half of the maximum bandwidth is utilized in 4K video streaming

SAFARI

2. Underutilization of eDP Bandwidth



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 to the display panel in a burst, thus increasing system idleness

- The DC sends decoded frame data to the display panel in a constant rate during

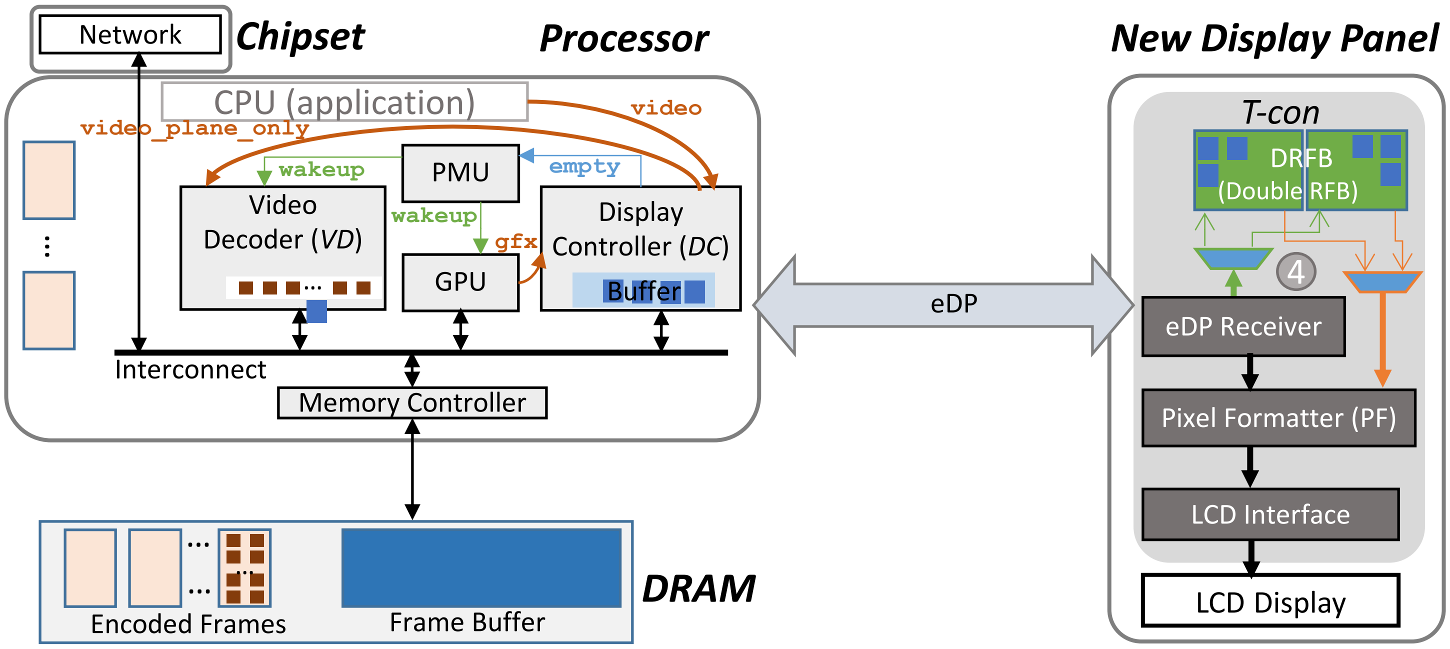
We eliminate the bottleneck in the display panel via "Frame Bursting" (described later)

- The eDP interface bandwidth is underutilized during video streaming
  - For example, only half of the maximum bandwidth is utilized in 4K video streaming

Presentation Outline

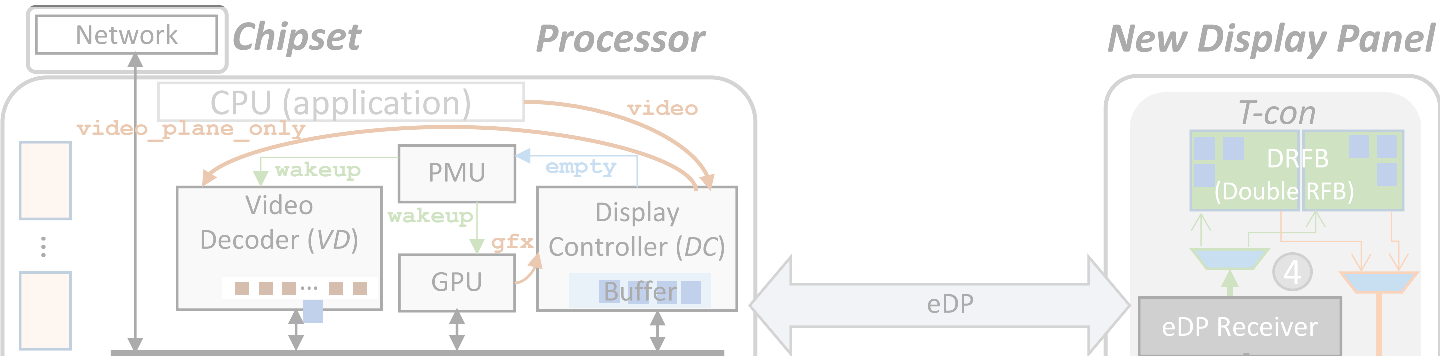
1. Overview of Mobile SoC Microarchitecture
2. Motivation and Goal
3. **BurstLink**
    I. Frame Buffer Bypassing
    II. Frame Bursting
4. Evaluation
5. Conclusion

1. Frame Buffer Bypassing



- The Frame Buffer Bypassing technique redirects the processed frame from the video decoder (VD) to the *display controller (DC)* via the on-chip interconnect if two conditions are satisfied:
  - The VD receives a signal (*video_plane_only*) from the DC indicating that only the video plane needs to be displayed (i.e., no need to merge the frame with any other plane frames)
  - The VD driver sets a flag (*single_video*) in the VD indicating that only a single video application is running (i.e., no need to merge the frame with any other video frames)

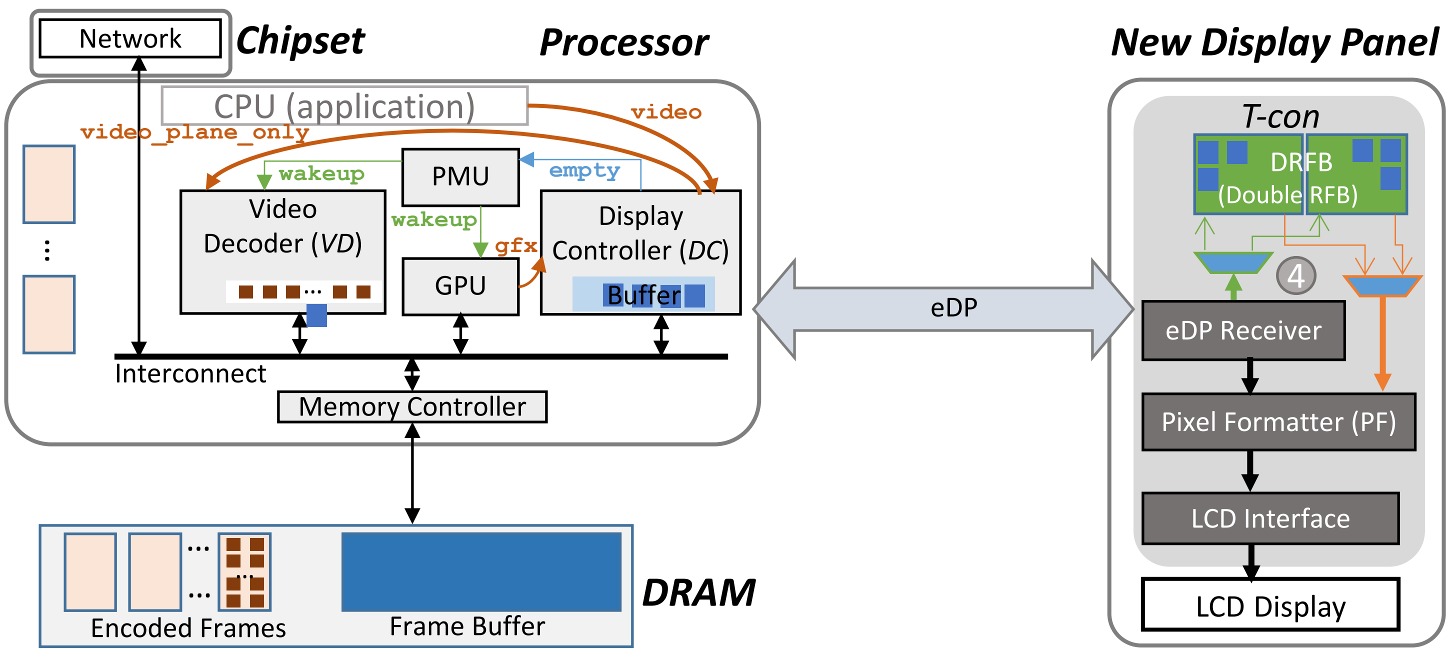
1. Frame Buffer Bypassing



Frame Buffer Bypassing reduces the energy consumption of the host DRAM by eliminating unnecessary data movement to/from the DRAM frame buffer

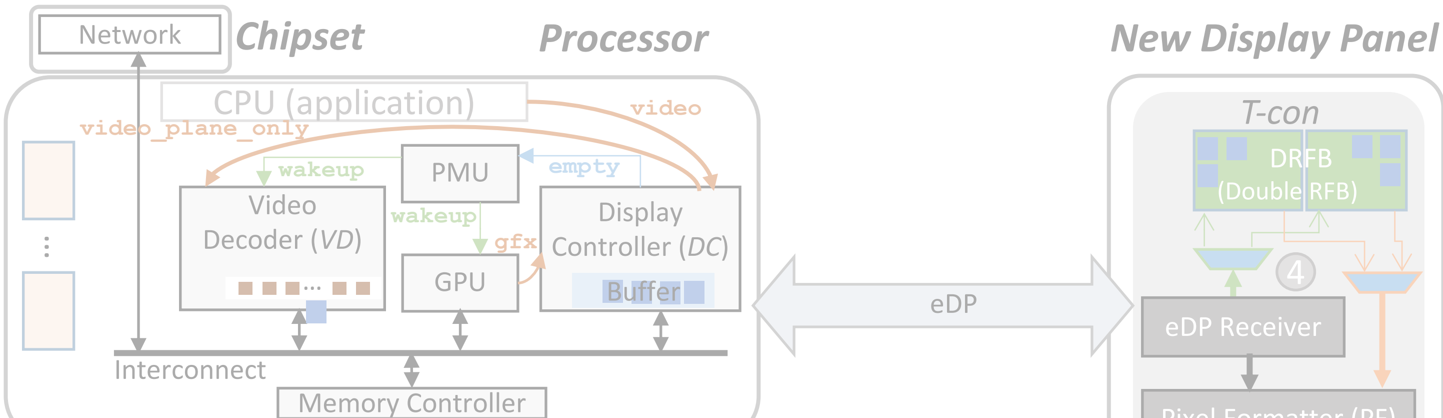
- The Frame Buffer Bypassing technique redirects the processed frame from the video decoder (VD) to the *display controller (DC)* via the on-chip interconnect if two conditions are satisfied:
  - The VD receives a signal (*video_plane_only*) from the DC indicating that only the video plane needs to be displayed (i.e., no need to merge the frame with any other plane frames)
  - The VD driver sets a flag (*single_video*) in the VD indicating that only a single video application is running (i.e., no need to merge the frame with any other video frames)

2. Frame Bursting



- The Frame Bursting technique transfers the decoded frame from the processor to the display panel in bursts
  - The display panel receives a full frame over the eDP interface and stores it directly into the double remote frame buffer (DRFB)
  - The Pixel Formatter (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 to the LCD display

2. Frame Bursting



The Frame Bursting technique reduces the utilization of the processor and the display subsystem

The system can enter deep low-power states between bursts for transferring the decoded frame from the DC to the remote frame buffer

- The display panel receives a full frame over the eDP interface and stores it directly into the double remote frame buffer (DRFB)
- The Pixel Formatter (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 to the LCD display

Other Details in the Paper

- System Power States in BurstLink
  - Details on the power state (i.e., package C-state) of a system that supports BurstLink

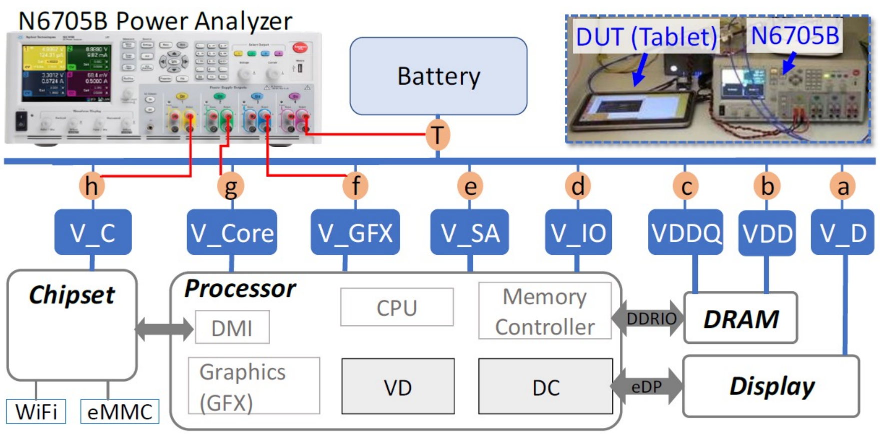
- Implementation and hardware cost of:
  - Double remote frame buffer (DRFB)
  - Destination Selector that selects the destination of the VD output
  - Changes to power management firmware

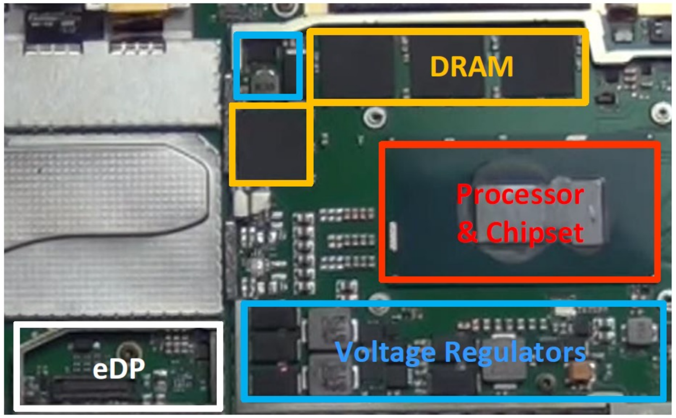
- Generalization of BurstLink techniques to other scenarios in modern mobile systems
  - Video capture (recording), audio streaming, video chat, social networking, and interactive games

Presentation Outline

1. Overview of Mobile SoC Microarchitecture
2. Motivation and Goal
3. BurstLink
   I. Frame Buffer Bypassing
   II. Frame Bursting
4. **Evaluation**
5. Conclusion

Methodology

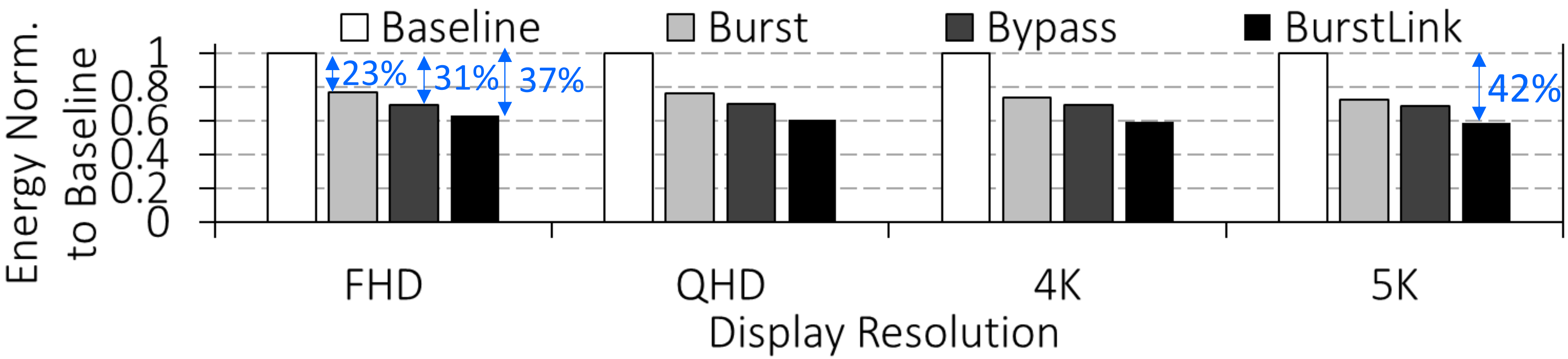




- **Framework:** we develop a new analytical power model
  - We validate our power model against power measurements from a real modern mobile device that is based on the Intel Skylake architecture
  - We use the Keysight N6705B power analyzer for system power measurements

- **Workloads**: planar and VR video-streaming workloads
  - Used in standard industrial benchmarks for battery-life and academic evaluations of video-streaming optimizations

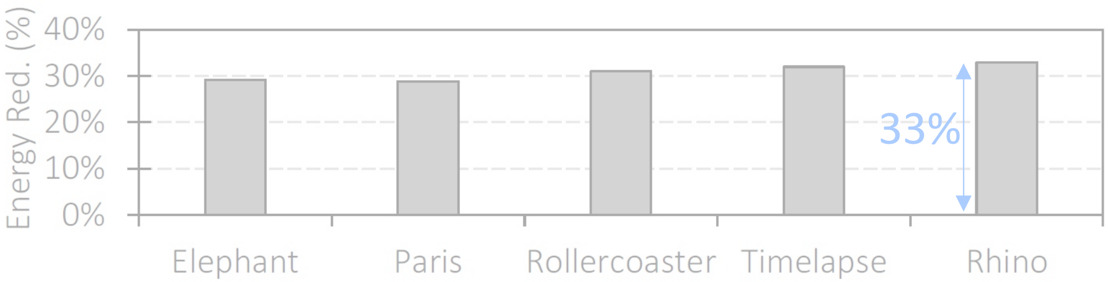
Evaluation - Planar Video Streaming

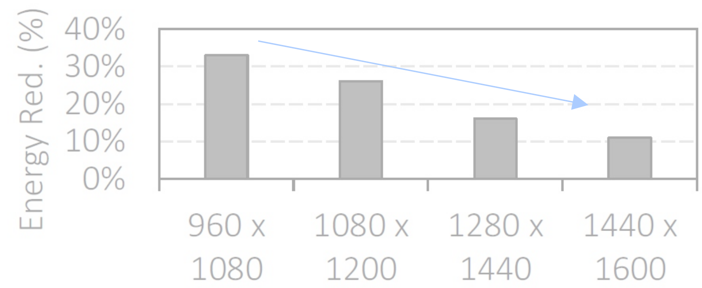


- BurstLink reduces the overall system energy consumption by 37% for an *FHD* (full high definition) display
  - Frame Buffer Bypassing and Frame Bursting reduce overall energy by 31% and 23% compared to the baseline, respectively

- BurstLink's energy reduction increases as display resolution increases
  - For a $5K$ display, BurstLink reduces the overall system energy by ~42%

Evaluation - VR Video Streaming





- BurstLink reduces the overall system energy consumption by up to 33%
  - Memory-energy dominant workloads have higher benefits compared to compute-energy dominant (mainly GPU) since BurstLink greatly reduces memory energy

- BurstLink's benefits decrease as VR display resolution increases
  - Compute energy becomes more dominant in VR workloads as display resolution increases
  - Higher compute energy decreases only the relative contribution of BurstLink's memory energy saving

Other Results in the Paper

- Effect of video frame rate on BurstLink benefits
  - BurstLink's energy consumption reduces as the video frame rate increases
- Comparison of BurstLink to existing techniques
  - 29% lower energy consumption than Frame Buffer Compression (FBC)
  - 35% lower energy consumption than Race-to-Sleep, Content Caching, and Display Caching techniques
- Benefits of BurstLink on other mobile workloads:
  - 40% lower energy consumption when playing local video files with different resolutions
  - Frame Buffer Bypassing reduces energy 12%-31% on four mobile workloads:
    - Video capturing, video conferencing, casual gaming, and MobileMark

Presentation Outline

1. Overview of Mobile SoC Microarchitecture
2. Motivation and Goal
3. BurstLink
   I. Frame Buffer Bypassing
   II. Frame Bursting
4. Evaluation
5. **Conclusion**

Conclusion

Problem: planar and virtual reality (VR) video streaming consumes significant system energy due to the high power consumption of major system components (e.g., DRAM, display interfaces, and display panel)

Goal: improve the energy efficiency of planar and VR video streaming by leveraging display panel local memory to eliminate buffering frames in main memory

Mechanism: BurstLink, a new planar and VR video streaming scheme that

- Directly transfers a full decoded video frame from the video-decoder (or GPU) to the display panel, completely bypassing the host DRAM
- Transfers a complete decoded frame to the display panel in a burst, exploiting the display interface's maximum bandwidth

Evaluation: we evaluate BurstLink using our open-sourced analytical power model that we rigorously validate on an Intel Skylake mobile system. BurstLink:

- Reduces system energy consumption for 4K planar/VR video streaming by 41%/33%
- Provides an even higher energy reduction in future planar video streaming systems with higher display resolutions and/or display refresh rates

https://github.com/CMU-SAFARI/BurstLink

BurstLink

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

Jawad Haj-Yahya *Jisung Park* *Rahul Bera* *Juan Gómez Luna*
Taha Shahroodi *Jeremie S. Kim* *Efraim Rotem* *Onur Mutlu*

SAFARI
SAFARI Research Group
safari.ethz.ch

ETH zürich

intel

TU Delft

Backup Slides

BurstLink: Key Results

- BurstLink reduces the energy consumption of the host DRAM by eliminating data movement to/from the DRAM frame buffer

- BurstLink 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

- We evaluate BurstLink using an analytical power model that we rigorously validate on an Intel Skylake mobile system. BurstLink:
  - Reduces system energy consumption for 4K planar/VR video streaming by 41%/33%
  - Provides an even higher energy reduction in future video streaming systems with higher display resolutions and/or display refresh rates

- We **show** that using main memory (DRAM) as a communication hub between system components is energy-inefficient
  - BurstLink uses small remote memory near the data consumer to significantly reduce the number of costly main memory accesses in frame-based applications