Driving futuristic resolutions with Display Stream Compression on Intel® Processor Graphics

MANASI NAVARE
Linux Graphics Kernel Developer
VPG, Intel
XDC 2018
What display resolution do you prefer? 

2K
2048 x 1080
~ 2.2 million Pixels

Present?

4K
3840 x 2160
~ 8 million Pixels

Past?

5K and higher
5120 x 2880
~ 15 million pixels and more…

What next?
Display Port link bandwidth
Is it keeping up with display evolution?

**BW Gbps**

- **2008**: RBR/HBR for DP 1.1, 2.7 Gbps/lane  2k @ 60
- **2009**: HBR2 for DP 1.2, 5.4 Gbps/lane  4K @ 60
- **2016**: HBR3 DP 1.3/1.4, 8.1 Gbps/lane

Can we do 5k@120?
What does it look like from an Engineer’s point of view?
5K is So many Pixels

14.7 Million pixels per 5K Frame

1769 Million pixels per Second

120 Frames per Second
5K is So much data

RGB 8bpc
8 bits 8 bits 8 bits
24 bits/pixel

1769 Million pixels/sec → 42.4 Gazillion bits/sec
Challenge in 5K Viewing—do we have the goods?

Required Display BW for 5K@120: 42.4 Gbits/sec
Available BW of DP @ HBR3: 32.4 Gbits/sec

What do we do with the remaining 10 Gazillion Bits?!
VESDA Display Stream Compression

- Industry wide standard for visually lossless compression
- Supported on eDP v1.4b, DP 1.4 and MIPI Display Serial Interface v1.2
- Supports RGB, YCbCr (4:2:0 and 4:2:2) video formats
- Compressed data rate is constant as opposed to JPEG
  - Video quality excellent with all content types
- Adopted by NVIDIA* Tegra X1* and Qualcomm* Snapdragon* 820 mobile SOCs for eDP/MIPI DSI
- Enabling on Intel® architectures starting Gen 11/ Icelake

https://www.vesa.org/vesa-standards/
5K@120 is too much data so compress it

Requested Display BW

Compressed Display BW

42.4 Gbps/sec

~14 Gbps/sec

Fits in DP HBR3 Link BW (32.4 Gbps)

Display Stream Compression 3x

24 bits/pixel

8 bits/pixel

5120 x 2880
## Encoding Process

### INDEPENDENTLY DECODED REGIONS - SLICES

<table>
<thead>
<tr>
<th>Slice</th>
<th>Slice</th>
<th>Slice</th>
<th>Slice</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

### GROUPS

<table>
<thead>
<tr>
<th>GRP1</th>
<th>GRP2</th>
<th>GRP3</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

### SETS

Consecutive pixels raster scan order
Encoded Bitstream

GRP# ➔ Entropy Based Variable Length Coding ➔ 000110001111 ➔ Substream Multiplexing ➔ Rate Buffer ➔ Constant Target Bitrate ➔ Encoded Bitstream Output

Entropy Based Variable Length Coding

https://www.vesa.org/vesa-standards/
Display Pipeline with Display Stream Compression

Intel Graphics Display Engine (Gen 11+)

5K Frame Buffer scanout

Pipe/CRTC

Atomic check - compute DSC params

Uncompressed Pixel Data

VESA DSC Encoder

Atomic commit - Send PPS infoframes to sink

Compressed Pixel Data

DP TX Port

Compressed Pixel Data Cable

DP Cable @HBR3 8.1 Gbps, 4 lanes

Encoding parameters

Compressed Pixel Data

VESA DSC Decoder

Uncompressed Pixel Data

5K Display

To Monitor DP RX Port

Memory Interface

Intel Graphics Display Engine (Gen 11+)
DSC Implementation across DRM and i915

△ DSC specification related helpers like all the compression parameter definitions, helpers for creating compression parameter PPS infoframes, helpers for parsing the DP DSC DPCD registers all in DRM subsystem

△ Hardware specific implementation for setting compression parameters on the source and computing PPS and RC parameters, enabling the HW pipeline all in i915

△ IGT tests for testing all bpc/compressed bpp combinations

https://patchwork.freedesktop.org/series/47514/
2 pipe challenge with DP DSC

- HW Limitation - VDSC engine operates at 1 Pixel/Clock throughput
- So if Pixel Clock > CDClock we need to use two VDSC engines
- Split across 2 pipes
  - Eg: Peak Pixel Rate for 5K@120 = 2672.75 MHz
Design Opens on ABI changes

△ Split 5K or higher resolutions across 2 pipes
△ Option 1: Kernel creates 2 CRTC states from a single modeset by X and configures 2 pipes
  △ Hide 2nd pipe from userspace
  △ Major refactoring in atomic
△ Option 2: Fake it as a tiled display with fake EDID to X, expose 2 modes
  △ Two separate modeset calls from userspace
  △ EDID challenges
Thank you!

intel-gfx@freedesktop.org
dri-devel@freedesktop.org

IRC: mdnavare@intel-gfx, dri-devel on freenode
Back-up Slides
Display Port Interfaces

- eDP Connection
- DP Cable
- Thunderbolt Cable 5k@60
- USB-C Cable 4k@60
Other display compressions

- End to End compression – happens before framebuffer scanout from memory to pipe/crtc
  - Ideal scenario non zero compression
  - Heavily content dependent so for some frames its 0 compression
  - Cannot rely on this to save memory bandwidth

- Framebuffer compression
  - After display fetches the pixel data on incoming display port
  - Compresses pixels, if need to reuse some parts it fetches from compressed buffer
Vesa DSC references

- https://www.vesa.org/vesa-standards/
- http://www.academia.edu/12895277/BDC-1_A_robust_algorithm_for_display_stream_compression