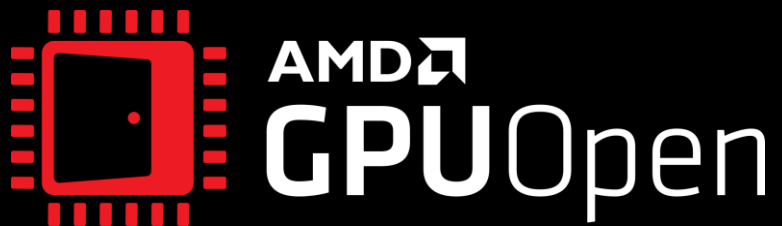




LET'S TALK ABOUT (GPU) CRASHES

ADAM SAWICKI

DEVELOPER TECHNOLOGY ENGINEER, AMD

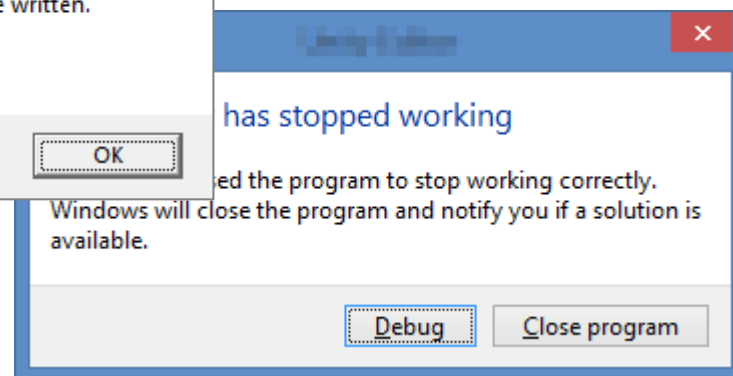
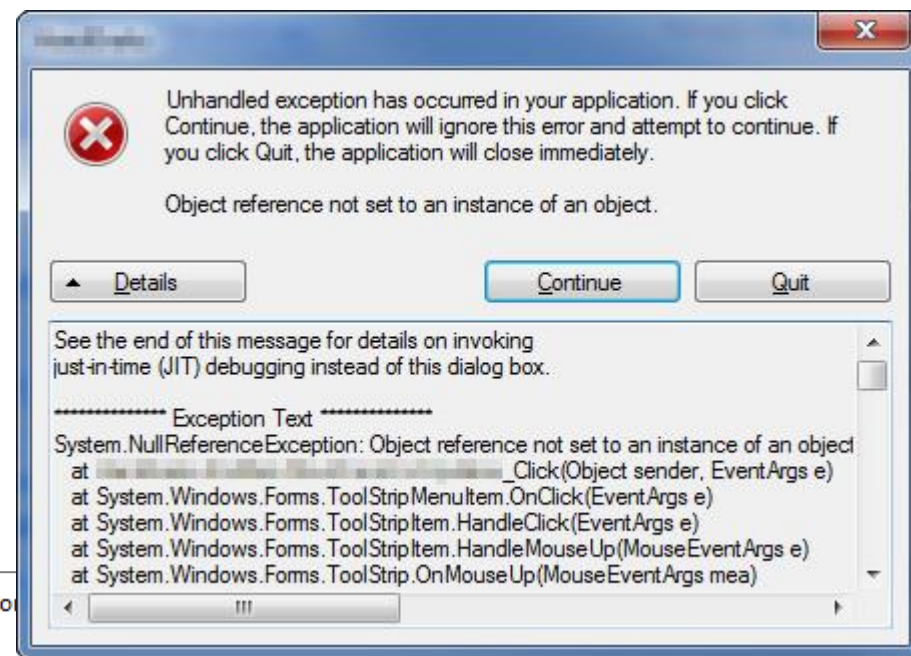
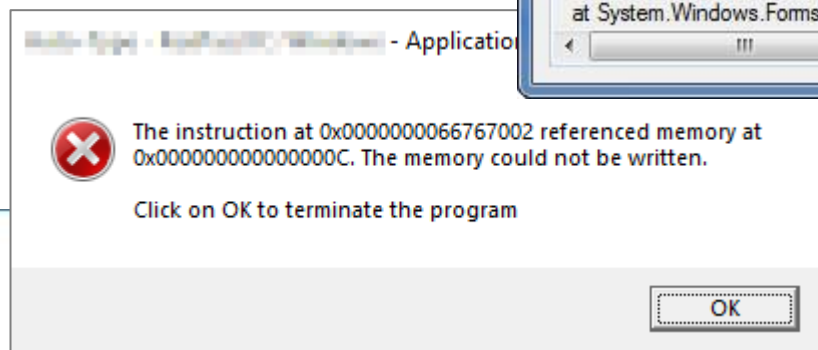
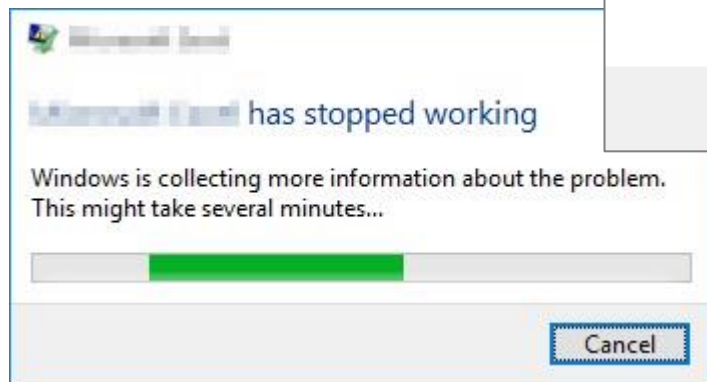


AGENDA

- Introduction
- GPU crashes
 - What is TDR?
 - What can cause TDR?
 - Why does it happen so often?
 - Effects
 - Why is it so difficult?
- Debugging
 - What can we do?
 - Breadcrumb markers
- Conclusions – general advice

INTRODUCTION

What is a crash?



CRASH

Program terminates abnormally

Thesis:

“Crashes are good” *

Crashes are one of the security measures employed by the OS.
Instead of a silent corruption, it lets you know you made a mistake.

* As long as they are easy to debug!

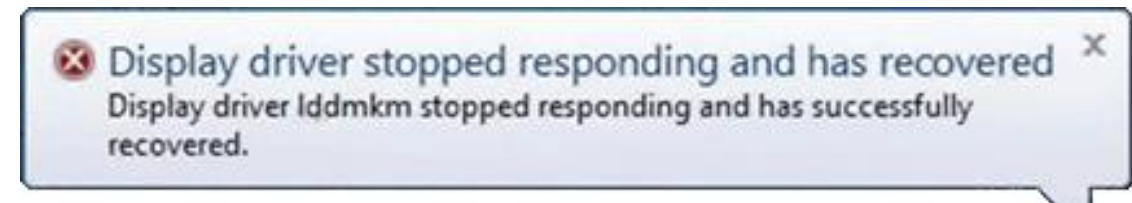
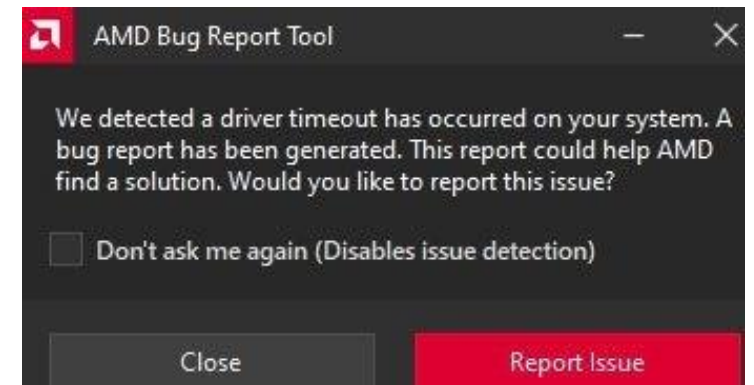
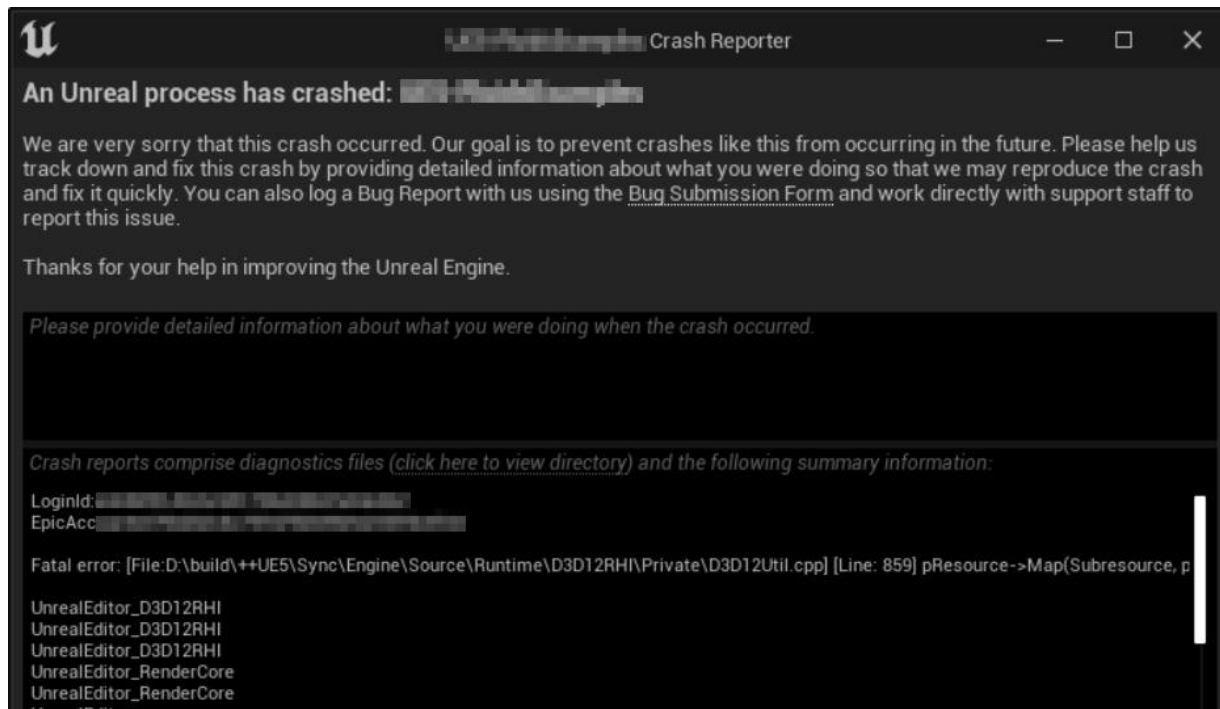
GPU CRASHES

What do I mean by that?



GPU CRASH

- We talk about graphics APIs here – mostly **DirectX® 12** | **Vulkan®**

- Timeout Detection and Recovery (TDR)



WHAT ARE POSSIBLE CAUSES?

-  Application bug – incorrect usage of the API ← most likely!
-  Driver bug
- External factors: driver update, hardware failure, ...

WHAT ARE POSSIBLE CAUSES?

- Infinite loop in a shader
- Memory page fault
 - Using a resource after `Release()` or `Evict()`
 - Indexing out of bounds
 - Incorrect address calculation
- Invalid/missing resource binding – null, wrong type, ...
- Corrupted data e.g., acceleration structure
- Other...



Application bug – incorrect usage of the API

WHY HAPPENS SO OFTEN?



Application bug – incorrect usage of the API

Old APIs (OpenGL, DX9, DX11):

- Driver is validating everything, each function returns error code
- GPU crash was likely a driver bug

New APIs (DX12, Vulkan):

- Driver is not validating, many functions return void
- Driver is simpler and faster 👍
- GPU crash is likely an application bug 😞

“Undefined behavior”

WHY HAPPENS SO OFTEN?

New APIs (DX12, Vulkan):

- Driver is not validating, many functions return void
 - Allocating functions like `CreateCommittedResource` return HRESULT
 - GPU commands like `DrawIndexedInstanced` return void
 - `Debug|validation` layers provide validation during development
- GPU crash is likely an application bug 😞
 - Driver bugs happen but shouldn't be your first thought

WHY HAPPENS SO OFTEN?



Application bug – incorrect usage of the API

Happens more often as we use raw memory addresses, dynamic indexing, bindless, indirect, ray tracing...



DX11:
ID3D11Buffer*

DX12:
D3D12_GPU_VIRTUAL_ADDRESS

Future:
void* ?? 🤪

“UNDEFINED BEHAVIOR”



Works fine

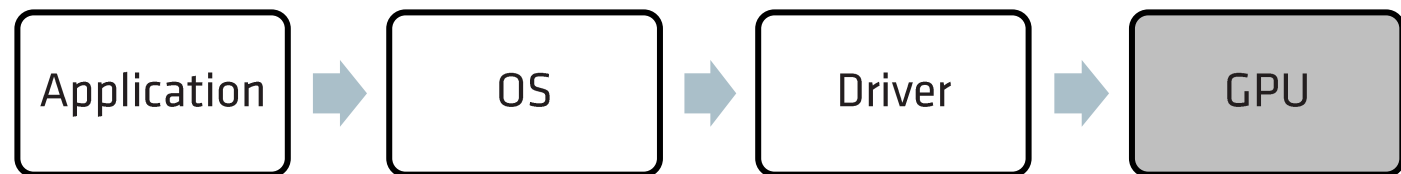
Visual corruption

Crash

What works on one GPU model may not work the same way on a different one

EFFECTS

- GPU and driver restarted
- Application observes an error code returned from API function
 - E.g., `IDXGISwapChain4::Present()` returns `DXGI_ERROR_DEVICE_HUNG`
 - E.g., `vkQueueSubmit` returns `VK_ERROR_DEVICE_LOST`
- Full machine hang or BSOD less frequent





Your device ran into a problem and needs to restart.
We're just collecting some error info, and then we'll
restart for you.

100% complete



For more information about this issue and possible fixes, visit
<https://www.windows.com/help>

If you still need support, please contact the
Windows Support team at <https://www.windows.com/help>

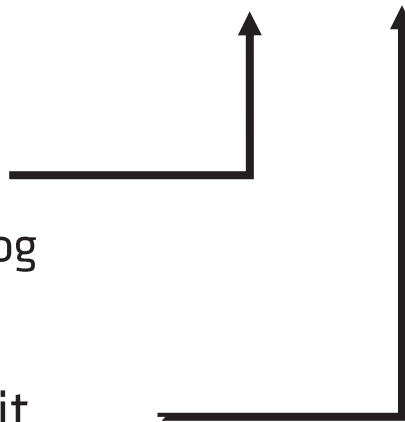
1

EFFECTS

Note that:

`IDXGISwapChain4::Present()` returns `DXGI_ERROR_DEVICE_HUNG`

- Doesn't imply our app crashing (in theory)
 - We can continue or at least save some dump/log
- Doesn't tell which pass or draw call is the culprit
 - Reported for the entire render frame



WHY IS IT SO DIFFICULT?

- “GPU Crash” can mean different things – timeout, page fault, ...
- GPUs are complex
 - Asynchronous – execute work submitted by the CPU
 - Pipelined – multiple commands in flight at various stages of the pipeline
 - Parallel – many threads, vertices, pixels processed at once
- Even if one hardware block fails, others may continue – no global STOP with break into a debugger

(Not an excuse for the lack of good debugging tools)

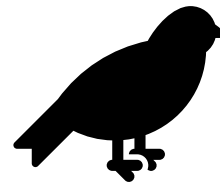
DEBUGGING

The solutions...

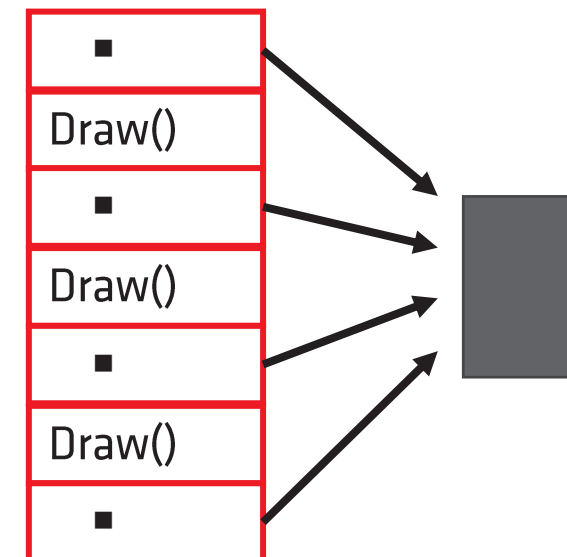
WHAT CAN WE DO?

- Capture with PIX or RenderDoc? No... They need a successfully rendered frame
 - Can still help with finding some issues
- **Debug|validation** layers
 - Validate correct API usage
 - Moderate performance overhead 🐢
 - Cannot validate what is not known on the CPU: GPU-generated data, descriptors, memory contents...
- **GPU-Based Validation (GBV) | GPU-Assisted Validation**
 - Extra validation on the GPU, shader instrumentation – descriptors etc...
 - Very high performance overhead 🐢🐢🐢

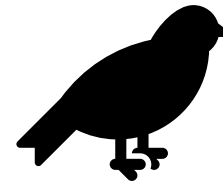
BREADCRUMB MARKERS



1. Startup: Create a buffer in the readback CPU memory, persistently mapped
 - `VirtualAlloc + OpenExistingHeapFromAddress + CreatePlacedResource`
 - `VK_AMD_device_coherent_memory - VK_MEMORY_PROPERTY_DEVICE_COHERENT_BIT_AMD / DEVICE_UNCACHED_BIT_AMD`
2. Rendering: Write numbers between passes or draw calls
 - `ID3D12GraphicsCommandList2::WriteBufferImmediate`
 - `VK_AMD_buffer_marker - vkCmdWriteBufferMarkerAMD`
3. After crash: Inspect the buffer pointer, see which breadcrumbs were successfully written last

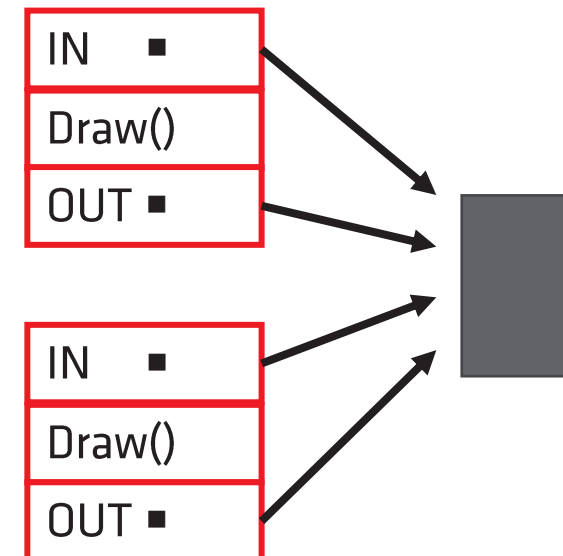


BREADCRUMB MARKERS



Not that simple...

- Multiple draw calls in flight simultaneously
- The IN-OUT semantics:
 - `D3D12_WRITEBUFFERIMMEDIATE_MODE_MARKER_IN/OUT`
 - `VkPipelineStageFlagBits pipelineStage`
- Still may not be reliable
 - No one said that after a crash the subsequent commands don't get executed



WHAT CAN WE DO?

contd.

- Breadcrumb markers
- **Device Removed Extended Data (DRED)** – new part of D3D12 doing such markers automatically
- Vendor-specific tools
- Last resort: disable individual effects and passes, see if the bug goes away
 - Ultra → High → Medium → Low
 - Disable ray tracing
 - Lower GPU memory usage: Texture quality = Low
 - Modify/simplify shaders

CONCLUSIONS – GENERAL ADVICE

Prevention is better than cure. Better to diagnose and treat earlier than later.

- Stability > Correctness > Performance
- Make the game repeatable – benchmark mode, saves, camera teleportation, reduce randomness
- Frequent testing on a range of GPUs (and update your drivers!)
- Use debug layer; make it free from any errors (the broken window theory)
- Ensure short iteration times – fast loading, command-line parameters, saves
- Make it easy to debug – toggles for various effects/passes/optimizations, intermediate texture preview
- Track regressions – if something used to work and stopped working, revert the culprit change

Ask for change! Better tools, better APIs, less crashes, better debugging experience. Talk to your ISV and IHV contacts.

THANK YOU!

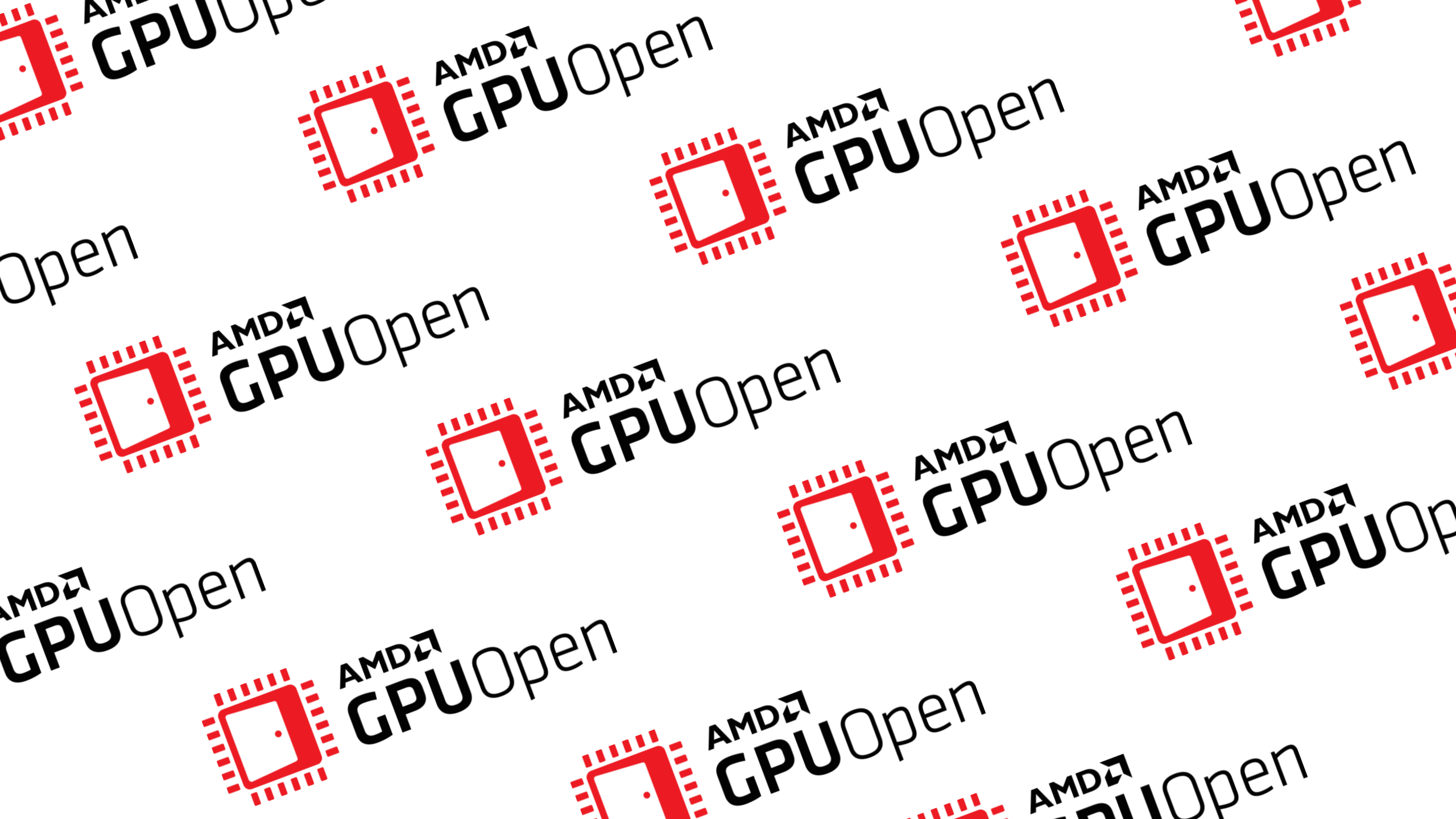
- Amit Ben-Moshe
- Jonas Gustavsson
- Luke Iwanski
- Marek Machliński
- Matthäus Chajdas
- Nicolas Thibieroz

DISCLAIMER & ATTRIBUTIONS

DISCLAIMER

The information contained herein is for informational purposes only and is subject to change without notice. While every precaution has been taken in the preparation of this document, it may contain technical inaccuracies, omissions and typographical errors, and AMD is under no obligation to update or otherwise correct this information. Advanced Micro Devices, Inc. makes no representations or warranties with respect to the accuracy or completeness of the contents of this document, and assumes no liability of any kind, including the implied warranties of noninfringement, merchantability or fitness for particular purposes, with respect to the operation or use of AMD hardware, software or other products described herein. No license, including implied or arising by estoppel, to any intellectual property rights is granted by this document. Terms and limitations applicable to the purchase or use of AMD products are as set forth in a signed agreement between the parties or in AMD's Standard Terms and Conditions of Sale. GD-18

© 2022 Advanced Micro Devices, Inc. All rights reserved. AMD, the AMD Arrow logo, Radeon, Ryzen, and combinations thereof are trademarks of Advanced Micro Devices, Inc. Vulkan and the Vulkan logo are registered trademarks of the Khronos Group Inc. PCIe and PCI Express are registered trademarks of the PCI-SIG Corporation. DirectX is a registered trademark of Microsoft Corporation in the US and other jurisdictions. Other product names used in this publication are for identification purposes only and may be trademarks of their respective owners.



AMD 