Advances in Computer, Signals and Systems (2023)
Clausius Scientific Press, Canada

DOI: 10.23977/acss.2023.070313
ISSN 2371-8838 Vol. 7 Num. 3

The Implementation and Feasibility Study of Supporting Dual Graphics Cards on Android Devices

Bing Zhao[1,*], Donghu Yang[2], Zihao Zheng[2]

[1]Feicheng Teacher Training School, Tai'an, Shandong, China
[2]System Architecture Department, BlackShark Technology, Shanghai, China
[*]Corresponding author

Keywords: Discrete graphics card, GMEM, GPU offloading, PCIe, Android

Abstract: The Mobile games have a more and more obvious trend towards high image quality. The demand for high resolution of AR/VR is also growing, and the demand for mobile GPU is also increasing. In order to solve this problem, this paper proposed a solution that supports discrete graphics cards on Android mobile devices similar to those on PCs. Discrete graphics memory does not preempt memory bandwidth with the system, thus freeing the increasing demand for GPU bandwidth on mobile phones. By modifying the display frame of Android, this method can run internal GPU and discrete graphics card at the same time, and can select different GPU according to different scenarios or applications, so that users can have a better game experience. The test results on the prototype show that the function is completely feasible, which can be directly connected to the display or Write-Backed to the mobile phone. The test results show that for mobile games, the size of video memory does not need to be particularly large, while the old GPU architecture, open source driver, PCIe latency and bandwidth have a great impact on performance. Compared with the write-back mode, the direct-connect mode is almost the only recommended method.

1. Introduction

Since 2020, we have found that the trend of mobile games not only toward high refresh rate, but also the trend of high image quality is becoming more and more obvious, For example, the newly released Protogod and Peace Elite HDR HD. In addition, the resolution of Protogod, Peace Elite and Wipeout 3 is still only 720p, and it is likely to develop to 2K resolution in the future. And AR/VR games themselves are even more demanding on the GPU. Predictably, mobile devices are also becoming more demanding for GPUs.

The discrete graphics card is configured with independent video memory, which does not occupy the memory bandwidth with the system. The video memory bandwidth is generally ten times that of the system memory, which is favorable for GPU parallel processing tasks. The performance advantage of the discrete graphics card is very obvious on the PC. For gaming laptop, discrete graphics cards are also basically standard. Therefore introducing discrete graphics (including discrete graphics) form factor to mobile is also a direction the industry is considering, and for gaming enthusiasts seeking the ultimate in performance, this certainly gives them a new option.

Laptop also need to take into account the power consumption, but the strategy is all different, the common solution has Nvidia Optimus technology, through the switching of dual graphics cards [1], only large games or programs will be switched to discrete graphics. This technology is also only available for mobile devices with higher power requirements. At present, there is no corresponding software implementation scheme on Android system, so our aim is to put forward such a solution, and give out the problems and feasibility analysis of the scheme. The method can realize the following effects: the independent display card can smoothly run on the Android mobile device, so that the high-load game runs more smoothly.

2. Related Studies

GPU has been developing continuously in the past 10 years. There are two centralized trends: one is a thin client, which is represented by rendering on the server and then transmitted to the client [2]; the other is a fat client, i.e. the mobile end is responsible for the main computing, which develops in parallel in both directions, but the demand for bandwidth is increasing [3]. Our main object of study is the latter. Qualcomm Snapdragon, Arm's Mali, Nintendo Switch's Tegra X1 are unified memory architecture, and Apple's latest Mac is also a unified memory architecture [4], The problem with this solution, which has become almost standard on mobile, is that the scarcity of bandwidth in such an architecture design can cause competition with devices including CPUs.

In order to solve the bandwidth problem, the research of mobile GPU also focuses on how to efficiently utilize bandwidth and reduce power consumption. For example, [5] proposed how to efficient render 3D games in real time, [6] is also a way to reduce power consumption by reducing bandwidth. In addition to this, there are also ways of configuring the hardware, for example by increasing the bit width, such as 8CX from 64 to 128 bits [7] (used by Qualcomm for the laptop market), such as using lpddr5 with a higher energy efficiency ratio. But this is not all; there are 2 other ways of reference.

a) The design power consumption of the game laptop PS is 360 W, and the memory uses high-speed GDDR, which also belongs to the unified memory architecture, but this design focuses on the game [8];

b) Except for PC and laptop, external graphics card + independent graphics card is adopted, which belongs to non-unified memory architecture, and the external graphics card has its own video memory.

Despite the scarcity of bandwidth, the use of discrete graphics cards in mobile phones has never been considered a formal direction, and there are no mass production solutions yet, mainly due to power consumption, cost and area. However, the discrete graphics card is not just no use scenario. For example, the dock is displayed on the TV. For example, the pressure of higher resolution in the future. In order to support the switching between the standalone graphics card and dual graphics card, our reference scheme is as follows:

Android platform supporting discrete graphics card: Android X86 platform is designed by Beyounn and Cwhuang [9]. The main purpose of the project is to provide a complete Android system solution for the X86 platform, mainly for PC, which can run intel, amd, neaveau graphics cards, and will not support both, such a configuration is not very suitable for mobile phones. Most mobile phone platforms are still used by Qualcomm and mail. If it supports common mobile phone functions and high energy efficiency, it is almost difficult to avoid. Therefore, it is necessary to switch between dual-display cards on mobile phones.

Optimus technology supporting dual graphics switching [1]: The concept of dual graphics switching generally refers to nVIDIA Optimus intelligent dual graphics switching technology promoted by Nvidia [1]. With Optimus technology, when the standalone GPU is responsible for all rendering tasks, the final image output on the display will still be done by the Intel Integrated Graphics Processor (IGP), the equivalent of the mobile DPU mentioned here.n practice, and the IGP is used as a simple display controller, enabling a seamless, flicker-free experience without the need for a restart. When running less demanding applications, the discrete GPU is switched off and the Intel IGP takes care of rendering and display calls to save power and maximise battery life, with Optimus' technology primarily used on window systems.

Dual graphics card switching on Linux system: Linux system open source code support should be implemented in mesa [10], the console can be switched to discrete graphics rendering through commands such as "DRI_PRIME=1 glmark2-wayland". The support for Wayland (more similar to that of Android) and XWindow is not complete, and even the problem of screen tearing exists. However, linear buffer and tiling buffer have a lot of inspirations for later developers, and the latest branch already supports this strategy. But such a strategy is not realistic for Android, as the built-in GPU that OEMs get is a closed-source UMD driver, not a standard mesa driver.

In fact, there are two forms to support the discrete graphics card at the mobile end. However, the above solutions do not discuss the difference in software implementation caused by the difference. However, we shall carefully discuss the specific implementation schemes of the two forms and give suggestions on which form is recommended to be feasible.

Common direct connection mode on PC: as shown in the left of Figure 1, select different GPU and DPU according to different scene modes (the graphics card includes GPU and DPU functions, and DPU is the display engine), such as game mode. All interfaces will be drawn by GPU in discrete graphics card and output to LCD or TV directly by DPU (non-internal DPU) in discrete graphics card;

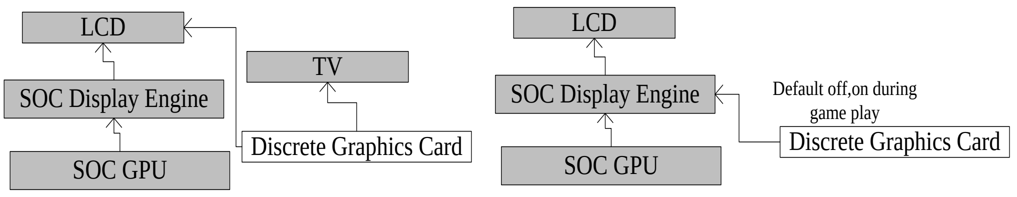

Figure 1: Direct mode (left) and Write-Back mode (right)

Common write-back mode on laptop computer: as shown in the right of Figure 1, select different GPUs according to different applications. For large games, select GPU with discrete graphics card. After drawing, it will be displayed back to LCD.

3. Implementation Method

The main work of the discrete graphics card we usually know is drawing (client) + sending (server), where buffer sharing connects these two functions. So there are three key modules. The main aim of this article is to support discrete graphics in Android, so it is necessary to understand the 3 key modules that correspond to the Android display system.

a) GPU Renderer: OpenGLES / Vulkan

b) GRALLOC: distribution mode of ION

c) HWC: composite display, dealing with matters related to DPU

In order to realize the stand-alone video card running on Android system, the main work of this article is to adjust the 3 main modules + adjust the data flow. We will give a detailed description of 2.1 direct mode and 2.2 write-back mode.

3.1. Direct Mode

First of all, we need to understand that the discrete graphics card has its own video memory. In Linux system, the buffer management method is GBM instead of ION, which is different from Android system. Another difference is that the GPU library is usually open-source mesa rather than closed-source vendor library on android. If it is in direct-connected mode (as shown in Figure 1), this operation of the hwcomposer is also different from the native hwc used on android systems.

Then, on the premise of not changing the whole large display frame, we can realize the function of simultaneously running the dual GPUs by adding parallel modules and switching logic. The added parallel modules are:

GPU rendering: mesa3d (support open source amd, nvida, intel graphics cards)

GRALLOC_ex: Buffer allocation uses GBM allocation mode of discrete graphics card;

HWC: drm_hwcomposer module is added to process the output of discrete graphics card, but write-back mode is not required

The software framework is shown in Figure 2 (orange module is the original module of the system, and blue is the new module).

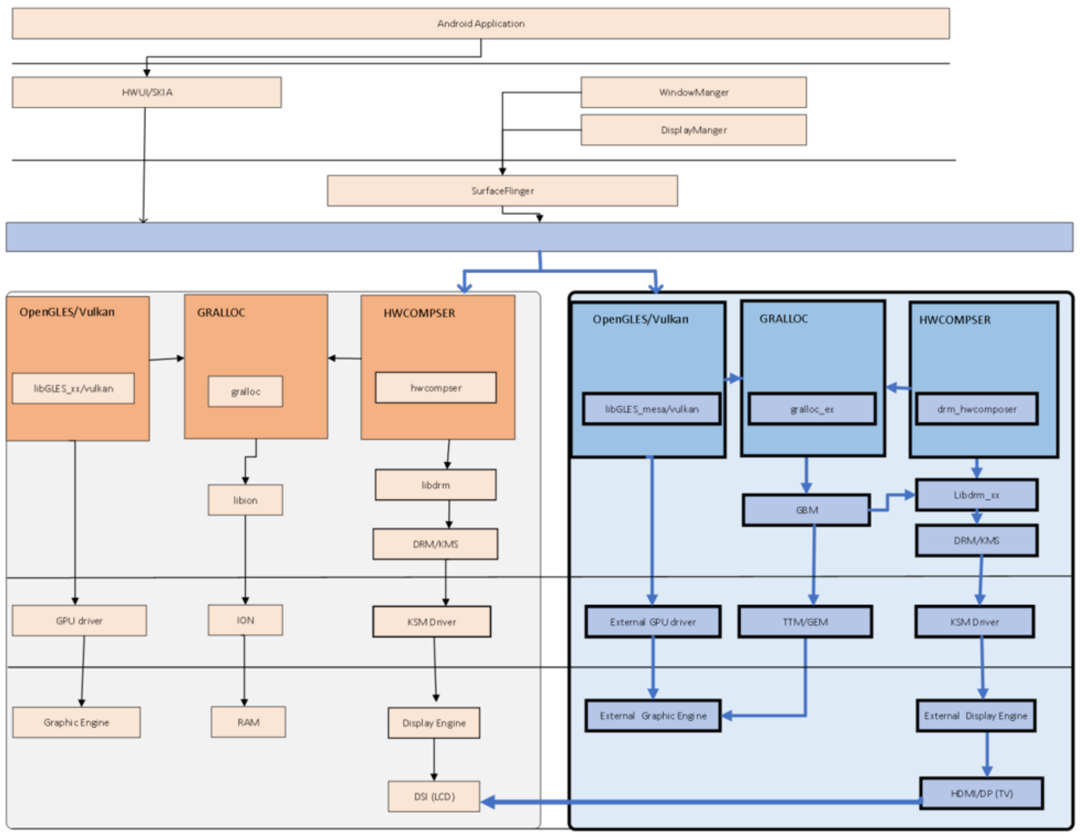

Figure 2: Display frame running dual GPU on Android system (directly connected)

The switching logic is that GPU and DPU switch at the same time (equivalent that the other GPU is completely off, so the data flow diagram is unchanged).

a) When normal mode is started, Buffer will also call the default libGLES_xx1 module when drawing according to the allocation of the default Gralloc module, and the default hwcomposer module will be selected when sending the display.

b) When the game mode is started, Buffer allocation will select the allocation mode of the discrete video card according to the Gralloc_ex module. During drawing, the libGLES_mesa3d module will be called. When the display is sent, the display will be sent in the drm_hwcomposer module. The final display is output directly to the TV via the DP port or to the LCD via a hardware converter interface. It can be directly output to the display through the DP interface, so as to be applied to the projection screen, which is also an important scene.

3.2. Write-Back Mode

The write-back mode can only be output through the built-in DPU, so the Buffer needs to be delivered back through PCIe. This efficiency loss is large. We will give some optimisation options in the next section and later we will give a comparison. The software framework is shown in Figure 3 (orange module is the original module of the system, and blue is the new module).

Switching logic-only the GPU is switched (external drm_hwcomper module is not being used, 2 GPUs are used simultaneously):

a) After the normal application is started, Buffer will call the default libGLES_xx1 when drawing according to the allocation mode of the default Gralloc module. The default hwcomposer module will be selected when sending the display.

b) When the target application is started, Buffer allocation will be based on the Gralloc_ex module to choose the allocation method for the standalone graphics card, and the libGLES_mesa module will be called when drawing, and then passed back to the internal display engine for compositing, and the display will be sent according to the default hwcomposer module when sending the display.

c) The hwc module can not only know the memory tiled buffer unique to the discrete graphics card, and convert it into a linear buffer that can be accessed by the built-in display engine in the hwc. At this time, the hwc is associated with the granoc_ex. This conversion can also be completed by mesa. At this point the hwc module gets the converted linear buffer, and hwc does not need any connection with the external gralloc_ex, so the blue dotted line above is optional.

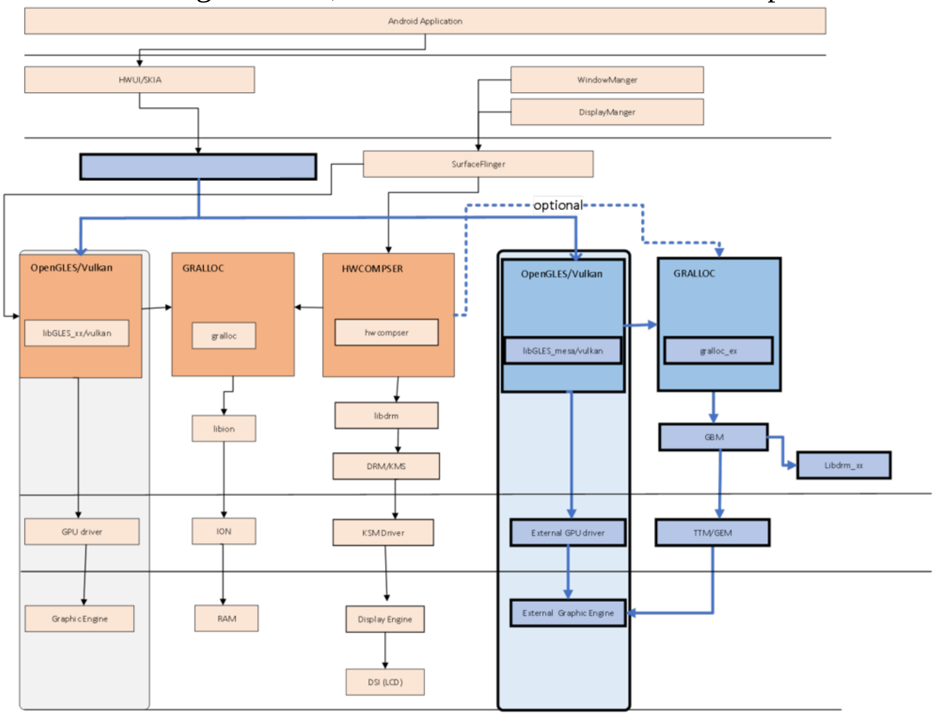

Figure 3: Display frame running dual GPU on Android system (Write-Back)

4. Storage Management

Since the VideoRAM of the standalone video card cannot be accessed by the SOC, a memory copy process is required, as shown in Figure 4 below.

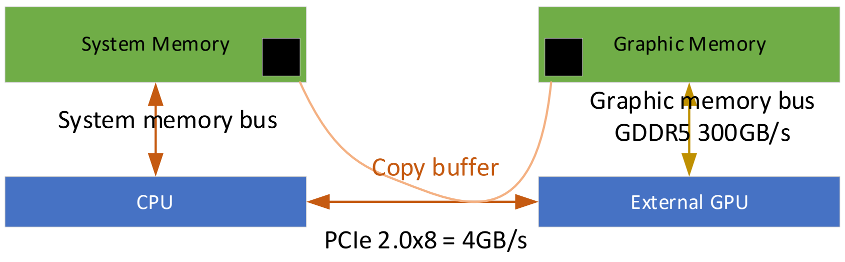

Figure 4: Buffer transfer mechanism of Write-Back mode

In Linux system, the dual GPU of X11 desktop is Write-Back mode, and its processing mode is simpler: linear buffer is used when distributing buffer, which is equivalent to VRAM without using discrete graphics card at all. Although it does not need to be copied, the efficiency is too low for large games.

In Linux, the Wayland desktop allocates VRAM, i.e. tiled buffer, when drawing in mesa. After drawing, the tiled buffer of each layer is converted to a linear buffer, which is better for larger games (compared to X11). In addition, because each buffer is processed in the Mesa of the Client side, the Server side is completely unaware. But there are two problems:

Firstly, processing at the Client side will wait for the current layer to be finished (glFinish), thus causing performance degradation;

Secondly, the simultaneous transmission efficiency of multiple Layers doubles the bandwidth pressure of PCIe (The bandwidth of PCIe devices on mobile devices is still relatively small, so minimising Buffer transfers between PCIe devices is also a priority for optimisation).

Based on these two points, we have the following two approaches to the target application.

Method 1: The target application allocates a tiled buffer (layer2, layerN as the target application layer, layer1 as the system layer, the same below); when converting the buffer, it is put in the server (hwc) for processing. In this way, it is not necessary to wait all the time after the drawing command is finished, while the waiting time is delayed to the composition, which improves the overall performance. At this time, the PCIe copy data amount=Layer2 loading +...+ layerN loading. Method 2 is proposed to further optimize this.

Method 2: The target application only has a single Layer (which is usually the case for games), and according to Method 1, the PCIe copy data size=1920 x 1080 x 4 (rgba) x 60 (fps) = 497 MB/s. The target application has multiple Layers (some games also exist) to ensure the performance and reduce the bandwidth of PCI-e:

The tiled buffer is used for all of them, and during the synthesis phase of the Server it is synthesised into a target such as a framebuffer, which can be either a linear buffer or a tiled buffer (Figure 6, left), or it can be converted to a linear buffer when the display is sent (Figure 5 right).

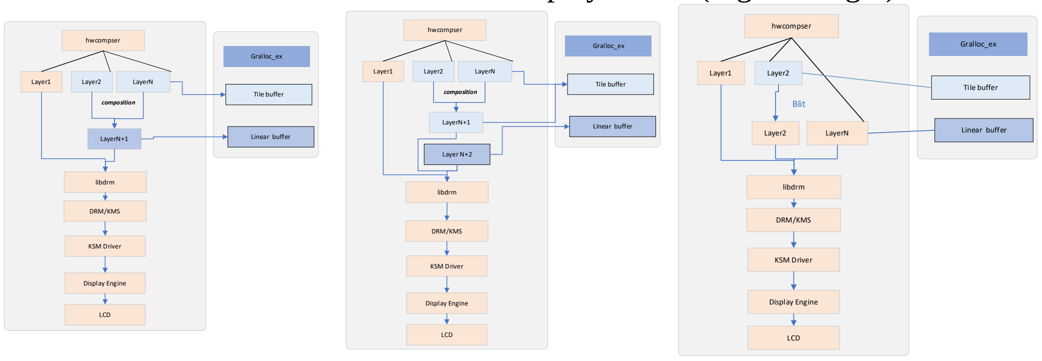

Figure 5: Tiled buffer and linear buffer

Two strategies combined in one is to use a tiled buffer for a complex Layer 2 that requires a lot of

GPU computation, and a linear buffer (no copy required) for a simple LayerN, which is usually used to display Icon only.

5. Test Results

Test platform: Qualcomm 855 (support PCIe3.0 x2) + Nvidia GT730 2G D3, which is an entry-level graphics card. The design power consumption is about 25 W. The graphics card can only support PCIe2.0. Where PCI express 2.0 x 2: 2 x 512 Mhz=1 GB/s

a) Impact of PCIe: The transmission delay caused by PCIe is as shown in the following figure (lock represents transmission from CPU to GPU, and copy yuvbuffer represents copy of CPU). It can be seen that the buffer of qcom is shared between CPU and GPU, so there is no need to transmit, less than 1 ms, while the transmission of nouveau requires 22 ms (yuv buffer of 1080P). This delay is too large, as shown in Table 1.

b) Bandwidth: The frequent interaction between CPU and GPU will result in tight PCIe bandwidth, and only 1GB/s bandwidth is very reluctant to play games. A single read of 2560x1440x60 fps x 4(RGBA)=842M/s takes up a huge amount of bandwidth.

As shown in the following figure, the single-layer uses the tiled buffer buffer and directly outputs it to the screen, and copies this to the framebuffer (the same is the tiled buffer buffer) to be displayed again. That is, tiled buffer-gpu synthesis has little effect on the frame rate, but it still has a certain impact. At this time, the bandwidth is not a problem, but only a performance problem. However, once linear buffer is used, the impact on performance is fatal, which is down by more than 80%. At this time, we rely heavily on PCIe, which we should avoid, so it is necessary to use tiled buffer. If the LCD is passed back, a full frame copy is required, which continues to increase the bandwidth requirements on top of the original (already insufficient PCIe). A direct performance drop of 60% can also be seen in the data, shown in Table 2.

Table 1: PCIe Delay Test

	Function execution time (ms)				
size	qcom lock	nouveau lock	copy yuvbuffer	nouveau unlock	qcom unlock
1920x1080	0.021	22.6	4.3	0.155	0.057
4096x2160	0.031	71.9	24.2	0.501	0.124

Table 2: Comparison of tiled buffer and linear buffer

		Direct TV: 2560x1440	Write-Back LCD: 3120x1440
Android comes with Angle bench marker	tiled buffer	26.24 fps	11.08 fps
	tiled buffer + copy	25.39 fps	10.63 fps
	Linear buffer + copy	3.94 fps	6.45 fps

Table 3: Performance Test

Test Application	Compatibility	View	Direct connection	WB 1	WB 2	GMEM
Peace Elite	OK	1	40 fps	20 fps	17 fps	No impact
King's Glory	OK	1	38 fps	17 fps	16 fps	No impact
Collapse 3	OK	1	38 fps	17 fps	16 fps	No impact
Touch ball	OK	1	60 fps	20 fps	17 fps	No impact
launcher	OK	2	30 fps	9 fps	15 fps	No impact

We have done separate tests on multiple videos from performance and compatibility. Despite the fact that we know that power consumption and heat are very important, here the GT730 is still independently powered and independently cooled, so the impact on this machine is very small and will not be discussed here. From Table 3, we can draw the following conclusions:

With few compatibility issues. Of course, the games we are investigating are still few. According to the statistical data of Huawei Kunpeng Cloud server, 15% of the Android games running on video cards may have compatibility problems; b) Because we use the nouveau open source driver, the performance loss is still relatively large. The power consumption of the discrete graphics card is 6 W (design power consumption is 25 W), which is still too high for the mobile platform; c) For mobile games, the old PC GPU architecture cannot exert the maximum performance; d) the mobile game occupies a relatively small bandwidth, and GMem is about 512 M, but it is not necessarily a reasonable configuration for larger games in the future; e) PCIe bandwidth and delay are both very limited, and PCIe 3.0x2 cannot meet the game requirements; f) The write-back strategy is less costly, but the performance loss is significant, which is consistent with the paradox reflected in laptops, where the power requirements are higher, so write-back is not a good recommendation; g) The direct connection strategy is the preferred option, and all buffers should use tiled buffer. And the single layer and multiple layer policies should also be different.

6. Conclusion

This paper presents a solution to support discrete graphics on Android mobile devices, in the hope of freeing up the increasing demand for GPUs on mobile phones. In our solution, not only the direct connection mode but also the Write-Back mode are supported. The Android display framework needs to be greatly modified, in particular, the addition of third party GPU drivers, support for GBM's Gralloc module, and the addition or modification of HWC modules to support card1 devices.

The results of our tests on the prototype prove that the functionality is fully functional, both directly to the display and back to the phone. And different GPU libraries can be selected depending on the application, without compatibility issues (Fewer test applications yet). However, there are some problems that cannot achieve mass production. Here we also give our own suggestions:

PCIe bandwidth is a very big limitation. PCIe 3.0x2 cannot meet the game requirements and needs to be upgraded to PCIe4.0 x2 at least; discrete graphics cards need to be miniaturised, otherwise the area is unusable on a mobile phone, plus the power consumption is unbearable; GPU architecture and driver should be optimized based on Opengles and Vulkan. Otherwise, even if the performance is powerful, the efficiency of open source driver is not high. For current mobile games, memory size is not a key factor; the direct connection strategy is still the preferred option.

References

[1] Zi Lei. From manual to automatic evolution of NVIDIA Optimus intelligent graphics switching technology full analysis. Microcomputer, 2010 (9):

[2] Cuervo E, Wolman A, Cox L P, et al. Demo: Kaha-wai: high-quality mobile gaming using GPU offload. 2014.

[3] Hoyong Jin, Donghun Jeong, Taewon Park, et al. Multi-Pre-diction Compression: An Efficient and Scalable Memory Compression Framework for GPGPU. IEEE Computer Archit-ecture Letters, 2022, 21 (2): 37-40.

[4] Kenyon C, Capano C. Apple Silicon Performance in Scientific Computing. IEEE, 2022.

[5] Bamashetti S., Govil H., Akkaraju P.K., et al. Efficient Use of DDR Bandwidth and Space for Rendering 3D Graphics. Lecture Notes in Electrical Engineering, 2021: 585-598.

[6] Farazmand N, Kaeli D R. Quality of Service-Aware Dynamic Voltage and Frequency Scaling for Mobile 3D Graph-ics Application. IEEE International Conference on Computer Design. IEEE, 2017: 513-516.

[7] Z-R A. Sophos Providing 'Next-Gen Always-Connected 5g Pc Cybersecurity With Intercept X For Qualcomm Snapdragon Compute Platforms'. Commswire magazine: Incisive, intormed, independent, objective, 2021 (Feb.24).

[8] Halperin X. Marvel's Spider-Man: Miles Morales Procedural Tools for PlayStation 5 Content Authoring. 2021.

[9] Rana R, Kumar A. ANDROID-X86–Future for Personal Computers. 2017.

[10] Yamato Y. Study and evaluation of automatic GPU offloading method from various language applications. International Journal of Parallel, Emergent and Distributed Systems, 2022 (1/2): 37.