WHITE PAPER

Jonathan Bergsagel
Software Architect
OMAP Platform Business Unit
Texas Instruments

Loic Leconte
Platform Architect
OMAP Platform Business Unit
Texas Instruments

TEXAS INSTRUMENTS

Super high resolution displays empowered by the OMAP4470 mobile processor

WUXGA resolution tablets now becoming a reality for the Android ecosystem

Introduction

Mobile devices are quickly becoming our entertainment and professional computing hubs. Consumers expect to use smartphones, tablets and ultrathin laptops to play games, view streaming HD movies, and share their latest business presentations on the road or via video teleconferencing—all without latency or visual experience degradation. Mobile devices must, therefore, push the envelope on display specifications toward larger screen sizes and higher resolutions.

Operating systems (OSs)—such as Android 4.0 or "Ice Cream Sandwich"—are also introducing more complex 3D-enabled UIs, increasing the overall demand on the GPU. To effectively leverage a mobile processor to meet larger display demands without sacrificing smooth UI experiences, it's important to understand the challenges involved in the composition process of putting together various graphics and video surfaces into the final rendered output for such high resolution displays.

This paper focuses on the benefits of the OMAP4470 mobile processor's distributed composition architecture to handle various composition scenarios required by mobile OSs, in particular Android 4.0.

The OMAP4470 mobile processor's balanced system architecture handles various composition scenarios required by mobile operating systems, in particular the latest version of the Android OS—Android 4.0. Specifically, this paper will introduce the architecture's benefits through the use of a distributed composition approach uniquely enabled by the OMAP™ platform. Further, this paper will outline how those benefits enable the OS to take advantage of the full resolution of a WUXGA display at 1920 x 1200 pixels while providing the ultimate user experience by matching the composition rate with the refresh rate of the display at 60 frames per second (fps).

For a high-performance user interface (UI) that provides a smooth user experience under active UI transitions, it is important to keep the composition rate very close to the refresh rate of the display no matter how many surfaces are involved in the composition. This helps to ensure that there will not be any noticeable lag in the UI responsiveness under user interaction.

The smart-multicore OMAP4470 system architecture includes three separate acceleration features for composition and a large system memory bandwidth enabled by its dual-channel 32-bit LPDDR2 SDRAM interface. Together, these architecture elements help enable the high composition rate required for the smooth user experience expected on a high resolution display such as a WUXGA LCD. The OMAP4470 processor's three acceleration features used for graphics content generation and display composition are:

- An upgraded PowerVR™ SGX544 graphics processing unit (GPU), which is 1.4x faster in its triangles/sec rate and 2x faster in shader performance than the SGX540 in the OMAP4460 processor.
- A high-performance composition and graphics processing unit (CGPU), which is used to optimize and offload composition work from the GPU to save power, thereby allowing the GPU to be used in a more focused manner for content generation. The CGPU can accomplish composition tasks in about half the time of the GPU, resulting in significant power savings and improved composition performance for complex scenarios.
- A Display Subsystem (DSS) with 4 hardware display pipelines (also called "hardware overlays") that are also used to further offload composition tasks by compositing graphics and video surfaces directly to the display in a single pass.

Many mobile OS composition engines today, including the one used by Android, can offload most or even all of the composition effort from the main CPU using various hardware acceleration options that are available on the applications processor. Usually, the first choice is to use the embedded 3D graphics processing unit (GPU) via calls to the OpenGL ES 2D/3D graphics API. However, other acceleration options are often supported as well, such as available display hardware overlays and any available graphics accelerators. The composition engine in Android 4.0 has been designed to support composition either solely on the available GPU or in combination with other composition acceleration options that may be available.

There are three main approaches that can be considered for composition in Android 4.0 for the large majority of composition scenarios. The table below lists these approaches along with each of their strengths and weaknesses:

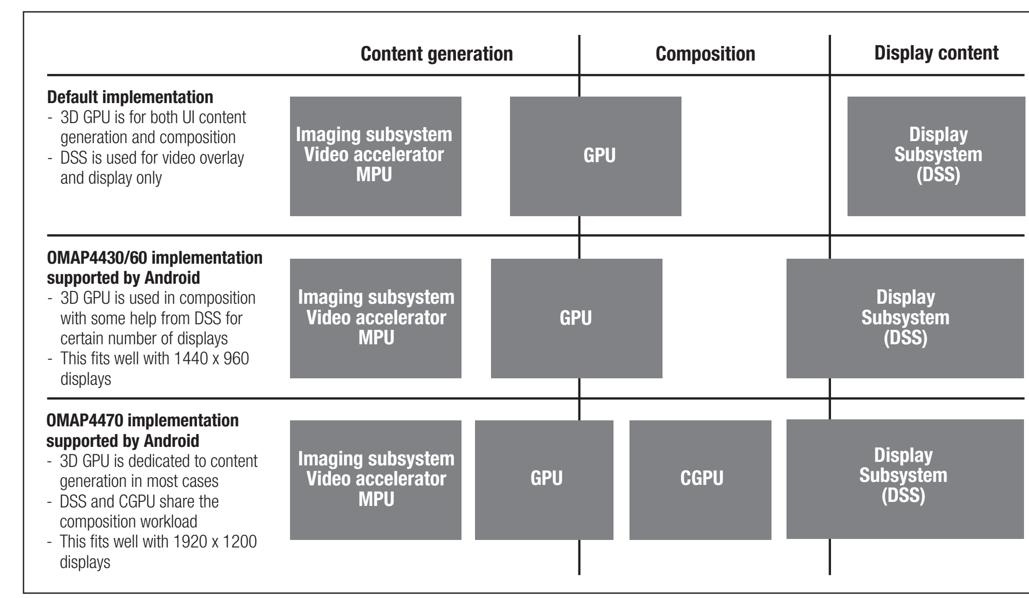
Composition Approach	Strengths	Weakness
Default: GPU handles composition work	Simple, straight-forward approach that can be applied in many systems where the GPU is the only graphics accelerator available for composition tasks.	High GPU loading due to increased usage of GPU in Android for UI content generation. GPU is not the most efficient for the 2D operations required in composition. Overall UI performance can suffer for a high-resolution display if the composition workload consumes too much of the processing capacity of the GPU. High memory bandwidth required for the GPU in the case of composition to a high resolution display.
Alternate: CGPU handles composition work	Offloads composition workload from the GPU to allow it to be used more fully for content generation. CGPU is more efficient for composition. Can achieve blitting and alpha-blending operations in half the time of the GPU. Power savings achieved due to greater efficiency of CGPU for composition work.	Requires same amount of memory bandwidth for composition as in the case above for the GPU.
Distributed: Composition work is split between CGPU and available display subsystem hardware overlays	Achieves significant memory bandwidth savings in most composition scenarios by directly compositing certain graphics and video surfaces to the display using hardware overlays. Remaining surfaces in the composition can be handled by the CGPU. Previous benefits of CGPU composition apply here as well when CGPU handles some of the surfaces in the composition. Reduction in memory bandwidth consumption also leads to further power savings.	The memory bandwidth savings for composition applies only to the extent that there are available hardware overlays for each of the graphics and video surfaces involved in the composition.

Table 1. Composition Approaches for Android 4.0

As we move to supporting high resolution displays, such as a WUXGA display, it is beneficial to offload some or all of the composition effort from the GPU to other hardware accelerators such as the OAMP4470 processor's CGPU or available display hardware overlays that are better-suited for handling 2D composition tasks (such as blitting and alpha-blend operations). There will often be multiple high resolution surfaces fed to the composition engine for a single composition, and the effort involved will consume a higher percentage of the available processing capacity of the GPU if the composition task is not offloaded. In many cases, the load for UI content generation and composition can together consume up to 100 percent of the total processing capacity of the GPU. In fact, more of the UI content generation load has been moved from the MPU to the GPU with Android 4.0, which further points to the benefits for composition performance that can be achieved if the composition work is offloaded from the GPU.

We also see that the composition of these high resolution surfaces comprising the final UI display will require the applications processor to support high memory bandwidth consumption between the graphics accelerators and the system memory. Given the right hardware acceleration blocks available in the system for both composition and display, like those on the OMAP4470 processor, we can optimize the bandwidth required for many display scenarios, which will yield further power savings.

The diagram below shows the progression of different system architectures, starting at the top and moving down, for handling various kinds of content generation, composition, and display. The default approach (at the top) shows the GPU handling all the composition tasks required. As we move down the implementations in the diagram, we see increased usage of other acceleration techniques in the system being used to handle composition as the GPU is increasingly made more available for UI content generation and other graphics processing tasks. This progression is a result of increasingly sophisticated OS capabilities that have been introduced recently on embedded application processors for the mobile space, such as in Android 4.0.



Progression of System Architectures for Android Composition

2 OMAP4470 Processor Features and Benefits for Composition

Listed below are the OMAP4470 processor's features that yield benefits for UI content generation and display composition, especially for large resolution displays such as WUXGA:

	OMAP4470 Mobile Processor
MPU	Two ARM® Cortex™-A9 MPCores with SMP, up to 1.5GHz each L1 cache: 32KB instruction, 32KB Data L2 cache: 1MB for both instruction and data
DDR Frequency	Dual-channel LPDDR2 at 466MHz Memory bandwidth above 5.2GB/s at 70% efficiency
3D UI/Graphics Acceleration	PowerVR™ SGX544 graphics core: - Up to 4096x4096 texture resolution - 2x GFLOPS of SGX540 at the same frequency - 1.4x triangles/sec rate of SGX540 at the same frequency
Composition Graphics Acceleration	CGPU: - BitBLT and StretchBLT - Blending with alpha channel (per-pixel) and global alpha simultaneously - YUV-to-RGB color space conversion - High-quality image and video scaling - 32 K _ 32 K coordinate system
Display Subsystem	DSS embeds the following resources: 3x video display pipes supporting - ARGB and YUV color formats (along with other color formats) - Re-scaling 1x GFX display pipe supporting - ARGB color format (along with other color formats) 1x write-back pipe for memory-to-memory operations supporting display contents capture DSS can be used to refresh main LCD and HDMI displays with different contents

Table 2. OMAP4470 Processor Features Relating to Composition and Display

3 System Profile in Android 4.0 for omposition to a WUXGA Display

When profiling OMAP4470 processor-based Android composition scenarios for a WUXGA display and considering the system loading, we want to look at three areas: 1) the main CPU loading; 2) the GPU loading; and 3) the memory bandwidth consumption.

When profiling composition for this high resolution display, it has been seen that the composition handling will require less than 10 percent CPU loading on the OMAP4470 processor's two ARM® Cortex™-A9 MPcores when running at full speed. In fact, unless there is any other simultaneous processing to be handled on the cores, the operating frequency is automatically dropped to a much lower frequency and the second of the two CPU's in the ARM MPU subsystem is shut off to save power. This is not surprising when you consider that Android's composition engine is offloading all of the main work of composition in each scenario to available hardware accelerators in the system.

When given the minimum set of hardware, the default approach in Android is to allow the GPU to handle all of the composition workload. The figure below shows this approach for handling UI content generation, composition, and display.

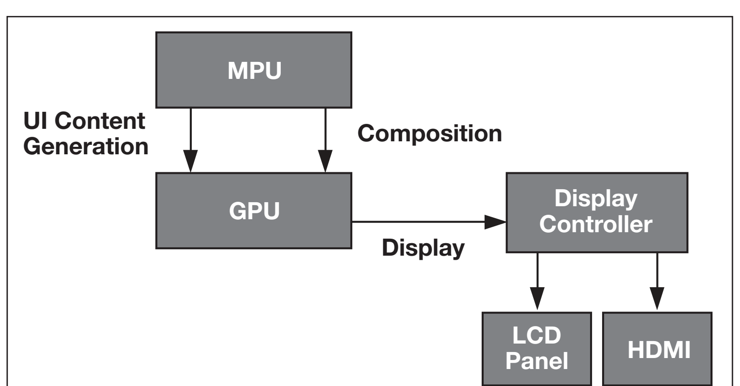


Figure 2. The TCI6489 block diagram with accelerators and peripherals.

However, while using the GPU for composition is the default approach, it is recommended for higher resolution displays to offload as much of the composition work as possible to the dedicated CGPU accelerator on the OMAP4470 processor and the DSS hardware overlay pipelines. This distributed approach will yield faster compositions and more power savings for most composition scenarios.

The composition engine in Android 4.0 provides a hardware adaptation layer (HAL) for just such a purpose. This HAL interface allows for offloading composition workload under most circumstances to any other available hardware accelerators on the applications processor that can handle composition of the surfaces as specified by the Android composition engine. It also allows for specifying a policy that will determine which surfaces should be composited by a particular hardware accelerator block under different conditions. The HAL interface for composition is very beneficial, as it allows for easily porting Android to run on various applications processors with different system designs for handling composition. Some may have additional hardware accelerators that can offload composition tasks and others may only have a single GPU that is used for all composition.

An additional reason for offloading composition from the GPU to the CGPU is that there is already increased usage of GPU resources (via OpenGL ES) in recent versions of the Android OS for other tasks such as creating the elements of the UI itself. Offloading composition from the GPU allows it to be freed for handling these additional graphics processing tasks so that there are no degradations in performance.

It is best to make use of the system's hardware accelerators that are the most efficient for 2D composition tasks. We can save system memory bandwidth (and therefore power), and also increase the speed of composition by using the available hardware overlays in the OMAP4470 processor to scan out a few surfaces directly and do the overlaying or the alpha-blending for the composition in the Display Subsystem (DSS) on-the-fly to the display output. We can also use the CGPU to handle composition of additional surfaces that are not assigned to a hardware overlay or to handle composition for all the surfaces whenever the display contents are not being changed and we want to save additional power. The dedicated CGPU introduced on the OMAP4470 processor is faster and much more power efficient than a GPU for 2D operations such as blitting and alpha-blending used in composition.

The figure below shows the best approach in Android for handling UI content generation, composition, and display for high resolution displays using the OMAP4470 processor with a distributed composition architecture.

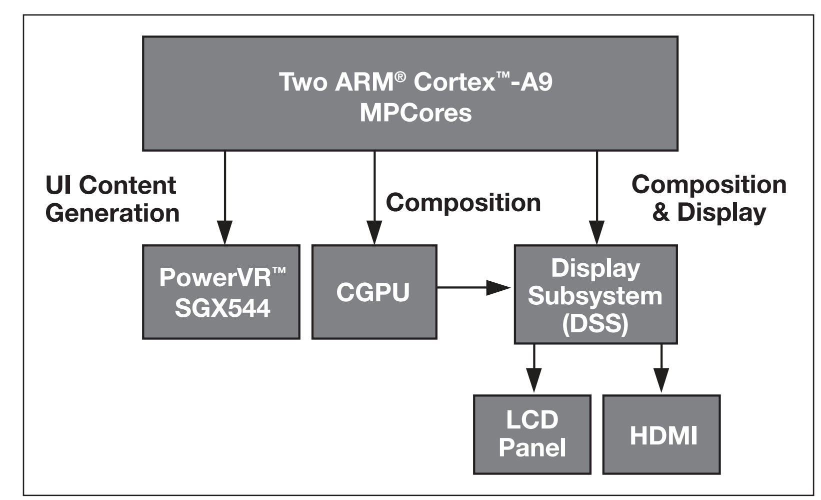


Figure 3. Distributed Approach to UI Composition and Display in Android

The last area to be considered is the memory bandwidth requirement that will be imposed on the system due to the various blitting and alpha-blend operations that are needed for the each display scenario's specific surfaces. As the number of surfaces involved in the composition increases, so will the demand on the memory throughput due to the typical reads and writes necessary for the blitting and blending operations involved in the composition of multiple surfaces to a single output.

The chart below shows both the maximum and the effective (70%) memory bandwidth available for different system solutions with single-channel and dual-channel connections to LPDDR2 SDRAM.

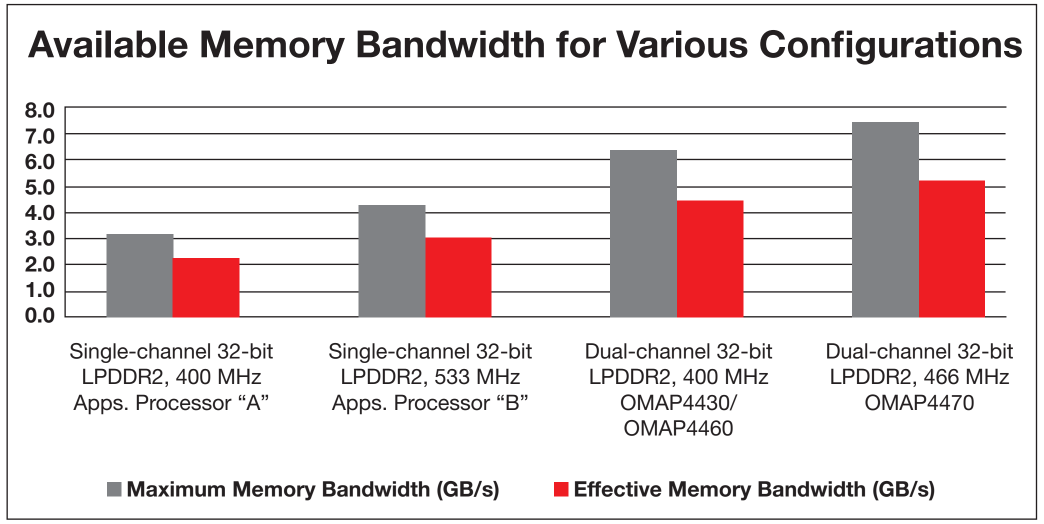


Figure 4. Available Memory Bandwidth for Various Memory Configurations

To determine the memory bandwidth demands that will be placed on the system under many composition scenarios, we'll analyze a few scenarios in particular under the condition that the UI is in an active state of transition. Therefore, the compositions will need to be continually updated at close to the same 60-fps rate as the refresh rate of our example WUXGA LCD display to provide a completely smooth visual UI experience.

4 Analysis of Example Scenarios for Composition to a WUXGA Display

Typical items to be considered for determining if an applications processor is able to handle the composition and output of a particular display scenario at this resolution are: 1) number and size of the graphics and video surfaces to be composited to the screen; 2) color format of each of the surfaces (YUV, ARGB, etc.); 3) which surfaces will require alpha blending; and 4) whether the output is only to the local display or is also required to be sent to an external display at the same time (e.g., connected via HDMI).

Note that the overlay order (Z-order) of the surfaces used in the composition will of course be taken into account during the composition process, but this information is not quite as critical to know when determining the demands on the system assuming that most of the surfaces in each scenario will likely have to be blended together anyway.

For simplicity in estimating the demands on the system, we'll assume that all the surfaces in each scenario will need to be blended.

The following are some typical display scenarios that are often encountered on an Android device. We'll look into the use of different options for handling composition in each of these scenarios as well as review the estimated system memory bandwidth requirements for each.

Home Screen

This composition scenario is for the typical "home" display on an Android device after it is powered on and unlocked. It consists of three surfaces to be composited to the screen:

1) a large background surface "wallpaper" (actually larger than the screen resolution, but only a portion of which is shown at any one time);

2) another large surface, called the "UI launcher", containing all of the application launcher widgets available for the user to choose from on the home screen; and

3) a thin "system bar" stripe displayed on the bottom of the screen.

Composition Output to Local Display

Assuming that each surface has a color format of ARGB (4 bytes/pixel) and, that due to user interaction or some feature of the UI itself, the composition rate needs to be the same as the refresh rate of the display (60 fps), we will have the following memory bandwidth requirements just for the composition and display of the UI (not including any other system traffic) if the GPU alone is used for composition to the frame buffer (FB), which is used to feed the display:

Surface read/write for composition or Display output	Average Memory Bandwidth Calculation	Memory Bandwidth (Mbytes/sec)
GPU: Wallpaper read	1920 x 1128 x 4 (bytes/pixel) x 60 fps	520
GPU: UI launcher read	1920 x 1128 x 4 (bytes/pixel) x 60 fps	520
GPU: System Bar read	1920 x 72 x 4 (bytes/pixel) x 60 fps	33
GPU: Composited output write to the FB	1920 x 1200 x 4 (bytes/pixel) x 60 fps	553
Display: FB scan-out to the display	1920 x 1200 x 4 (bytes/pixel) x 60 fps	553
	Sum Total:	**2,179**

Table 3. Home Screen Composition Memory Bandwidth Using Default Approach (GPU Only)

With the OMAP4470 processor, we also have the option to offload composition directly to the DSS pipelines (hardware overlays) available in the display subsystem. The pipelines can be assigned together to a single display output or else split up between different display outputs. As there are four of these pipelines, we can easily offload all three of the surfaces in this composition example to three of the hardware overlays and then simply bypass the frame buffer altogether.

So, in this case, the final output will be composited directly onto the screen without an intermediate step. This allows us to reduce the memory bandwidth requirements for this scenario since we can bypass the normal series of reads and writes required by the GPU for each composition. As a result, we also gain power savings from the significant memory bandwidth savings (51%) as you can see from the results in the table below:

Surface read/write for composition or Display output	Average Memory Bandwidth Calculation	Memory Bandwidth (Mbytes/sec)
Display: Wallpaper direct to screen	1920 x 1128 x 4 (bytes/pixel) x 60 fps	520
Display: UI launcher direct to screen	1920 x 1128 x 4 (bytes/pixel) x 60 fps	520
Display: System Bar direct to screen	1920 x 72 x 4 (bytes/pixel) x 60 fps	33
	Sum Total: **Savings of 51% vs. GPU Composition**	**1,073**

Table 4. Home Screen Composition Memory Bandwidth Using Display HW Overlays

Composition Output to Local Display and HDMI External Display

Now, for the case where simultaneous output to the local display and to an external display over HDMI is required, we simply fall back to CGPU composition to the frame buffer with the OMAP4470 processor and then mirror the frame buffer output for the local display to the HDMI output as well. In this scenario, we can either use display hardware scaling for the UI to match the 1080p output for the HDMI connection, or else the UI is reconfigured for an exact 1080p output once the System Bar has been cropped. So, we see the following memory bandwidth requirements for this scenario in the table below:

Surface read/write for composition or Display output	Average Memory Bandwidth Calculation	Memory Bandwidth (Mbytes/sec)
CGPU: Wallpaper read	1920 x 1128 x 4 (bytes/pixel) x 60 fps	520
CGPU: UI launcher read	1920 x 1128 x 4 (bytes/pixel) x 60 fps	520
CGPU: System Bar read	1920 x 72 x 4 (bytes/pixel) x 60 fps	33
CGPU: Composited output write to the FB	1920 x 1200 x 4 (bytes/pixel) x 60 fps	553
Display: FB scan-out to the display	1920 x 1200 x 4 (bytes/pixel) x 60 fps	553
Display: FB scan-out to HDMI	1920 x 1200 x 4 (bytes/pixel) x 60 fps	553
	Sum Total:	**2,732**

Table 5. Home Screen Composition Memory Bandwidth with Simultaneous HDMI Output

Note that this shows significant memory bandwidth consumption for even what would seem to be a simple composition and display scenario. Also, we would need to add a bit of margin on top of this composition and display memory bandwidth requirement to ensure that other subsystem bandwidth requirements are accounted for during this scenario. There will be other CPU amd GPU activity involved for creating the elements of the UI and also for handling anything else that might be going on in the system at the same time.

The chart below shows the memory bandwidth requirements for the home screen scenario (composition and display only) under different composition and display scenarios, along with the limits on the effective single-channel LPDDR2 memory bandwidth and the OMAP4470 processor's dual-channel LPDDR2 memory bandwidth for comparison.

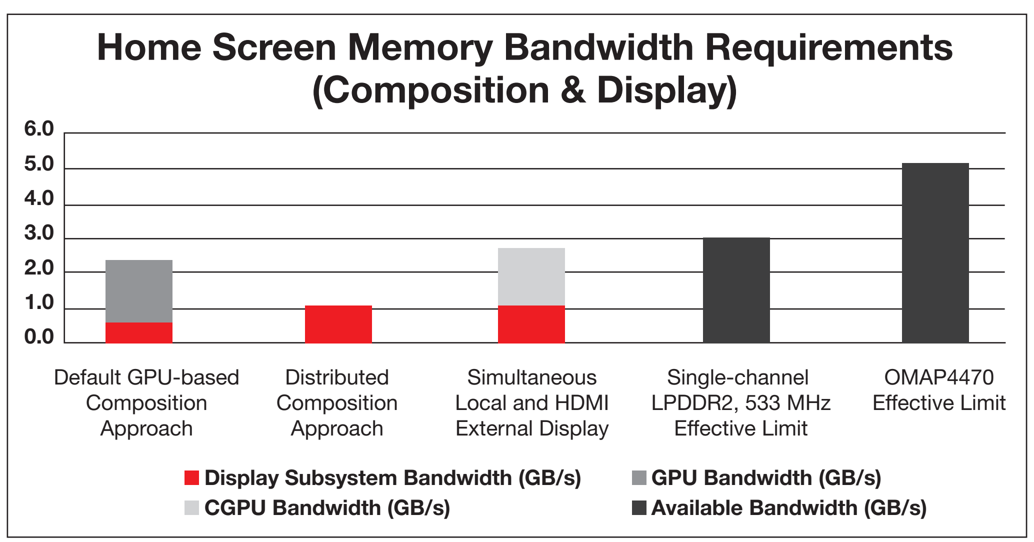


Figure 5. Home Screen Memory Bandwidth Requirements

As we can see from the chart, the spare memory bandwidth remaining in the case of a device with single-channel LPDDR2 configuration is not sufficient to sustain the necessary user experience of a 60-fps update rate while realizing the background OS and application processing required by today's systems at this display resolution. However, in the case a dual-channel LPDDR2 configuration, the system will still have about 50

percent of its bandwidth available to support such activities. In reality, this means that systems with only single-channel LPDDR2 will have to scale back their support to lower screen resolutions or else provide a much less compelling composition rate to the display.

The table below summarizes the bandwidth savings and composition power savings (estimated) that can be achieved relative to the default GPU-only composition approach for this scenario (for single display output):

Composition Approach	Overall Scenario Bandwidth Savings	Composition Power Savings Estimate
GPU-only	—	—
CGPU-only	—	~ 50% (due to CGPU efficiency)
Distributed Approach (DSS / CGPU)	51%	~ 55% (due to DSS usage and BW savings)

Table 6. Home Screen Bandwidth and Power Savings for Different Approaches

Video Player

For this scenario, we have removed the background wallpaper of the home screen and have added a planar YUV video surface for a 1080p resolution video to be played. So, we have a different set of three surfaces to be composited to the screen:

1) a large surface for the application UI (player controls, etc.);
2) a planar YUV video surface at 1080p resolution; and
3) a thin "system bar" stripe displayed on the bottom of the screen.

Composition Output to Local Display

For the case of output only to the local display, we have enough DSS pipelines on the OMAP4470 processor to assign each layer to a hardware overlay. So, in this case composition becomes fairly simple and the three layers are composited directly to the screen. Assume that the planar YUV format of our video surface is 12 bits per pixel and that our graphics surfaces are again ARGB (4 bytes/pixel). Then, with the display refresh occurring at 60fps, we will have the following memory bandwidth requirements just for the composition and display of the UI (not including any other system traffic):

Surface read/write for composition or Display output	Average Memory Bandwidth Calculation	Memory Bandwidth (Mbytes/sec)
Display: UI controls direct to screen	1920 x 1128 x 4 (bytes/pixel) x 60 fps	520
Display: System Bar direct to screen	1920 x 72 x 4 (bytes/pixel) x 60 fps	33
Display: 1080p video direct to screen	1920 x 1080 x 1.5 (bytes/pixel) x 60 fps	187
	Sum Total:	**740**

Table 7. Video Player Composition Memory Bandwidth Using Display HW Overlays

Composition Output to Local Display and HDMI External Display

For the case where simultaneous output to the local display and to an external display over HDMI is required (cloning both the video and the UI to the external display), we will fall back to CGPU composition to the frame buffer on the OMAP4470 processor, and then mirror the frame buffer output to the HDMI output as well as mirroring the video output. For this scenario, we'll use two hardware overlays for each of the two display outputs – one UI output pipe and one video overlay pipe for each display. So, we see the following memory bandwidth requirements for this scenario in the table below:

Surface read/write for composition or Display output	Average Memory Bandwidth Calculation	Memory Bandwidth (Mbytes/sec)
CGPU: UI controls read	1920 x 1128 x 4 (bytes/pixel) x 60 fps	520
CGPU: System Bar read	1920 x 72 x 4 (bytes/pixel) x 60 fps	33
CGPU: Composited output write to the FB	1920 x 1200 x 4 (bytes/pixel) x 60 fps	553
Display: FB scan-out to the display	1920 x 1200 x 4 (bytes/pixel) x 60 fps	553
Display: 1080p video direct to the display	1920 x 1080 x 4 (bytes/pixel) x 60 fps	187
Display: FB scan-out to HDMI	1920 x 1200 x 4 (bytes/pixel) x 60 fps	553
Display: 1080p video direct to HDMI	1920 x 1080 x 1.5 (bytes/pixel) x 60 fps	187
	Sum Total:	**2,586**

Table 8. Video Player Composition Memory Bandwidth with Simultaneous HDMI Output

Note that we would need to add margin on top of this composition and display memory bandwidth requirement, especially for the 1080p video decode operation in this scenario. This will push the total memory bandwidth requirements for the scenario up over 3.0 GBytes/sec, even if it's only for a short amount of time while composition updates are occurring during some period of user interaction.

The chart below shows the memory bandwidth requirements for the video player scenario (full system scenario bandwidth requirements), under different composition and display scenarios, along with the limits on the effective single-channel LPDDR2 memory bandwidth and the OMAP4470 processor's dual-channel LPDDR2 memory bandwidth for comparison.

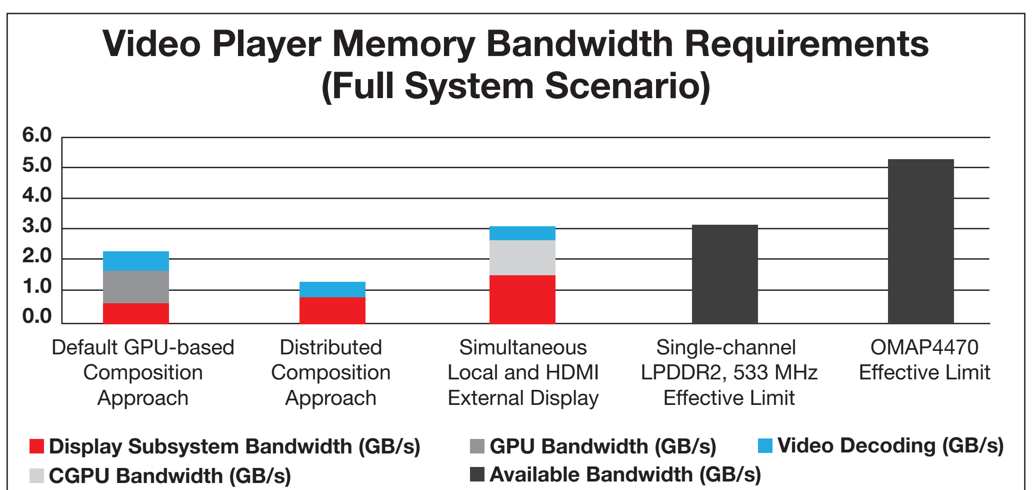


Figure 6. Video Player Memory Bandwidth Requirements

As we can see in the chart, there is no spare bandwidth in the case of a single-channel LPDDR2 configuration and this will not be sufficient to sustain the necessary user experience. However, in the dual-channel LPDDR2 configuration, the system will still have at least 40 percent or more of its bandwidth available to support the required user experience even with additional activity in parallel. Again, this means that systems with only single-channel LPDDR2 will have to scale back their support to lower screen resolutions or else provide a much less compelling composition rate to the display.

The table below summarizes the bandwidth savings and composition power savings (estimated) that can be achieved relative to the default GPU-only composition approach for this scenario (for single display output):

Composition Approach	Overall Scenario Bandwidth Savings	Composition Power Savings Estimate
GPU-only	—	—
CGPU-only	—	~ 50% (due to CGPU efficiency)
Distributed Approach (DSS / CGPU)	48%	~ 66% (due to DSS usage and BW savings)

Table 9. Video Player Bandwidth and Power Savings for Different Approaches

Again, we see that by making use of DSS hardware overlays for composition both the memory bandwidth savings and the composition power savings are significant. Having this additional system margin on the OMAP4470 allows for other parallel usage of system resources, such as would be needed when enabling an external WiFi display simultaneously to view the same video content that is being shown on the local LCD display.

Map Application

For this scenario, we have a map application with multiple surfaces that need to be composited, due to different layers being applied on-screen at the same time. So, in this case, we have a set of seven different surfaces: 1) a surface for the application UI; 2) a surface for the map; 3) a surface for the "directions entry" overlay from the application; 4) a surface for dimming animation activity; 5) a popup window surface from the application for destination choices; 6) a surface for the touch keyboard input method; and 7) a thin "system bar" stripe displayed on the bottom of the screen.

Composition Output to Local Display

For the case of output only to the local display, we can split up the composition of the surfaces between the hardware overlay pipelines and the CGPU on the OMAP4470 processor in order to save some memory bandwidth and therefore some power consumption. So, we'll assign three of the surfaces to hardware overlays and then composite the rest of the surfaces to the frame buffer for output to the display on the remaining display pipeline. Again, we assume that our graphics surfaces are ARGB (4 bytes/pixel). Then, with the display refresh occurring at 60fps, we will have the following memory bandwidth requirements just for the composition & display of the UI (not including any other system traffic):

Surface read/write for composition or Display output	Average Memory Bandwidth Calculation	Memory Bandwidth (Mbytes/sec)
Display: App. UI surface direct to screen	1920 x 1128 x 4 (bytes/pixel) x 60 fps	520
Display: Map layer direct to screen	1920 x 1128 x 4 (bytes/pixel) x 60 fps	520
CGPU: Directions entry overlay read	1920 x 564 x 4 (bytes/pixel) x 60 fps	260
CGPU: Dimming layer read	1920 x 1128 x 4 (bytes/pixel) x 60 fps	520
CGPU: Popup window layer read	1840 x 212 x 4 (bytes/pixel) x 60 fps	94
CGPU: Touch keyboard overlay read	1920 x 684 x 4 (bytes/pixel) x 60 fps	315
CGPU: Composited output write to the FB	1920 x 1200 x 4(bytes/pixel) x 60 fps	553
Display: System Bar direct to the screen	1920 x 72 x 4 (bytes/pixel) x 60 fps	33
Display: FB scan-out to the display	1920 x 1200 x 4 (bytes/pixel) x 60 fps	553
	Sum Total:	**3,368**

Table 10. Map Application Composition Memory Bandwidth

The maps application scenario with multiple surfaces involved demands a large amount of memory bandwidth to be available in the system for short intervals while composition updates are occurring during periods of frequent user interaction. Otherwise, the composition output to the frame buffer for most of the surfaces involved will be done once and then the system bandwidth load will be reduced significantly at that point.

Composition Output to Local Display and HDMI External Display

For the case where simultaneous output to the local display and to an external display over HDMI is required, we will fall back to CGPU composition to the frame buffer on the OMAP4470 processor, and then mirror the frame buffer output to the HDMI output. So, we see the following memory bandwidth requirements for this scenario in the table below:

Surface read/write for composition or Display output	Average Memory Bandwidth Calculation	Memory Bandwidth (Mbytes/sec)
CGPU: App. UI surface read	1920 x 1128 x 4 (bytes/pixel) x 60 fps	520
CGPU: Map layer read	1920 x 1128 x 4 (bytes/pixel) x 60 fps	520
CGPU: Directions entry overlay read	1920 x 564 x 4 (bytes/pixel) x 60 fps	260
CGPU: Dimming layer read	1920 x 1128 x 4 (bytes/pixel) x 60 fps	520
CGPU: Intermediate output write + read	1920 x 1200 x 4 (bytes/pixel) x 60 fps x 2	1106
CGPU: Popup window layer read	1840 x 212 x 4 (bytes/pixel) x 60 fps	94
CGPU: Touch keyboard overlay read	1920 x 684 x 4 (bytes/pixel) x 60 fps	315
CGPU: System Bar read	1920 x 72 x 4 (bytes/pixel) x 60 fps	33
CGPU: Composited output write to the FB	1920 x 1200 x 4(bytes/pixel) x 60 fps	553
Display: FB scan-out to the display	1920 x 1200 x 4 (bytes/pixel) x 60 fps	553
Display: FB scan-out to HDMI	1920 x 1200 x 4 (bytes/pixel) x 60 fps	553
	Sum Total:	**5,027**

Table 11. Map Application Composition Memory Bandwidth with Simultaneous HDMI Output

This is obviously the greatest memory bandwidth demand of any of the scenarios seen so far, but it will only be required for a short time during cases of frequent user interaction with this application (and only with multiple pop-up surfaces present together at once). During this time, the composition rate may temporarily drop down to 50fps while using simultaneous WUXGA and 1080p display outputs. However, the bandwidth demand will be reduced as soon as the current composition update has completed and there is a pause in the user interaction.

The chart below shows the memory bandwidth requirements for the map application scenario (composition and display only), under different composition and display scenarios, along with the limits on the effective single-channel LPDDR2 memory bandwidth and the OMAP4470 processor's dual-channel LPDDR2 memory bandwidth for comparison.

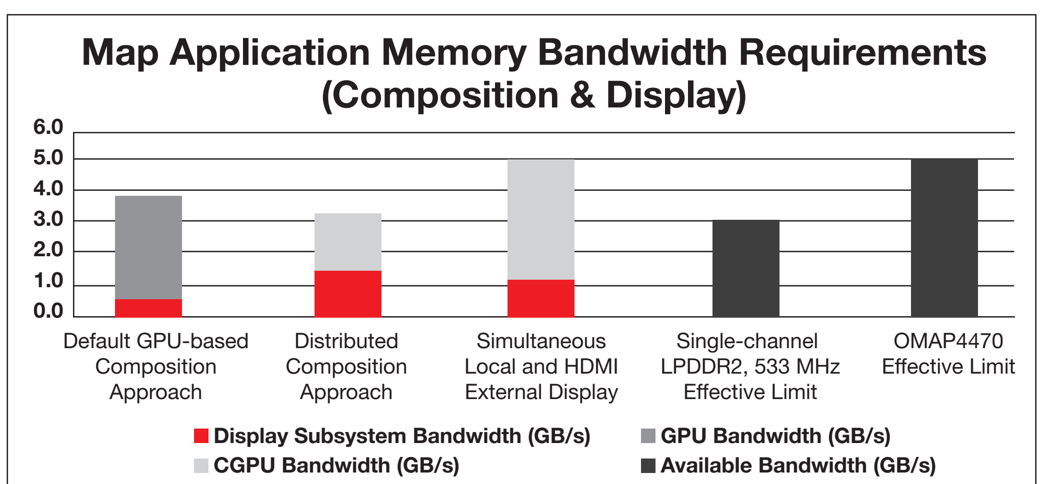


Figure 7. Map Application Memory Bandwidth Requirements

As we can see in the chart, there is not even enough bandwidth in the case of a single-channel LPDDR2 configuration to sustain the necessary user experience at the WUXGA resolution for a single display output. However, in the dual-channel LPDDR2 configuration, the system will still have over 35 percent of its bandwidth available (in the single-display scenario) to support the required user experience even with additional activity in parallel.

The table below summarizes the bandwidth savings and composition power savings (estimated) that can be achieved relative to the default GPU-only composition approach for this scenario (for single display output):

Composition Approach	Overall Scenario Bandwidth Savings	Composition Power Savings Estimate
GPU-only	—	—
CGPU-only	—	~ 50% (due to CGPU efficiency)
Distributed Approach (DSS / CGPU)	14%	~ 55% (due to due to combined DSS / CGPU usage)

Table 12. Map Application Bandwidth and Power Savings for Different Approaches

From the table we can see that by using the distributed composition approach the composition power savings is still significant, even for such a complex scenario with multiple surfaces.

1080p HD Video Teleconference

For this scenario, we have a video teleconference application with two separate video surfaces, along with the rest of the UI. So, we have a set of four surfaces to be composited to the screen: 1) a large surface for the application UI (controls, etc.); 2) a planar YUV video surface for the remote video view at 1080p resolution; 3) another planar YUV video surface for the local video view from the camera at 1080p resolution; and 4) a thin "system bar" stripe displayed on the bottom of the screen.

Composition Output to Local Display

For the case of output only to the local display, we have enough DSS pipelines on the OMAP4470 processor to assign each layer to a hardware overlay. So, in this case composition becomes fairly simple and the four layers are composited directly to the screen. With the display refresh occurring at 60fps, we will have the following memory bandwidth requirements just for the composition & display of the UI (not including any other system traffic):

Surface read/write for composition or Display output	Average Memory Bandwidth Calculation	Memory Bandwidth (Mbytes/sec)
Display: UI controls direct to screen	1920 x 1128 x 4 (bytes/pixel) x 60 fps	520
Display: System Bar direct to screen	1920 x 72 x 4 (bytes/pixel) x 60 fps	33
Display: 1080p local video direct to screen	1920 x 1080 x 1.5 (bytes/pixel) x 60 fps	187
Display: 1080p remote video direct to screen	1920 x 1080 x 1.5 (bytes/pixel) x 60 fps	187
	Sum Total:	**927**

Table 13. 1080p HD VTC Composition Memory Bandwidth Using Display HW Overlays

However, note that we have not yet included any other system memory bandwidth requirements for this scenario yet. We need to add in some approximate memory bandwidth numbers for each of the other system components involved in this video teleconference scenario: 1) imaging subsystem for local camera video at ~200 MB/s; 2) local video H.264 encoding at ~400 MB/s; 3) remote video H.264 decoding at ~500 MB/s; and 4) MPU subsystem traffic for the VTC application and the connectivity at ~ 50 MB/s. Then we arrive at a total system memory bandwidth requirement of 2.1 GBytes/sec.

Composition Output to Local Display and HDMI External Display

For the case where simultaneous output to the local display and to an external display over HDMI is required (cloning both the video and the UI to the external display), we will fall back to CGPU composition to the frame buffer on the OMAP4470 processor, and then mirror the frame buffer output to the HDMI output as well as mirroring the video output. For this scenario, we'll use two hardware overlays for each of the two display outputs – one UI output pipe and one video overlay pipe (for the remote video) for each display. So, we see the following memory bandwidth requirements for this scenario in the table below:

Surface read/write for composition or Display output	Average Memory Bandwidth Calculation	Memory Bandwidth (Mbytes/sec)
CGPU: UI controls read	1920 x 1128 x 4 (bytes/pixel) x 60 fps	520
CGPU: System Bar read	1920 x 72 x 4 (bytes/pixel) x 60 fps	33
CGPU: 1080p local video read	1920 x 1080 x 1.5 (bytes/pixel) x 60 fps	187
CGPU: Composited output write to the FB	1920 x 1200 x 4 (bytes/pixel) x 60 fps	553
Display: FB scan-out to the display	1920 x 1200 x 4 (bytes/pixel) x 60 fps	553
Display: 1080p remote video direct to screen	1920 x 1080 x 1.5 (bytes/pixel) x 60 fps	187
Display: FB scan-out to HDMI	1920 x 1200 x 4 (bytes/pixel) x 60 fps	553
Display: 1080p remote video direct to HDMI	1920 x 1080 x 1.5 (bytes/pixel) x 60 fps	187
	Sum Total:	**2,773**

Table 14. 1080p HD VTC Composition Memory Bandwidth with Simultaneous HDMI Output

Again, note that we have not yet included the other system memory bandwidth requirements for this VTC scenario yet. If we add in the memory bandwidth requirements for the other components involved in the VTC scenario, as we did before in the local display case, then we arrive at a total system memory bandwidth requirement of 3.9 GBytes/sec.

The chart below shows the memory bandwidth requirements for the 1080p HD VTC scenario (full system scenario bandwidth requirements), under different composition and display scenarios, along with the limits on the effective single-channel LPDDR2 memory bandwidth and the OMAP4470 processor's dual-channel LPDDR2 memory bandwidth for comparison.

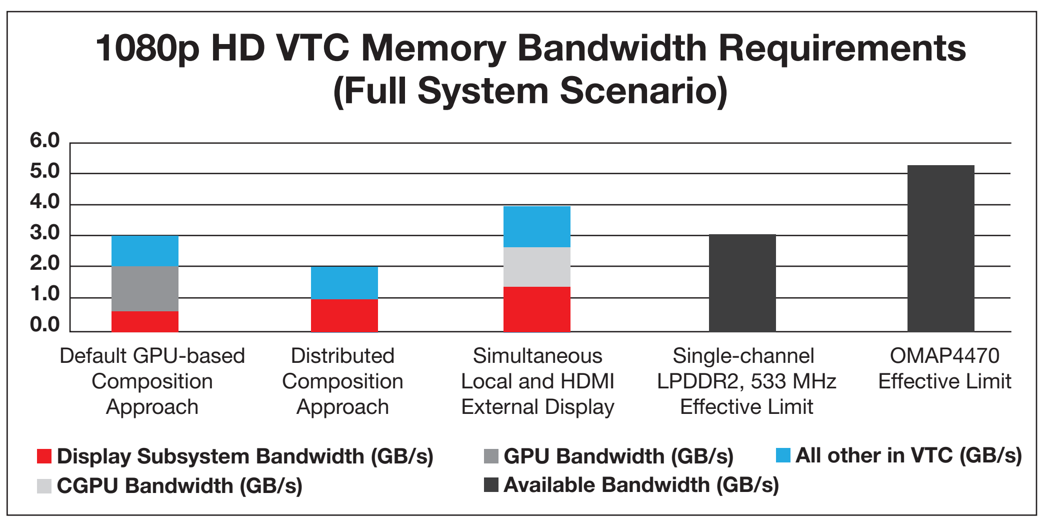


Figure 8. 1080p HD VTC Memory Bandwidth Requirements

The chart above shows the importance of the distributed composition approach (in this case, making use of available DSS hardware overlays) in saving system memory bandwidth for the case of single display output to a WUXGA LCD. In the case of a single-channel LPDDR2 configuration, the system may not even be able to meet the performance required to support this use case at the WUXGA resolution, depending on the

composition approach that is used. However, the dual-channel LPDDR2 configuration of the OMAP4470 allows it to handle the 1080p HD VTC scenario at full performance even with simultaneous output to an external HDMI-connected display.

The table below summarizes the bandwidth savings and composition power savings (estimated) that can be achieved relative to the default GPU-only composition approach for this scenario (for single display output):

Composition Approach	Overall Scenario Bandwidth Savings	Composition Power Savings Estimate
GPU-only	—	—
CGPU-only	—	~ 50% (due to CGPU efficiency)
Distributed Approach (DSS / CGPU)	34%	~ 60% (due to DSS usage and BW savings)

Table 15. 1080p HD VTC Bandwidth and Power Savings for Different Approaches

From the table we can see that by using the distributed composition approach (in this case making use of DSS hardware overlays), both the memory bandwidth savings and the composition power savings are significant.

Advantages of the OMAP4470 Processor's Distributed Approach for Android Composition

Summarizing what we have discussed so far, the advantages for the distributed approach to composition in Android 4.0 on the OMAP4470 processor are the following:

- Very low impact on the MPU with use of the Android Composition Engine (less than 10percent CPU load).
- Offloads most of the composition workload from the GPU to provide margin on the GPU for other, more appropriate 3D graphics processing tasks.
- Makes use of the DSS hardware overlay pipes to handle a significant portion of the composition work and to save memory bandwidth and power in the case of a single display output.
- Makes use of the CGPU accelerator for all remaining composition work to be handled in a fast and power efficient manner.

The OMAP4470 processor also has the system memory bandwidth to handle even the most complicated scenarios for composition and display to a WUXGA LCD display, even when simultaneously displaying to an external display over HDMI.

The OMAP4470 processor can keep a continuous composition rate of at least 60fps or greater, and it can be sustained for multiple high-resolution surfaces being composited using the CGPU accelerator to a 1920 x 1200 resolution frame buffer at the same time that additional high-resolution surfaces are being sent to the screen directly using the display hardware overlays for final composition with the frame buffer output to the WUXGA display.

Conclusion

Today's consumer expectations for mobile experiences increase almost as rapidly as competitive products hit the market. As noted earlier, next generation smartphones, tablets and ultrathin laptops will be expected to do more, particularly with respect to display support and graphics performance. Super-high 1080p resolution is not a future requirement; it is today's requirement if we as an industry expect to deliver on the potential of the mobile processors' and operating systems' capabilities in 2012.

As proven in this paper, Android 4.0-based display composition can effectively make use of the OMAP4470 processor's composition and display accelerators, including the CGPU and the DSS hardware overlay pipelines, to offload composition work from the GPU. This frees the GPU for other more intensive 3D graphics processing tasks common to Android. Also as proven, the OMAP4470 processor yields a valuable additional memory bandwidth margin when driving the single display output scenarios outlined in the paper. We can therefore see that the OMAP4470 architecture can easily handle additional display mirroring and processing required by more complex display scenarios.

For example, adding a Wi-Fi-connected display output to any of this paper's scenarios would not require a change in the composition approach for the local WUXGA display. The OMAP4470 processor has the additional capability to take the local display's output content and feed it into the video hardware accelerator for compression to transmit to a simultaneously-connected Wi-Fi display. This is yet another benefit of the large system memory bandwidth available on the OMAP4470 processor.

However, as shown previously, another mobile processor with only a single-channel LPDDR2 SDRAM interface will not be able to meet the demands on system memory bandwidth required to support composition to a WUXGA display along with simultaneous output to an external display (either HDMI- or Wi-Fi-connected). In addition, for this type of processor, even some composition scenarios where the output is sent only to the local display would result in a significant reduction in composition performance, resulting in a lag in the user's overall experience.

In short, a distributed architecture supported by the advanced, uniquely-balanced system offered by OMAP4470 mobile processor directly benefits high resolution display composition requirements for WUXGA (1920 x 1200 pixels) at 60fps—particularly those mapped out by Android 4.0—even with simultaneous output to a HDMI-connected external display. What's more, the OMAP4470 processor is future-proofed, leaving sufficient headroom in memory, GPU and CGPU performance for future display scenario demands.

Appendix A: Acronyms, Abbreviations & Definitions

Acronym	Definition
API	Application programming interface
CGPU	Composition and graphics processing unit
CPU	Central processing unit
DSS	Display subsystem
FB	Frame buffer
GPU	Graphics processing unit
HW	Hardware
MPU	Microprocessor unit
OMAP	Open Multimedia Application Platform: A multi-core SoC from Texas Instruments
SMP	Symmetric multiprocessing

E010208

Texas Instruments