NVIDIA GeForce GTX 680 Review: Retaking The Performance Crown

by **Ryan Smith** on 3/22/2012 9:00:00 AM

Posted in GPUs , nvidia , Kepler , GeForce , 28nm

The Kepler Architecture: Fermi Distilled

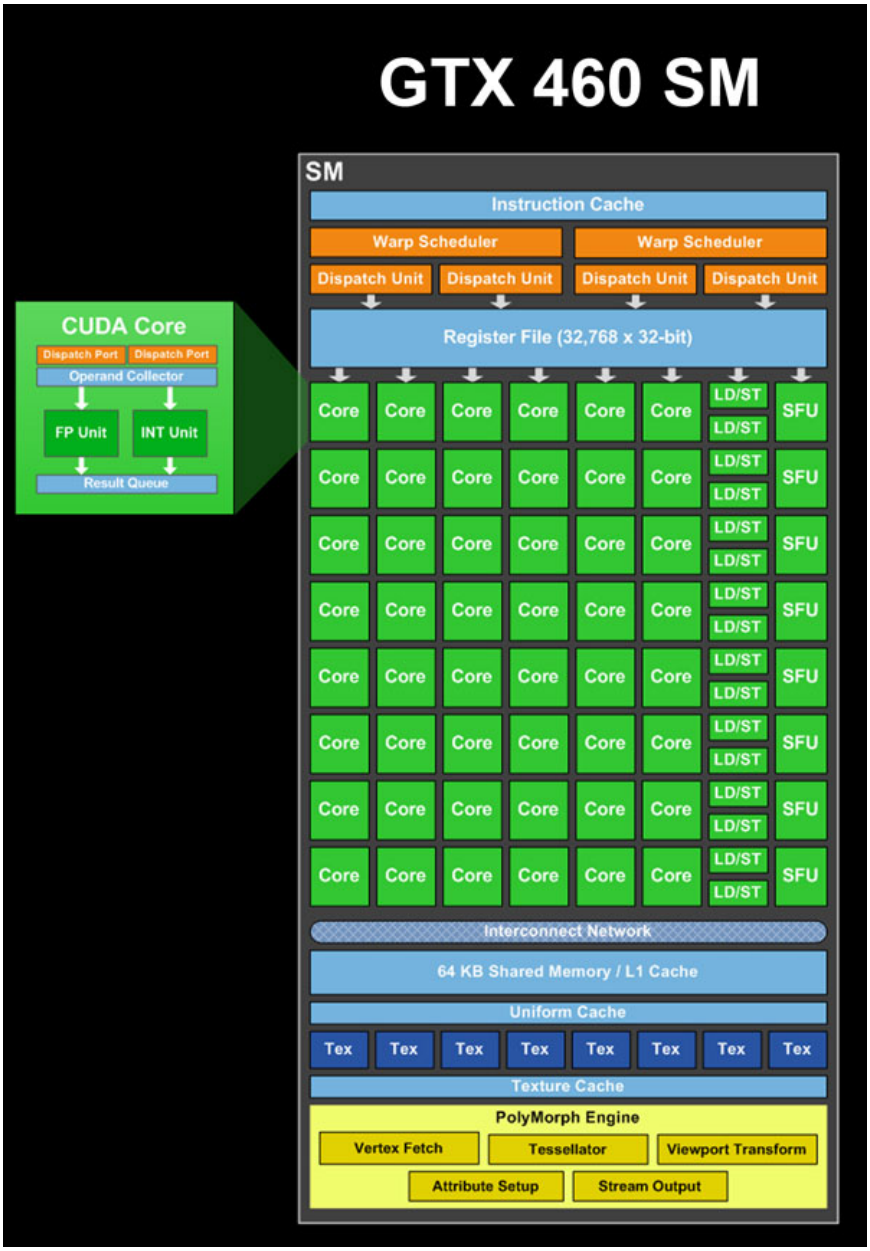
As GPU complexity has increased over the years, so has the amount of time it takes to design a GPU. Specific definitions vary on what constitutes planning, but with modern GPUs such as Fermi, Kepler, and Tahiti planning basically starts 4+ years ahead of time. At this point in time GPUs have a similarly long development period as CPUs, and that alone comes with some ramifications.

The biggest ramification of this style of planning is that because designing a GPU is such a big undertaking, it's not something you want to do frequently. In the CPU space Intel has their tick-tock strategy, which has Intel making incremental architecture updates every 2 years. While in the GPU space neither NVIDIA or AMD have something quite like that – new architectures and new process nodes still tend to premiere together – there is a similar need to spread out architectures over multiple years.

For NVIDIA, Kepler is the embodiment of that concept. Kepler brings with it some very important architectural changes compared to Fermi, but at the same time it's still undeniably Fermi. From a high level overview Kepler is identical to Fermi: it's still organized into CUDA cores, SMs, and GPCs, and how warps are executed has not significantly changed. Nor for that matter has the rendering side significantly changed, with rendering still being handled in a distributed fashion through raster engines, polymorph engines, and of course the ROPs. The fact that NVIDIA has chosen to draw up Kepler like Fermi is no accident or coincidence; at the end of the day Kepler is the next generation of Fermi, tweaked and distilled to improve on Fermi's strengths while correcting its weaknesses.

For our look at Kepler's architecture, we're going to be primarily comparing it to GF114, aka Fermi Lite. As you may recall, with Fermi NVIDIA had two designs: a multipurpose architecture for gaming and graphics (GF100/GF110), and a streamlined architecture built with a stronger emphasis on graphics than compute (GF104, etc) that was best suited for use in consumer graphics. As hinted at by the name alone, GK104 is designed to fill the same consumer graphics role as GF114, and consequently NVIDIA built GK104 off of GF114 rather than GF110.

So what does GK104 bring to the table? Perhaps it's best to start with the SMs, as that's where most of the major changes have happened.



In GF114 each SM contained 48 CUDA cores, with the 48 cores organized into 3 groups of 16. Joining those 3 groups of CUDA cores were 16 load/store units, 16 interpolation SFUs, 8 special function SFUs, and 8 texture units. Feeding all of those blocks was a pair of warp schedulers, each of which could issue up to 2 instructions per core clock cycle, for a total of up to 4 instructions in flight at any given time.

GF104/GF114 SM Functional Units

- *16 CUDA cores (#1)*
- *16 CUDA cores (#2)*
- *16 CUDA cores, FP64 capable (#3)*
- *16 Load/Store Units*
- *16 Interpolation SFUs (not on NVIDIA's diagrams)*
- *8 Special Function SFUs*
- *8 Texture Units*

Within the SM itself different units operated on different clocks, with the schedulers and texture units operating on the core clock, while the CUDA cores, load/store units, and SFUs operated on the shader clock, which ran at twice the core clock. As NVIDIA's warp size is 32 threads, if you do the quick math you realize that warps are twice as large as any block of functional units, which is where the shader clock comes in. With Fermi, a warp would be split up and executed over 2 cycles of the shader clock; 16 threads would go first, and then the other 16 threads over the next clock. The shader clock is what allowed NVIDIA to execute a full warp over a single graphics clock cycle while only using enough hardware for half of a warp.

So how does GK104 change this? The single most important aspect of GK104, the thing that in turn dictates the design of everything else, is that NVIDIA has dropped the shader clock. Now the entire chip, from ROP to CUDA core, runs on the same core clock. As a consequence, rather than executing two half-

Acura Presents 2014 RLX Sedan, Hybrid
AMD Looking to Sell Texas Site As PC Sales Continue to Nosedive
Old Stem Cells Can Be Young Again Thanks to "Fountain of Youth"
Many South Korean Children Have "Internet Addiction," Schools to Teach Dangers
Microsoft: We Will Fix Windows Phone Reboots With OTA Update in December
Windows 8 Beats Windows 7, Sets New Sales Record
HP to Autonomy's ex-CEO Lynch: Have Your Say in Court, Under Threat of Perjury
11/28/2012 Daily Hardware Reviews

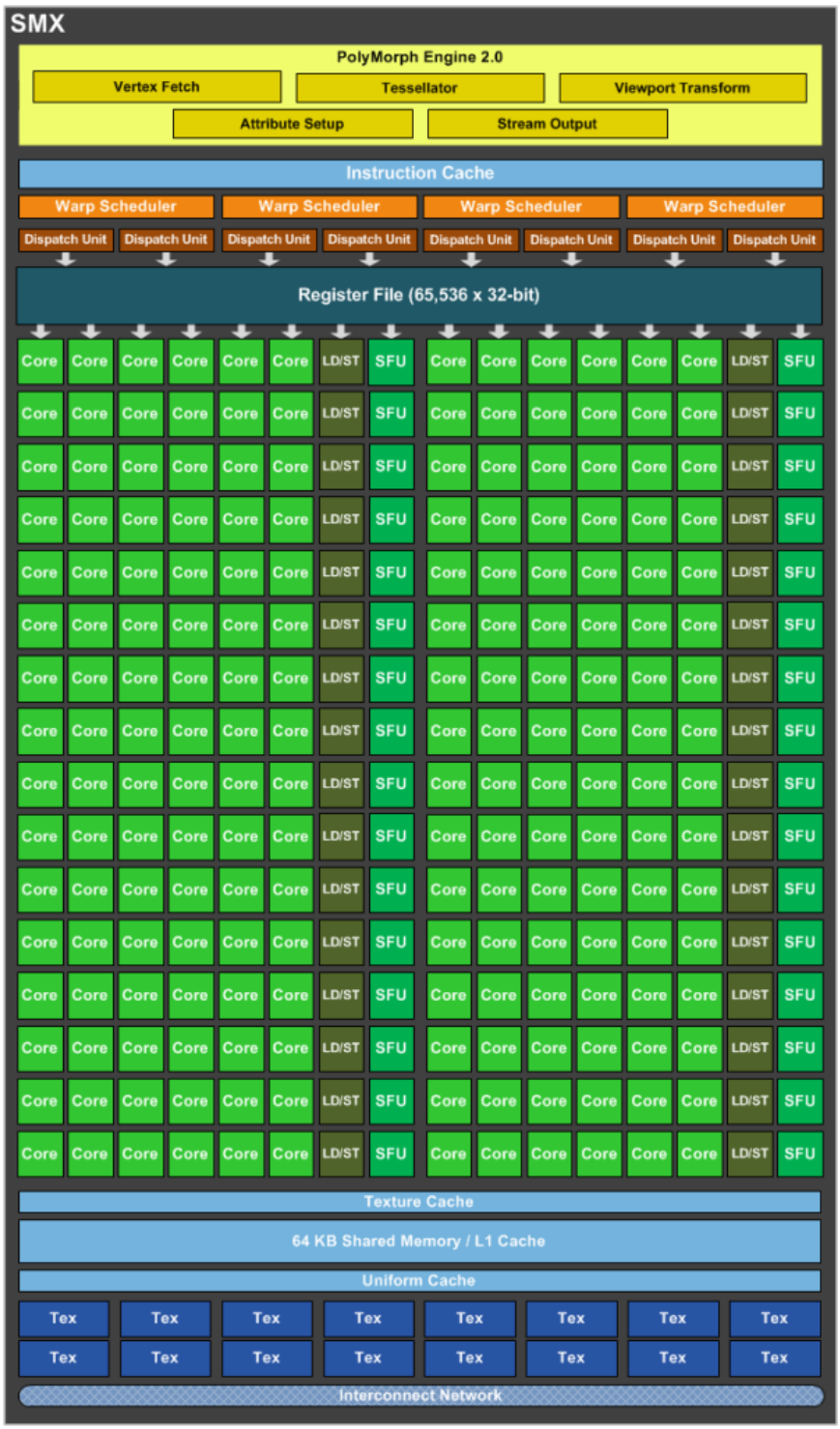
TWITTER

@JeffBessling that's a problem for MS/Intel though, there are a lot of iPad users out there
@MarkRein there seems to be a reluctance to put in the work to enable cellular connectivity unless you know it's going to be a hit
@mcrommert @nerdtalker just recently got his hands on a 920, he's on it :)
@MarkRein I agree, the bigger mistake is launching without Haswell, I hope they update it as soon as Haswell is available though
@silvaalejandro and that's the downside to not using Haswell :-/
@CDemerjian this could be a horribly great idea
@CDemerjian more than USB sure, less than it is today, why not? Cactus Ridge did lower costs after all
@CDemerjian @DrPizza so you think TB just dies eventually, ends up as another failed apple interface?
@CDemerjian if we assume it's never opened up, we can at least assume it'll get cheaper no?
@darkhorse166 I think that's the type of user MS has in mind with all of this Surface stuff, but I feel like it may take one more rev

warps in quick succession GK104 is built to execute a whole warp at once, and GK104's hardware has changed dramatically as a result.

Because NVIDIA has essentially traded a fewer number of higher clocked units for a larger number of lower clocked units, NVIDIA had to go in and double the size of each functional unit inside their SM. Whereas a block of 16 CUDA cores would do when there was a shader clock, now a full 32 CUDA cores are necessary. The same is true for the load/store units and the special function units, all of which have been doubled in size in order to adjust for the lack of a shader clock. Consequently, this is why we can't just immediately compare the CUDA core count of GK104 and GF114 and call GK104 4 times as powerful; half of that additional hardware is just to make up for the lack of a shader clock.

But of course NVIDIA didn't stop there, as swapping out the shader clock for larger functional units only gives us the same throughput in the end. After doubling the size of the functional units in a SM, NVIDIA then doubled the number of functional units in each SM in order to grow the performance of the SM itself. 3 groups of CUDA cores became 6 groups of CUDA cores, 2 groups of load/store units, 16 texture units, etc. At the same time, with twice as many functional units NVIDIA also doubled the other execution resources, with 2 warp schedulers becoming 4 warp schedulers, and the register file being doubled from 32K entries to 64K entries.



Ultimately where the doubling of the size of the functional units allowed NVIDIA to drop the shader clock,

it's the second doubling of resources that makes GK104 much more powerful than GF114. The SMX is in nearly every significant way twice as powerful as a GF114 SM. At the end of the day NVIDIA already had a strong architecture in Fermi, so with Kepler they've gone and done the most logical thing to improve their performance: they've simply doubled Fermi.

Altogether the SMX now has 15 functional units that the warp schedulers can call on. Each of the 4 schedulers in turn can issue up to 2 instructions per clock if there's ILP to be extracted from their respective warps, allowing the schedulers as a whole to issue instructions to up to 8 of the 15 functional units in any clock cycle.

GK104 SMX Functional Units

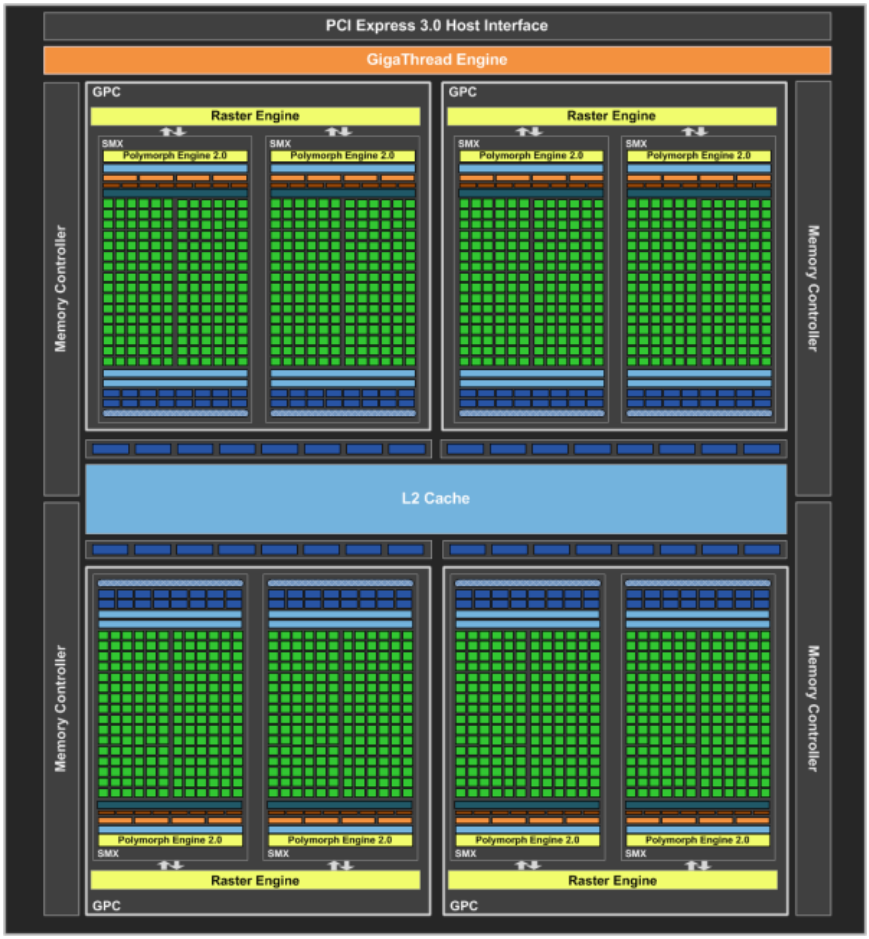
- *32 CUDA cores (#1)*
- *32 CUDA cores (#2)*
- *32 CUDA cores (#3)*
- *32 CUDA cores (#4)*
- *32 CUDA cores (#5)*
- *32 CUDA cores (#6)*
- *16 Load/Store Units (#1)*
- *16 Load/Store Units (#2)*
- *16 Interpolation SFUs (#1)*
- *16 Interpolation SFUs (#2)*
- *16 Special Function SFUs (#1)*
- *16 Special Function SFUs (#2)*
- *8 Texture Units (#1)*
- *8 Texture Units (#2)*
- *8 CUDA FP64 cores*

While that covers the operation of the SMX in a nutshell, there are a couple of other things relating to the SMX that need to be touched upon. Because NVIDIA still only has a single Polymorph Engine per SMX, the number of Polymorph Engines hasn't been doubled like most of the other hardware in an SMX. Instead the capabilities of the Polymorph Engine itself have been doubled, making each Polymorph Engine 2.0 twice as powerful as a GF114 Polymorph Engine. In absolute terms, this means each Polymorph Engine can now spit out a polygon in 2 cycles, versus 4 cycles on GF114, for a total of 4 polygons/clock across GK104.

The other change coming from GF114 is the mysterious block #15, the CUDA FP64 block. In order to conserve die space while still offering FP64 capabilities on GF114, NVIDIA only made one of the three CUDA core blocks FP64 capable. In turn that block of CUDA cores could execute FP64 instructions at a rate of ¼ FP32 performance, which gave the SM a total FP64 throughput rate of 1/12th FP32. In GK104 **none** of the regular CUDA core blocks are FP64 capable; in its place we have what we're calling the CUDA FP64 block.

The CUDA FP64 block contains 8 special CUDA cores that are not part of the general CUDA core count and are not in any of NVIDIA's diagrams. These CUDA cores can only do and are only used for FP64 math. What's more, the CUDA FP64 block has a very special execution rate: 1/1 FP32. With only 8 CUDA cores in this block it takes NVIDIA 4 cycles to execute a whole warp, but each quarter of the warp is done at full speed as opposed to ½, ¼, or any other fractional speed that previous architectures have operated at. Altogether GK104's FP64 performance is very low at only 1/24 FP32 (1/6 * ¼), but the mere existence of the CUDA FP64 block is quite interesting because it's the very first time we've seen 1/1 FP32 execution speed. Big Kepler may not end up resembling GK104, but if it does then it may be an extremely potent FP64 processor if it's built out of CUDA FP64 blocks.

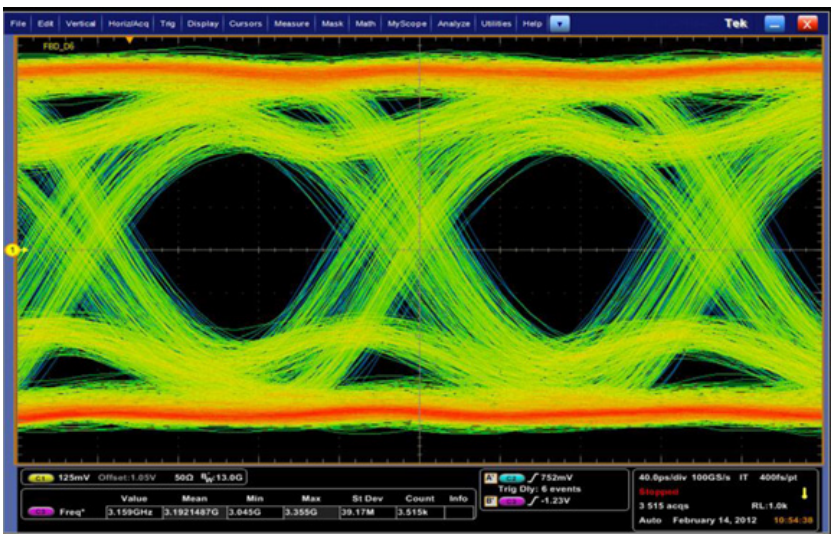
Moving on, now that we've dissected GK104's SMX, let's take a look at the bigger picture. Above the SM(X)es we have the GPCs. The GPCs contain multiple SMs and more importantly the Raster Engine responsible for all rasterization. As with the many other things being doubled in GK104, the number of GPCs has been doubled from 2 to 4, thereby doubling the number of Raster Engines present. This in turn changes the GPC-to-SM(X) ratio from 1:4 on GF114 to 1:2 on GK104. The ratio itself is not particularly important, but it's worth noting that it does take more work for NVIDIA to lay down and connect 4 GPCs than it does 2 GPCs.



Last, but certainly not least is the complete picture. The 4 GPCs are combined with the rest of the hardware that makes GK104 tick, including the ROPs, the memory interfaces, and the PCIe interface. The ROPs themselves are virtually identical to those found on GF114; theoretical performance is the same, and at a lower level the only notable changes are an incremental increase in compression efficiency, and improved polygon merging. Similarly, the organization of the ROPs has not changed either, as the ROPs are still broken up into 4 blocks of 8, with each ROP block tied to a 64 bit memory controller and 128KB of L2 cache. Altogether there are 32 ROPs, giving us 512KB of L2 cache and a 256bit memory bus.

On a final tangent, the memory controllers ended up being an unexpected achievement for NVIDIA. As you may recall, Fermi's memory controllers simply never reached their intended targets – NVIDIA hasn't told us what those targets were, but ultimately Fermi was meant to reach memory clocks higher than 4GHz. With GK104 NVIDIA has improved their memory controllers and then some. GK104's memory controllers can clock some 50% higher than GF114's, leading to GTX 680 shipping at 6GHz.

On a technical level, getting to 6GHz is hard, really hard. GDDR5 RAM can reach 7GHz and beyond on the right voltage, but memory controllers and the memory bus are another story. As we have mentioned a couple of times before, the memory bus tends to give out long before anything else does, which is what's keeping actual shipping memory speeds well below 7GHz. With GK104 NVIDIA's engineers managed to put together a chip and board that are good enough to run at 6GHz, and this alone is quite remarkable given how long GDDR5 has dogged NVIDIA and AMD.



GK104 GDDR5 Signal Analysis

Perhaps the icing on the cake for NVIDIA though is how many revisions it took them to get to 6GHz: one. NVIDIA was able to get 6GHz on the very first revision of GK104, which after Fermi's lackluster performance is a remarkable turn of events. And ultimately while NVIDIA says that they're most proud of the end result of GK104, the fact of the matter is that everyone seems just a bit prouder of their memory controller, and for good reason.

NVIDIA GEFORCE GTX 680 **THE KEPLER ARCHITECTURE: EFFICIENCY & SCHEDULING**

THE KEPLER ARCHITECTURE: FERMI DISTILLED

Like 1,723 people like this. Be the first of your friends.

PRINT THIS ARTICLE

NVIDIA GeForce GTX 680 Review: Retaking The Performance Crown

by **Ryan Smith** on 3/22/2012 9:00:00 AM

Posted in GPUs , nvidia , Kepler , GeForce , 28nm

The Kepler Architecture: Efficiency & Scheduling

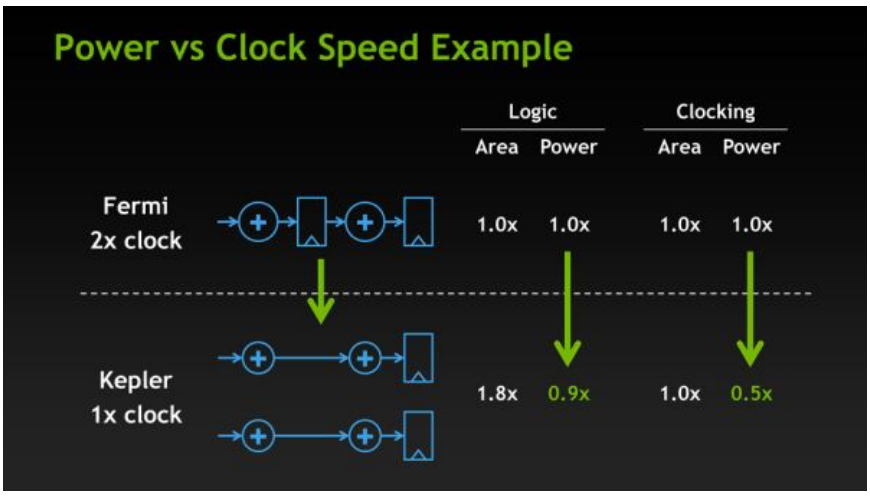
So far we've covered how NVIDIA has improved upon Fermi for; now let's talk about *why*.

Mentioned quickly in our introduction, NVIDIA's big push with Kepler is efficiency. Of course Kepler needs to be faster (it *always* needs to be faster), but at the same time the market is making a gradual shift towards higher efficiency products. On the desktop side of matters GPUs have more or less reached their limits as far as total power consumption goes, while in the mobile space products such as Ultrabooks demand GPUs that can match the low power consumption and heat dissipation levels these devices were built around. And while strictly speaking NVIDIA's GPUs haven't been inefficient, AMD has held an edge on performance per mm2 for quite some time, so there's clear room for improvement.

In keeping with that ideal, for Kepler NVIDIA has chosen to focus on ways they can improve Fermi's efficiency. As NVIDIA's VP of GPU Engineering, Jonah Alben puts it, "[we've] already built it, now let's build it better."

There are numerous small changes in Kepler that reflect that goal, but of course the biggest change there was the removal of the shader clock in favor of wider functional units in order to execute a whole warp over a single clock cycle. The rationale for which is actually rather straightforward: a shader clock made sense when clockspeeds were low and die space was at a premium, but now with increasingly small fabrication processes this has flipped. As we have become familiar with in the CPU space over the last decade, higher clockspeeds become increasingly expensive until you reach a point where they're too expensive – a point where just distributing that clock takes a fair bit of power on its own, not to mention the difficulty and expense of building functional units that will operate at those speeds.

With Kepler the cost of having a shader clock has finally become too much, leading NVIDIA to make the shift to a single clock. By NVIDIA's own numbers, Kepler's design shift saves power even if NVIDIA has to operate functional units that are twice as large. 2 Kepler CUDA cores consume 90% of the power of a single Fermi CUDA core, while the reduction in power consumption for the clock itself is far more dramatic, with clock power consumption having been reduced by 50%.

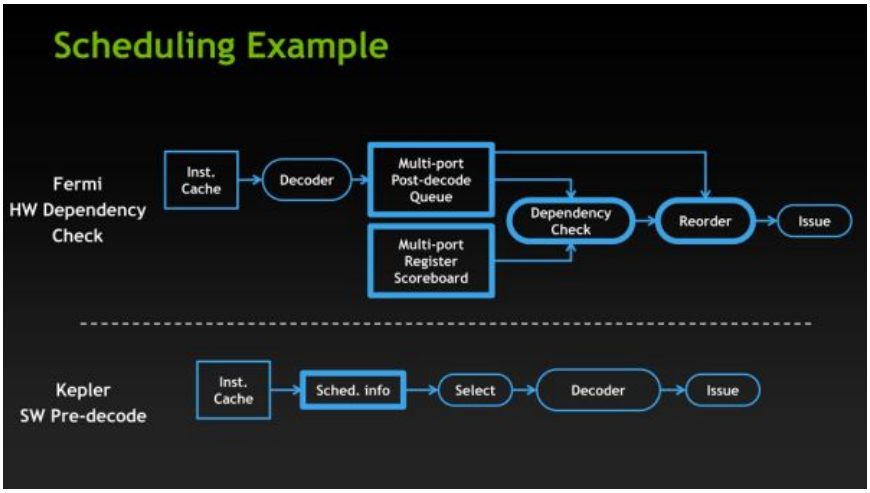


Of course as NVIDIA's own slide clearly points out, this is a true tradeoff. NVIDIA gains on power efficiency, but they lose on area efficiency as 2 Kepler CUDA cores take up more space than a single Fermi CUDA core even though the individual Kepler CUDA cores are smaller. So how did NVIDIA pay for

their new die size penalty?

Obviously 28nm plays a significant part of that, but even then the reduction in feature size from moving to TSMC's 28nm process is less than 50%; this isn't enough to pack 1536 CUDA cores into less space than what previously held 384. As it turns out not only did NVIDIA need to work on power efficiency to make Kepler work, but they needed to work on area efficiency. There are a few small design choices that save space, such as using 8 SMXes instead of 16 smaller SMXes, but along with dropping the shader clock NVIDIA made one other change to improve both power and area efficiency: scheduling.

GF114, owing to its heritage as a compute GPU, had a rather complex scheduler. Fermi GPUs not only did basic scheduling in hardware such as register scoreboarding (keeping track of warps waiting on memory accesses and other long latency operations) and choosing the next warp from the pool to execute, but Fermi was also responsible for scheduling instructions within the warps themselves. While hardware scheduling of this nature is not difficult, it is relatively expensive on both a power and area efficiency basis as it requires implementing a complex hardware block to do dependency checking and prevent other types of data hazards. And since GK104 was to have 32 of these complex hardware schedulers, the scheduling system was reevaluated based on area and power efficiency, and eventually stripped down.



The end result is an interesting one, if only because by conventional standards it's going in reverse. With GK104 NVIDIA is going *back* to static scheduling. Traditionally, processors have started with static scheduling and then moved to hardware scheduling as both software and hardware complexity has increased. Hardware instruction scheduling allows the processor to schedule instructions in the most efficient manner in real time as conditions permit, as opposed to strictly following the order of the code itself regardless of the code's efficiency. This in turn improves the performance of the processor.

However based on their own internal research and simulations, in their search for efficiency NVIDIA found that hardware scheduling was consuming a fair bit of power and area for few benefits. In particular, since Kepler's math pipeline has a fixed latency, hardware scheduling of the instruction inside of a warp was redundant since the compiler already knew the latency of each math instruction it issued. So NVIDIA has replaced Fermi's complex scheduler with a far simpler scheduler that still uses scoreboarding and other methods for inter-warp scheduling, but moves the scheduling of instructions in a warp into NVIDIA's compiler. In essence it's a return to static scheduling.

Ultimately it remains to be seen just what the impact of this move will be. Hardware scheduling makes all the sense in the world for complex compute applications, which is a big reason why Fermi had hardware scheduling in the first place, and for that matter why AMD moved to hardware scheduling with GCN. At the same time however when it comes to graphics workloads even complex shader programs are simple relative to complex compute applications, so it's not at all clear that this will have a significant impact on graphics performance, and indeed if it did have a significant impact on graphics performance we can't imagine NVIDIA would go this way.

What is clear at this time though is that NVIDIA is pitching GTX 680 specifically for consumer graphics while downplaying compute, which says a lot right there. Given their call for efficiency and how some of Fermi's compute capabilities were already stripped for GF114, this does read like an attempt to further strip compute capabilities from their consumer GPUs in order to boost efficiency. Amusingly, whereas AMD seems to have moved closer to Fermi with GCN by adding compute performance, NVIDIA seems to have moved closer to Cayman with Kepler by taking it away.

With that said, in discussing Kepler with NVIDIA's Jonah Alben, one thing that was made clear is that NVIDIA does consider this the better way to go. They're pleased with the performance and efficiency they're getting out of software scheduling, going so far to say that had they known what they know now about software versus hardware scheduling, they would have done Fermi differently. But whether this only applies to consumer GPUs or if it will apply to Big Kepler too remains to be seen.

THE KEPLER ARCHITECTURE: FERMI DISTILLED GPU BOOST: TURBO FOR GPUS
THE KEPLER ARCHITECTURE: EFFICIENCY & SCHEDULING