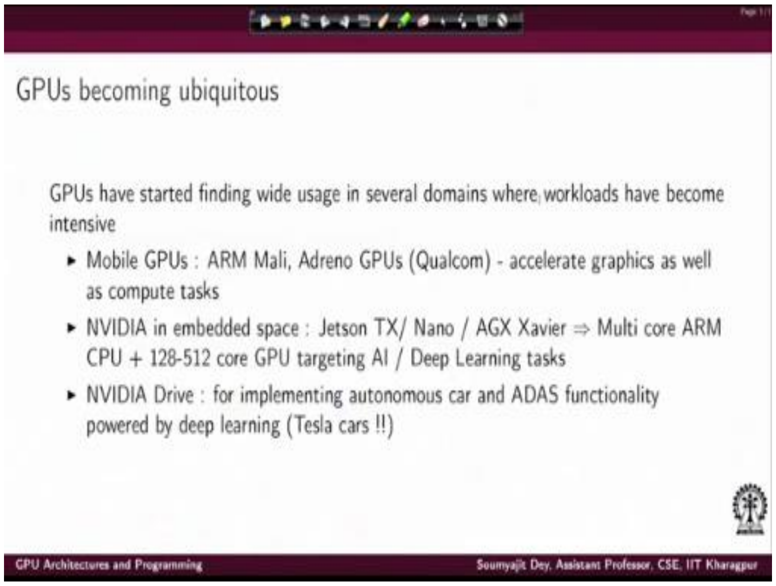
GPU Architectures and Programing

Prof R. Soumyajit Dey
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture no. #8
Intro to GPU Architectures (Contd.)

So, in the last lecture we have been discussing about the different internal architecture sub standard families like Fermi family of GPUs. So, in this lecture, this is going to be the last part.

(Refer Slide Time: 00:37)



GPUs becoming ubiquitous

GPUs have started finding wide usage in several domains where workloads have become intensive

- Mobile GPUs : ARM Mali, Adreno GPUs (Qualcom) - accelerate graphics as well as compute tasks
- NVIDIA in embedded space : Jetson TX/ Nano / AGX Xavier ⇒ Multi core ARM CPU + 128-512 core GPU targeting AI / Deep Learning tasks
- NVIDIA Drive : for implementing autonomous car and ADAS functionality powered by deep learning (Tesla cars !!)

GPU Architectures and Programming   Soumyajit Dey, Assistant Professor, CSE, IIT Kharagpur

Of our topics on GPU architecture basics, we just will just introduce you to this idea that whatever we have been discussing. Maybe there are, I mean they are more about GPU architectures, but we also need to discuss a beat on the current reach of GPU architectures. Otherwise one may start thinking that ok GPU is nothing but a card that you attach to a desktop system, or maybe it's also a specific kind of processing hardware which is present in the cloud.

In this high performance systems around us, which are executing all the nice and fine workloads that kind of priority to their lives and all that. However, in contrast, there is a significant amount of GPU computing going on in devices around us. Which is something many of us may not be very much aware of. So is the sequel in the domain of embedded GPUs. So, just to make my point again is not that we have GPUs present in our workstation class systems.

Our laptops for gaming workloads, or in the cloud. They are also present around us in mobile systems in the embedded space. And also in autonomous driving systems which are slowly getting built. So coming to this domain of mobile. So, in most modern smartphones, you have a GPU core, which is responsible for accelerating the graphics, or the rendering pipelines, for your mobile screen.

A very popular choice of GPU core, which is provided by arm is a mali GPU code. It is present in several families of processors, for example the Samsung Exynos chips, which are present in many of the Samsung smartphones. There is also another very popular offering called the Adreno GPUs, which are present in, mostly I mean almost all Qualcomm chips, there's a Snapdragon chips, which also actually drive mobile smartphones.

A significant number of mobile, mobile systems. So, more or less, these are the two popular choices of GPUs. Of course there are many other choices also but these are the significant ones.
So they can be used to accelerate the graphics workloads and provide a smooth rendering in the smartphone display. Also they can be leveraged for doing computers, just as they can be different for doing computers in the desktop, or the SPs systems.

So, apart from mobile GPUs. We have significant amount of offerings from Nvidia in the embedded space, for example, is Jetson family of devices. The Jetson T Jetson nano, which is the smaller IOT is the type of system offering, and also the high end AGX Xavior of your kind of system. So essentially, these are all embedded boards, comprising multiple from multiple cores, so they have a processing system they have an SOC.

Which has got multi core arm CPU. + GPU, comprising cores varying numbers, maybe starting from 128 to upt 512 cores inside the GPU. So, and that depends on the kind of embedded board we are talking about if we talk about just a nano is the most lightweight embedded board, which are available from Nvidia for accelerating tasks like deep learning or AI in edge devices, not in cloud systems but in the edge devices.

And the next important class of systems where we have significant drive coming from GPUs is the Nvidia drive systems. So, these are the system architectures which are used for implementing autonomous car, which are used for implementing the ADAS functionalities and autonomous driving systems in cars, as we are, I think everybody here is aware of the company (04:55()) manufactures Tesla cars.

And it happens to be the case, at least that many of them. Many of these companies actually implement. They are autonomous driving computers inside the cars on NVIDIA drive kind of platforms. We also implement significant number of ADAS functionality. So what is ADAS? Its autonomous drive assist systems. So, just to introduce you a bit to this topic. I mean, we always keep on hearing about autonomous driving systems which are present inside cars.

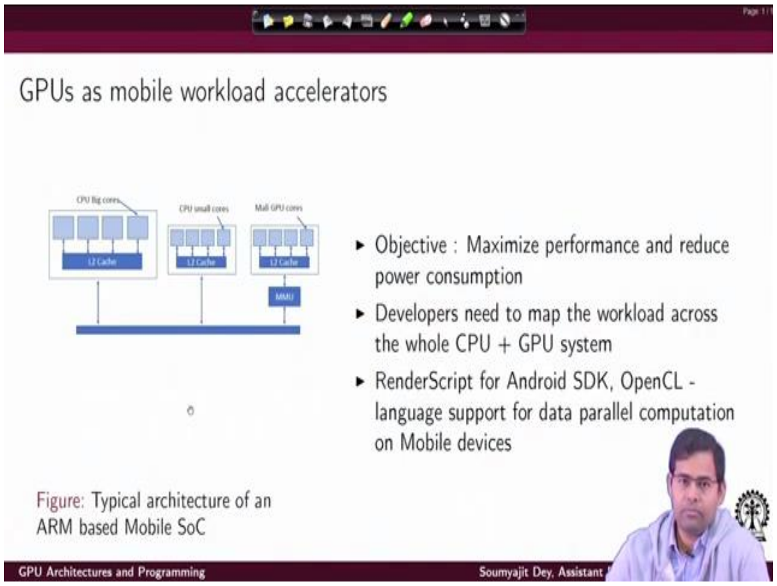
So that a car is going to be driven automatically in like a driverless in a driverless fashion. Where there would be a deep learning computer sitting inside it, which is that computer is this Nvidia drive kind of system is not a normal computer, but it is. It contains a significant amount of engineered resources. Currently containing specific GPUs specialized for this purpose, and also arm cores and other kinds of cores.

So, they will execute our deep learning pipeline, which will try to take all the driving decisions.
On the behalf of the driver, when to accelerate ,when to turn right,when to turn left when to change gears and all that. However, while these are still some of them on an experimental stage, but there are significant vehicle functionalities which get automated is not that the vehicle has always got to move in an autonomous mode.

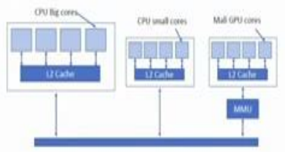
But these are known as drive assist systems that means the vehicle is not fully autonomous. But certain operations of the vehicle can be autonomous automated, for example a park assist system, which can, which can park the vehicle slowly automatically without assistance from the driver. So all those kind of systems the assist systems reform, these kind of ADAS functionalities for a vehicle.

And some of these ADAS functionalities also include are implemented using AI based techniques. And of course, AI based techniques to the NVIDIA drive system for popularly used.

(Refer Slide Time: 07:01)



GPUs as mobile workload accelerators



- Objective : Maximize performance and reduce power consumption
- Developers need to map the workload across the whole CPU + GPU system
- RenderScript for Android SDK, OpenCL - language support for data parallel computation on Mobile devices

Figure: Typical architecture of an ARM based Mobile SoC

GPU Architectures and Programming   Soumyajit Dey, Assistant

So, coming into this idea of mobile workloads, like, what is the GPU doing inside mobile? As we discussed that the primary reason can be, for driving the huge amount of graphics computation required for rendering, nice videos or high speed gaming graphics in a smartphones big display. Also, it can be used for compute loads. For example, trying to do some AI based inferencing on a mobile platform to do an inferencing task on a mobile platform.

For example this smart cameras and live recognition systems you are taking a picture in using a mobile camera and trying to also recognize the people in the picture that would that is also an AI task. If It can be executed on the mobile can save a lot of communication with the cloud. So for this also the mobile GPUs can be leveraged. Now, in this slide we present a typical picture of how current generation mobile resources are organized representative picture.

And is trying to show that a mobile SOC can be quite heterogeneous in nature. For example, you can have multiple CPU systems, along which if GPU cores. So here in this picture we should interconnect system, which is connecting a cluster of CPU big cores. So these are typical kinds of course which which are which are provided by arm, and there can be a cluster of new small cores. And then even have a mali based GPU.

So why are this CPU big cores cluster together because they can, big core I mean, it is clocked at a higher rate, it can have more number of SIM units to present. By small Core i mean it's a low power core, having low amount of processing bandwidth. They have their internal L2 caches so that again the structure as you can see is similar or the SM designs at a larger scale for fermi. So L2 is unified.

It is available to all of different CPU bit cores inside them we will have the L1 cores. Again, they'll do is again unified and is available individually for all the CPU cores, is the big cores from a cluster, the small cores form of clusters. And again, here you have the Mali GPU with four GPU cores. Again, they're connected by unified integration, and there is a memory management tool.

So why is a mobile chip going to be managed in this kind of way, where there are good reasons, as we discussed that this bit core means it can do lot of computing compute ,also executing and consuming more power. These can do less amount of compute, because they are clocked at a smaller frequency, but they will also take less power. And for doing graphics worker processing, you can use the mali GPU.

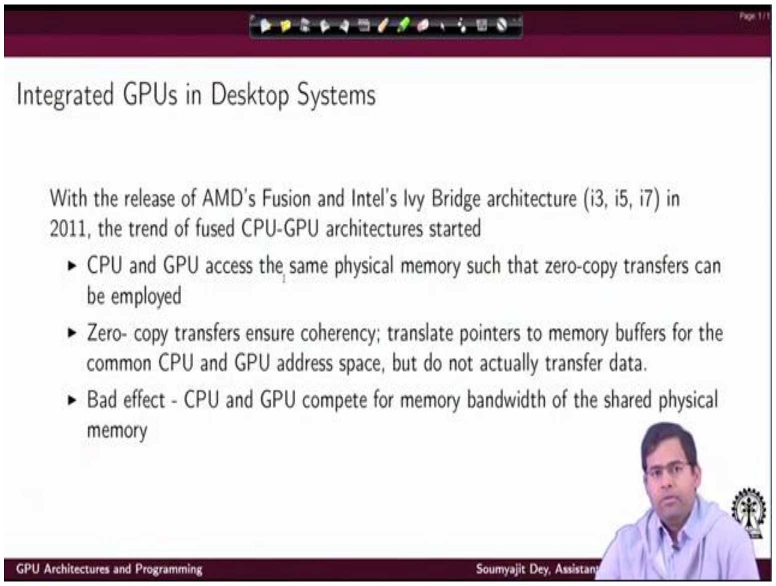
With this fabric of heterogeneous cores available on the mobile platform. It provides significant handle to the developer to decide, and map workloads across this whole CPU,GPU system. So if I have a heavy workload, I would like to map it on this big cluster. If I have a lightweight workload. Or maybe I have a workload which is a bigger of, which is a mixture of heavyweight tasks.

And light weight tasks, I can decide and accordingly intelligently map as a programmer the tasks into the big cluster, or the small cluster, if I have a graphics workload I can intelligently map it to the Mali GPU. So, all these options are available in a modern mobile SOC which is kind of heterogeneous in nature. Now of course, what are the different programming languages available from the SIMP paradigm.

Now we are in this course we are almost speaking about SIMP parallel programming languages which are of the style that is single instruction multiple data. So from that perspective. We are parallel programming support available for mobile devices in terms of languages like open CL, so they provide is data parallel select a language for writing code which is data parallel in nature and the code will execute in a mobile device.

Also, if you are looking into the well known Android SDK Android SDK provides a facility called Render Script, through which you can specify your data parallel program that can be executed inside this cores.

(Refer Slide Time: 11:44)



Integrated GPUs in Desktop Systems

With the release of AMD's Fusion and Intel's Ivy Bridge architecture (i3, i5, i7) in 2011, the trend of fused CPU-GPU architectures started

- CPU and GPU access the same physical memory such that zero-copy transfers can be employed
- Zero- copy transfers ensure coherency; translate pointers to memory buffers for the common CPU and GPU address space, but do not actually transfer data.
- Bad effect - CPU and GPU compete for memory bandwidth of the shared physical memory

GPU Architectures and Programming Soumyajit Dey, Assistant

Now, coming out from the mobile space. Let us also take a look into desktop systems. Now, in desktop systems. Of course you can add an Nvidia card and you have that you have a GPU. But it's not necessary that without that card you do not have a GPU. Now this is something new, right, because we have always thought that is a GPU comes with a separate card. Well in modern CPUs, starting from if I go into AMD is offering starting from AMDs fusion.

And also, if I look into Intel supporting Intel's ivy bridge architecture. So they provide kind of CPU, which also has a fused GPU inside it, so it's like an integrated CPU, GPU system. So, the difference between a large GPU card and this kind of system is that you have a small graphics processing part, sitting on the same dye, and they are sharing the same physical memory. So that zero copy transfers can be employed.

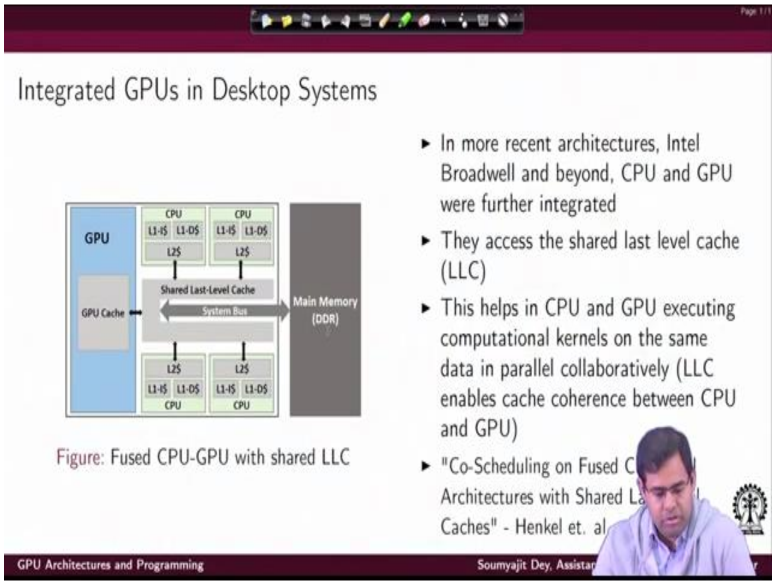
So, essentially, if you have a CPU chip. And you attach a separate GPU card on your desktop system, then they communicate to the PCI Express, and there is always this communication over it. So until and unless you are going to throw some significant amount of parallel computation for the GPU. Maybe the mapping doesn't make sense because apart from gaining in terms of parallel processing you are also losing in terms of the data transfer over it. But that problem is not present here because CPUs and GPUs access the same physical memory.

So, that there is no physical data transport, or you can have this logical zero copy transfers that can be employed. Zero copy transfers, ensure coherency. And essentially what they do is they transferred the pointers to removing their first for a common address is shared by both the CPU and GPU, but physically there is no transfer of data, because everything is in the same memory space, but again there is a benefit to this.

So since the moment is this the physical memory space is shared between the CPU segment and the GPU. Both of them compete for the memory bandwidth of the same shared memory, and that can create interference issues. So, I mean if both of them are trying to compete for the same segment of the memory, some of them, only one of them will will always fit a stall and due to that overall computation will slow and all that.

So, there is always a good thing and also a bad thing with respect to

(Refer Slide Time: 14:25)



Integrated GPUs in Desktop Systems

- In more recent architectures, Intel Broadwell and beyond, CPU and GPU were further integrated
- They access the shared last level cache (LLC)
- This helps in CPU and GPU executing computational kernels on the same data in parallel collaboratively (LLC enables cache coherence between CPU and GPU)
- "Co-Scheduling on Fused C Architectures with Shared L Caches" - Henkel et. al

Figure: Fused CPU-GPU with shared LLC

This kind of an arrangement. Now, something more we have is, with respect with integrated GPUs So, in recent times, from Intel broadwell architecture and beyond CPUs and GPUs are further integrated. So earlier there was the CPU chip and CPU and GPU together sitting on a same chip, and they were faster they were they were actually looking into the same physical main memory.

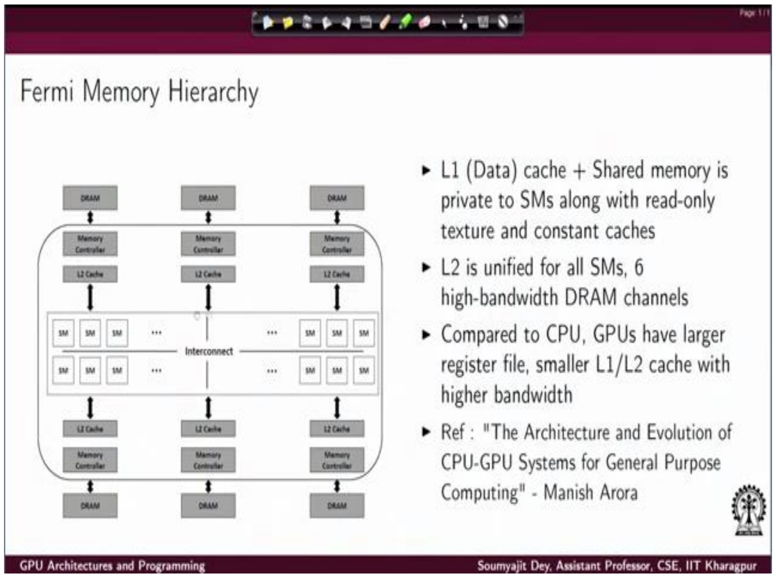
So, but, from this broadwell time. What happened is they added a 3rd level of cache, which is the shared last level cache or LLC. Earlier it was not there earlier you have the GPU with GPU cache, CPU, you have multi well cores of the CPU, each of the cores will have the L1s for instruction and data the L2s. And that is all right. So you have these multiple cores of GPUs and CPUs, and they will be accessing the memory.

But we found the broadwell architecture, what happened is the people added this last level cache inside the system. So what is the good thing about the last level cache, it provides you additional on chip memory. So this helps the CPU and GPU in executing compute kernels on the same data in parallel, collaborative. So, earlier it was not possible, but since I have this last level cache on chip.

They can execute on the kernels, compute kernels both CPU and GPU. And, I mean, I will be accessing the same data and coherence of the data will be maintained through that last level cache. Now, coming to this issue of coherence. So what was the cash really going to do this going to take care by hardware of the coherence decisions. But again, is also will introduce some problems with respect to this we will discuss.

But just to understand the point that, to which the systems have evolved earlier, there was only CPUs and GPUs which are integrated on the chip but with the addition of more cache the hierarchy. Since the cash is also uniformly available to the motor CPUs and GPUs. They do not need to synchronize using the main memory. They can synchronize their computations using the shared last level cache itself so that is the takeaway from this. Coming to this issue of coherence and all that,

(Refer Slide Time: 16:59)



Fermi Memory Hierarchy

- L1 (Data) cache + Shared memory is private to SMs along with read-only texture and constant caches
- L2 is unified for all SMs, 6 high-bandwidth DRAM channels
- Compared to CPU, GPUs have larger register file, smaller L1/L2 cache with higher bandwidth
- Ref : "The Architecture and Evolution of CPU-GPU Systems for General Purpose Computing" - Manish Arora

GPU Architectures and Programming

Soumyajit Dey, Assistant Professor, CSE, IIT Kharagpur

Let us jump back, jump back to one of our earlier slides to understand it better. So, if you remember we were discussing. Let us take this picture that you have in a standard so I'm going back to our original discussion on Fermi GPUs where we have discussed that the L2 unified, and the L1s are distributed across the SMs. Now observe one thing since L1s are distributed across SMs.

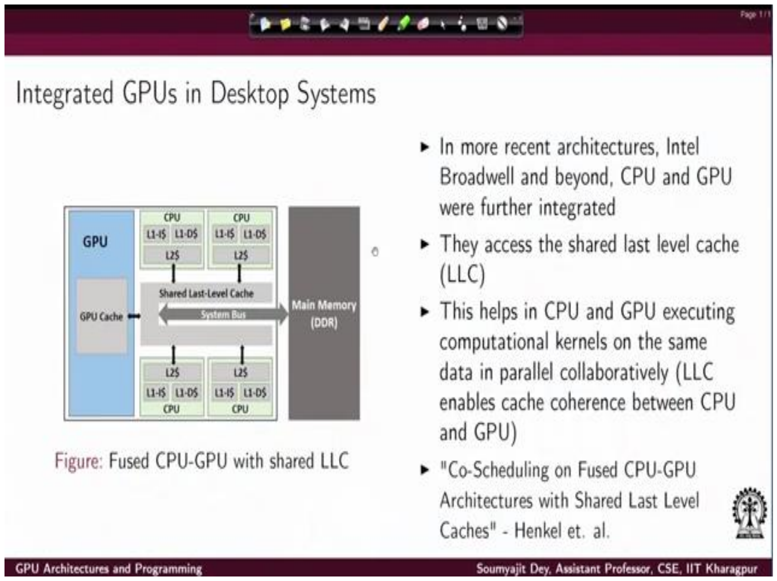
That means each thread. Can, can load a copy of data into, I mean the threats are sitting in different SMS, so they can load copies of data and update those data. And those updates will be visible to the ones, so it can always happen that across SMs, L1s have different updated copies of variables. So, that since the L1s are not unified here. I can technically have the L1s which are not synchronized.

They are, they are containing different possible updated values or variables, and then may lead to consistency issues with respect to data values. This is something we are not going to detail, we are just trying to give you an idea that what can be a bad thing that may have been due to the memory being segmented right. Now this is something that a programmer also needs to keep in mind, and accordingly write the code.

When is trying to do a specific kind of computation such that threads across SMs are trying to work on similar data segments. We 're just trying to highlight that there can be problems with respect to this which a programmer has to manage. These are the things which we will explore in more detail later. So coming back to our current discussion. So as we were discussing here that since we have the shared last level cache.

The good thing is the CPUs and GPUs need not collaborate on data updates. By the by, by synchronizing through them in the morning, but they can do that using this unified cache itself.

(Refer Slide Time: 19:03)



Integrated GPUs in Desktop Systems

- In more recent architectures, Intel Broadwell and beyond, CPU and GPU were further integrated
- They access the shared last level cache (LLC)
- This helps in CPU and GPU executing computational kernels on the same data in parallel collaboratively (LLC enables cache coherence between CPU and GPU)
- "Co-Scheduling on Fused CPU-GPU Architectures with Shared Last Level Caches" - Henkel et. al.

Figure: Fused CPU-GPU with shared LLC

GPU Architectures and Programming — Soumyajit Dey, Assistant Professor, CSE, IIT Kharagpur

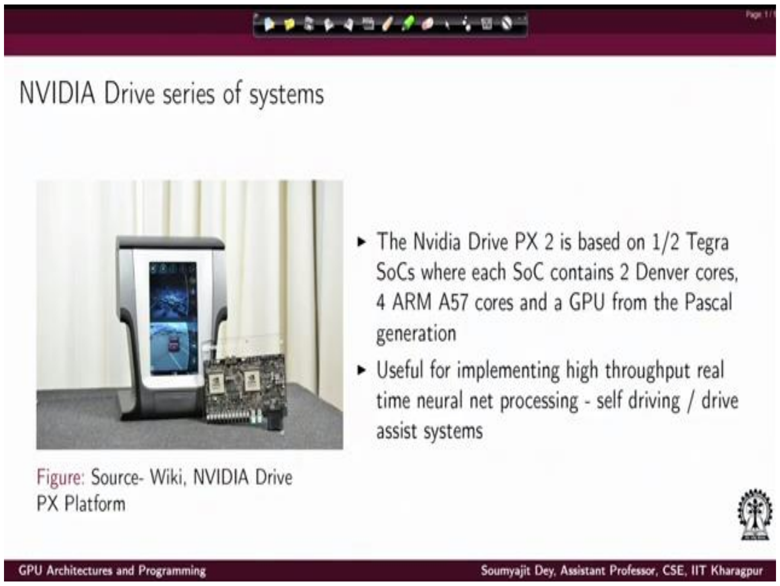
So coming to other kinds of offerings from Nvidia that is the Jetson series, so if you remember we were discussing the embedded space embedded Nvidia has offerings from Jetson for different kinds of offerings a lightweight nano nano systems also the heavy weight AGX Xavior kind of systems. So what are these. So we take specific example of a TK1 SOC from the Jetson series. It incorporates a quad core arm machine.

So it incorporates a quad core arm machine so you have more CPU cores. Each clocked at 2.2 gigahertz, and they are 32 bit systems, and it also has an integrated Kepler family of GPU, the CPUs again share a 2-MB L2 cache, so the L2 is again unified for the CPUs, and the GPU has a 192 core system, and 128 kilobytes L2 cache, so that L2 is also unified for the cores for the GPU.

Now, what's not shown in this picture is that apart from having this cores, if you bit cores, the CPU sub part also has little arm course, of course they are there for low power, low I mean low power compute, which is not frequent too much of performance but then I can always optimize the power and then I will map such workloads to the little, arm cores. Again for you, increasing the bandwidth of memory access that DRAM is banked into 32 banks.

Here banks is of 64 MB. So this is a typical architecture of a Jetson kind of system from Nvidia.

(Refer Slide Time: 20:42)



NVIDIA Drive series of systems

- The Nvidia Drive PX 2 is based on 1/2 Tegra SoCs where each SoC contains 2 Denver cores, 4 ARM A57 cores and a GPU from the Pascal generation
- Useful for implementing high throughput real time neural net processing - self driving / drive assist systems

Figure: Source- Wiki, NVIDIA Drive PX Platform

GPU Architectures and Programming   Soumyajit Dey, Assistant Professor, CSE, IIT Kharagpur

Now, okay, what are the kinds of computers, there are people are offering for cars. So this is one such example, which is basically the Nvidia drive series of system. So the Nvidia drive PX 2 is one of the standard systems that is based on these Tegra Associates from Nvidia Tegra associates another, more powerful associate containing 4 arm a A57 cores And some cores called the Denver cores most that is something that Nvidia separately created.

I mean, for this kind of system, and they also include a GPU from this latest pascal generation. Now, what is the use of such a high performance system in a car like as we discussed that a car in future will have lot of drivers is, as well as still driving facilities, which are, which are essentially deep learning computers. Which also need to take deliver lot of, regular decisions in high throughput inside real time. And those can be implemented in this kind of system targets.

And that is one of the reason for building these kind of drive systems which are much more powerful if we compared them to those in the embedded offer. So just to summarize these are the different kinds of other systems, other computer architectures which are available to us, which provide several forms of data processing capabilities. You have GPUs available in mobiles. You have GPUs available in the embedded space in form of this Jetsons series computers.

You also have heavyweight compute systems from involving GPUs build for the Nvidia drive systems which are part of many well known electric cars, and they provide significant drivers is functionality and also. So with this will end our coverage of the basics of GPU architectures and will shift to the introduction to CUDA programming.And will introduce CUDA as a language and how to write specific kind of CUDA codes.

The kind of programs for the GPU, and leveraging this parallel architectures. That is something we will start covering from the next lecture. Thank you.