Proceedings of the Sixth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC-2022).
DVD Part Number: CFP22OSV-DVD; ISBN: 978-1-6654-6940-1

A review on statistical power modelling for a graphics processing unit (GPU)

Yojan Chitkara
Intel Corporation
yojan.chitkara@intel.com

***Abstract*—The proliferation of portable applications has become a driving force for low power design making it crucial in the development of new processor architectures. Accompanying this is a scaling down of processor nodes which has caused many small channel effects to become more prominent eventually leading to a slow-down in CPU scaling. Smaller nodes in theory could provide better performance at higher frequencies but the apparent slow down has led to a saturation in performance and clock frequencies. This has led to the adoption of heterogeneous computing as a sustainable alternative in computing environments. Graphics Processor Units have gained popularity as a powerful "CPU Co-processor" by reducing the immense workloads on these Processing Units in the computing environments. However, such systems currently lack an effective methodology for power and performance modelling for design optimization. This research study presents a review on the basics of a Graphics Processing unit and the requirement of modelling for power and performance using statistical techniques that help optimize its design to obtain better perf-per-watt results. Methodologies used to obtain prioritized features that affect the power consumed and remove any correlations in data to prevent skewing are also discussed to build an effective Power Model.**

***Index Terms*—Graphics Processing Unit, Performance, Power Neural Network, Compute**

I. Introduction

With the integration of the Personal Computer (or PC) in the everyday life of working and non-working individuals of the world, came a requirement for faster and more efficient computations. As people started to utilize the PC for applications apart from computing (in the general sense) like gaming, there came a requirement for a processor which would render such graphics for the required applications. With this came the development of the GPU or the Graphics Processing Unit, the first of its kind developed by NVIDIA. NV1, the very first GPU by NVIDIA led them to the 2D/3D graphics market with a firm hand. Fueled by the massive growth of the gaming market and its insatiable demand for better 3D graphics NVIDIA developed and evolved the GPU to a computer brain at the intersection of Virtual Reality.[5] For many years, the dynamics of Moore's law have held true. But as of recent CPU scaling has become a challenging task in view of sustainable computing. Since then, GPU computing is defining a new Supercharged Law, specifically due to the huge parallelism in its computing architecture, and hence the world is jumping on board – today there are some 800,000 GPU developers.[6] The Graphics Processing Unit (GPU) or the Virtual Processing Unit (VPU) is a specialized electronic circuit that is designed for the specific purpose of rapidly manipulating and altering memory to accelerate the creation of images in a frame buffer. This has various applications in the field of Embedded Systems, Mobile Phones, Gaming Consoles, Personal Computers etc. As stated earlier, Moore's Law dictates the performance of a device (say CPU) increases with the number of transistors increasing per sq inch on the device double every two years. This creates the obvious problem of higher power density inside chips with decrease in size. Power and Performance modelling comes to an advantage for System Design Engineers as it initiates a feedback mechanism during the design stage and helps make changes in architectures that consume a considerably large amount of power to ones that don't consume as much.[5] An essential step in developing a power and performance model is to understand or characterize the features that affect the change in power and performance and prioritize them from the lesser important characters.

II. GPU Basics

A. Parallelism in CPUs

Gordon Moore of Intel formulated the "Moore's Law" which states that, the clock frequency of a semiconductor core doubles with every passing year. With clock frequencies of single cores reaching saturation points, this law now comes with exceptions. This has caused a shift in paradigm to multi-core and many-core processors. To better comprehend the program flow and execution in GPUs, it is important to understand Parallelism (helps delineate differences between CPUs and GPUs). To carefully understand parallelism in CPUs, it is important to familiarize with the concepts of Pipelining (or Instruction Level Parallelism).

a) Pipelining: Pipelining is a technique of executing multiple instructions in parallel per clock. If Pipelining was absent in CPUs, all the below-mentioned stages would occur in a sequential order, i.e., the processor first, fetches the instructions, then decodes, followed by execution and access to the memory and finally write back to the results to the registers. For this it would require at least 5 cycles to execute one compute instruction. Instruction Level Parallelism can fix the inefficiencies brought about by sequential execution and hence improve resource utilisation, latency, etc. as displayed in Fig.1

978-1-6654-6941-8/22/$31.00 ©2022 IEEE

Inst. No	Pipeline Stage						
1	IF	ID	Ex	Mem	WB		
2		IF	ID	Ex	Mem	WB	
3			IF	ID	Ex	Mem	WB
4				IF	ID	Ex	Mem
5					IF	ID	Ex
Clock Cycle	1	2	3	4	5	6	7

Fig. 1. Instruction Parallelism in CPU's.[6] This figure represents the Instruction Parallelism in CPUs were IF represents an Instruction Fetch, ID represents an Instruction Decode, Ex represents Instruction Execution, Mem represents a Memory Read and WB represents a register Write Back.

B. GPU Design

A magnified description of the GPU architecture is displayed in Figure 2, where the Host represents the CPU, and the GPU consists of multiple SMs (Streaming Multiprocessors), each SM further consists of SPs (Streaming Processors). Each SP is massively threaded and can run thousands of threads per application. At this point it becomes essential to understand the different types of memory available in a GPU.[1]

DRAM or Dynamic Random Access Memory is the most common type of RAM found in systems today. It is relatively slow in terms of accesses and cheaper compared to other memory types. It stores bits as a capacitance charge which is susceptible to leaks and hence requires frequent refreshes to maintain, hence slowing down accesses. SRAM or Static Random Access Memory unlike the DRAM, doesn't need to be refreshed continuously. This is also significantly faster than DRAM and is sometimes also referred to as cache RAM. [6]

Apart from the device DRAM, CUDA hierarchy includes several other types of memory. This hierarchy arises to offset the need to access the slow and expensive DRAM frequently. Several high bandwidth memories are present on a CUDA GPU which CUDA uses as temporary stores (or caches) hence obviating the need for the processor to access DRAM every time. Following are the supported memory types on a CUDA enabled GPU

- Registers (Read/Write per thread)
- Local Memory (Read/Write per thread)
- Shared Memory (Read/Write per block)
- Global Memory (Read/Write per Grid)
- Constant Memory (Read only per Grid).

a) Global Memory: The Host transfers data to-and-from the global and constant memories over the standard PCIe bus. The global memory is the highest latency memory (slowest) from the above-mentioned list. To increase the performance or arithmetic intensity of the kernels, global memory accesses must be as in-frequent as possible. Since there are no restrictions on this memory, all threads in every block can access this unlike the registers and shared memory which are exclusive to threads of a particular block.

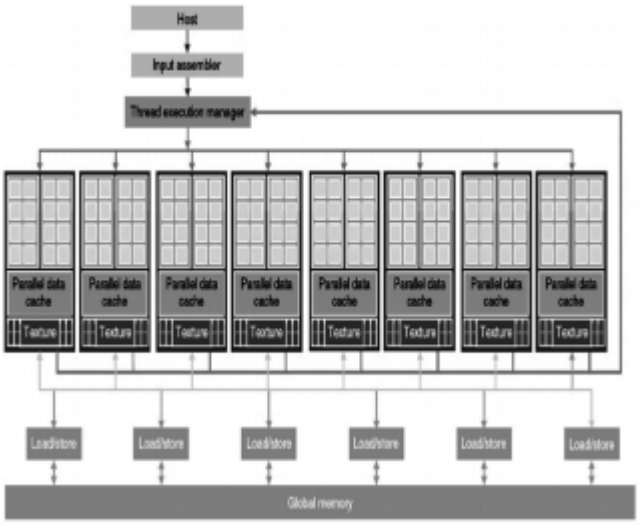

Fig. 2. GPU Architecture.[6] This figure represents the basic architecture of a GPU with its processor units on chip and memory (Global) off chip along with the data flow from Host to GPU.

b) Constant Memory: This memory is Read/Write modifiable by the Host , much like the global memory and can be declared using the keyword "constant". The only thing that separates this memory from the global memory is that it is used for storing data that remains constant over the course of the kernel execution. It supports low-latency, high bandwidth and read only access (data remains constant) by the device when all threads simultaneously access the same location.

c) Shared Memory: All threads of a block can access its shared memory. Since it is present on-chip, like the registers, it has shorter latency and high bandwidths when compared to global memory. They differ, in practice from registers, in the fact that they need to load memory while accessing data, just like in the case of global memory.

III. Silicon Power

Power consumed by an electronic device can be broadly divided into two types:

a) Dynamic Power: Consists of the power consumed during logic transitions and can be sub-divided into two types, Switching and Internal Power.

- Switching Power: As the name suggests, this power manifests itself when there are transitions between logic's as displayed in Fig 3a. It equals the product of the effective charge (Ceff), Voltage (Vdd) and clock frequency (fclk) as displayed in Eq 1.1.

$$P_{sw} = \left(C_{eff}.{V_{dd}}^2.f_{clk}\right) \quad (1.1)$$

- Internal Power: When the input signal is at an intermediate voltage level, a relatively large amount of current flows through the transistors for a brief period of time. This contributes to the Internal Power. If the ramp of the signal is kept short, the short circuit current occurs

only for a small time and hence dynamic power is largely affected only by switching power.

$$P_{int} = (t_{sc}.V_{dd}.I_{peak}.f_{clk}) \quad (1.2)$$

This equals the product of Short Circuit Time (tsc), Voltage (Vdd), Short Circuit current (Ipeak) and clock frequency (fclk)

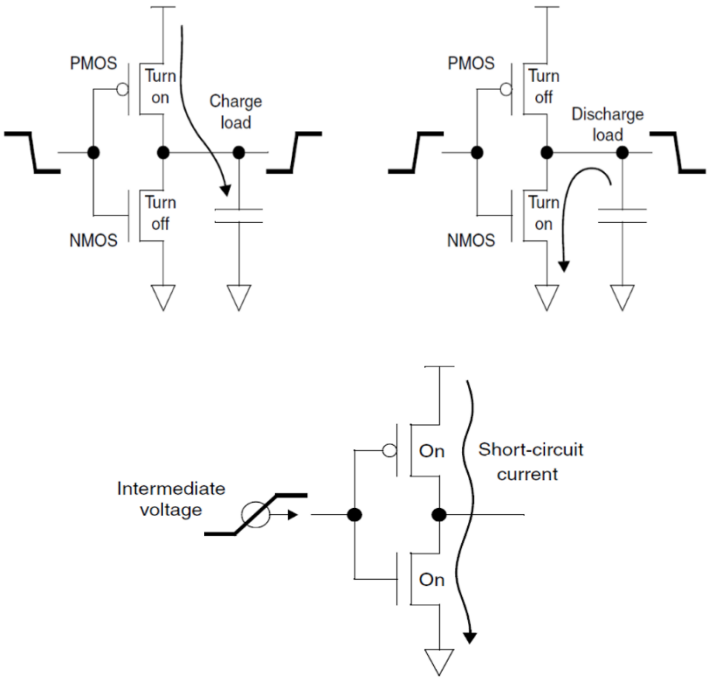

Fig. 3. (Top) Logic transitions in an inverter. Displays logic transitions which are responsible for Switching Power in a MOS circuit.
(Bottom) Short Circuit Current. Displays the cause of short circuit current which is caused when both pmos and nmos are active for a short period of time, giving rise to Internal Power.[18]

b) Static (Leakage) Power: Static power arises whenever power is applied to the device leading to a transistor leakage current. It is consumed even when the chip is quiescent and draws power from nominally OFF devices [18]. Its main causes are

- Reverse bias p-n junction diode leakage caused by minority carrier drift. The width of the depletion region increases when the diode is reverse biased leading to minority carriers getting pushed through the depletion zone causing a small leakage current.
- Sub threshold leakage current flows from drain to source when transistor operates in weak inversion region. Sub threshold leakage is the biggest contributor of leakage power.
- Gate leakage Current which flows directly from the gate to the substrate either through hot carrier injection or oxide tunneling.

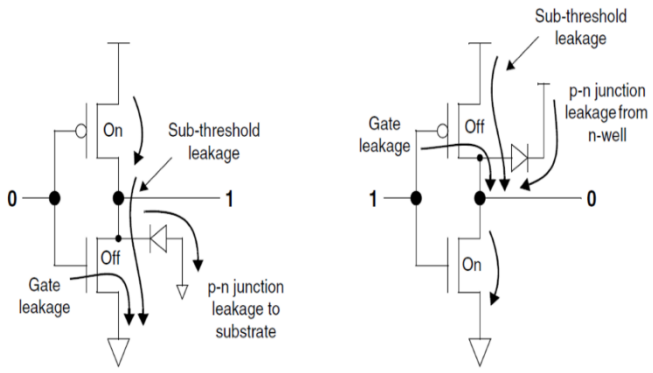

Fig. 4. Static (Leakage) Power sources. Displays the different sources of Static (Leakage) Power during different logic's on a CMOS.[18]

IV. Statistical Modelling

This section provides a modelling approach for GPU accelerated systems using the obtained power and performance data by running compute applications on the GPU. The modelling for power and performance can be approached using simple linear regression tools using the obtained data from performance monitors (PerfMon counters) as independent variables and the obtained power as a dependent variable. The main goal of the power model is to be able to predict the power across various architectures of the GPU and provide timely feedback to the design engineers before the fabrication phase.

a) Power Modelling: Since operating workloads switch the transistor logic on a GPU, the component of power that varies with its changes is Dynamic Power. Hence modelling for Dynamic Power is considered for Power Modelling.

A. Feature Selection

The process for power modelling is initiated by prioritizing the different performance counters by selecting the most important counters that affect the power consumed. This is referred to as Feature Selection. Algorithms that perform this function, return feature priority values that vary heavily with changes in power. Obtaining accurate results requires the data to be varying across workloads and hence requires the execution of multiple workloads.

B. Correlations in Data

Following the above step requires the removal of redundant data that could have emerged from a different configuration of executing the benchmark/workload. After effectively pruning the data from prioritizing to removing any redundant information, Power is modelled using the available data using multiple linear regression. It becomes important to mention a pre-processing step (before implementing the said pruning techniques), Scaling or Normalizing is preferred since obtained performance data are not of equal scale (Ex., Number of instructions executed vs Cache Hits vs Total Bytes transferred,

etc). The data should be scaled to fit within a nominal range and normalised to remove scale dependencies.

V. Conclusion

With the implementation of the mentioned techniques in Section IV an effective power model can be created which helps predict the power accurately during the design phase. To obtain an accurate regression model, data obtained from the benchmark tests must be uncorrelated and evenly distributed across all performance counters i.e., appropriate benchmark tests should be selected to obtain varied data.

References

[1] Karuppusamy, P. ""Design of Inverter Voltage Mode Controller by Backstepping Technique for Nonlinear Power System Model."" Journal of Electrical Engineering and Automation 3, no. 4 (2021): 265-276.
[2] Siddhartha Sankar Mondal, "GPU: Power vs Performance", Department of Information Technology, Uppsala Universitet, June 2014.
[3] S Sundar, M Panchatcharam, "GPU Basics", Department of Electronics and Communication, Indian Institute of Technology, August 2014, Madras.
[4] Sunpyo Hong, Hyesoon Kim, "An Integrated GPU Power and Performance Model", School of Computer Science, Georgia Institute of Technology, in International Parallel and Distributed Processing Symposium, USA, July 2010.
[5] D.C Price, M.A Clark, "Optimizing performance-per-watt on GPUs in High Performance Computing", arXiv.org, vol 3, 20th October 2015, pp 1-9.
[6] Hitoshi Nagasaka, Naoya Maruyama, "Statistical Power Modelling of GPU Kernels using Performance Counters", in IEEE Explore, Japan Science and Technology Agency, CREST, July 2010, pp 1-8.
[7] CUDA – Introduction to GPU, www.tutorialspoint.com/cuda
[8] GPU Fundamentals, https://slideshare/GPU
[9] Y. Abe, H. Sasaki, S. Kato et al., "Power and performance characterization and modeling of gpu-accelerated systems," in Parallel and Distributed Processing Symposium, 2014 IEEE 28th International, May 2014, pp. 113–122.
[10] I.Paul, W.Huang, M.Arora et al., "Harmonia: balancing compute and memory power in high-performance GPU," in Proceedings of the 42nd Annual International Symposium on Computer Architecture, ser. ISCA '15. New York, NY, USA: ACM, 2015, pp. 54–65.
[11] G. Wu, J.L.Greathouse, A.Lyashevsky et al., "gpu power and performance estimation using ML," in 2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA), Feb 2015, pp. 564–576.
[12] J.Coplin and M.Burtsher, "Energy, power and performance characterization of gpu benchmarks," in 2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), May 2016, pp. 1190–1199.
[13] G.Chen, B.Wu, D.Li, and X.Shen, "Porple: An extensible optimizer for portable data placement on gpu," in 2014 47th Annual IEEE/ACM International Symposium on Microarchitecture, Dec 2014, pp. 88–100.
[14] R. Ge, R. Vogt, J.Majumder et al., "Effects of dynamic voltage and frequency scaling on k20gpu," in 2013 42nd International Conference on Parallel Processing, Oct 2013, pp. 826–833.
[15] J. Guerreiro, A. Ilic, N. Roma et al., "Multi-kernel auto-tuning on gpu : Performance and energy-aware optimization," in 2015 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, March 2015, pp. 438–445.
[16] E. Lindholm, J. Nickolls, S. Oberman et al., "Nvidia tesla: A unified graphics and computing architecture," in IEEE Micro, vol. 28, no. 2, pp. 39–55, Mar. 2012.
[17] Junke Li, Bing Guo, "Power Modelling Approach for GPU Source Program" in Journal of EE and Technology, Sichuan University, vol.13, Jan. 2018, pp. 181-191.
[18] Neile H.E Weste, CMOS VLSI Design, 4th Edition, pp. 123-135