6th International Conference on
I-SMAC (IoT in Social, Mobile, Analytics and Cloud)
I-SMAC 2022

A REVIEW ON STATISTICAL POWER MODELLING FOR A GRAPHICS PROCESSING UNIT (GPU)

Presented By

Yojan Chitkara

Table of Contents

- INTROCUCTION
- GPU BASICS
- SILICON POWER
- RESULTS
- REFERENCES

INTRODUCTION

Scaling Architectures

- GPU computing is defining a new Supercharged Law, specifically due to the huge parallelism in its computing architecture.

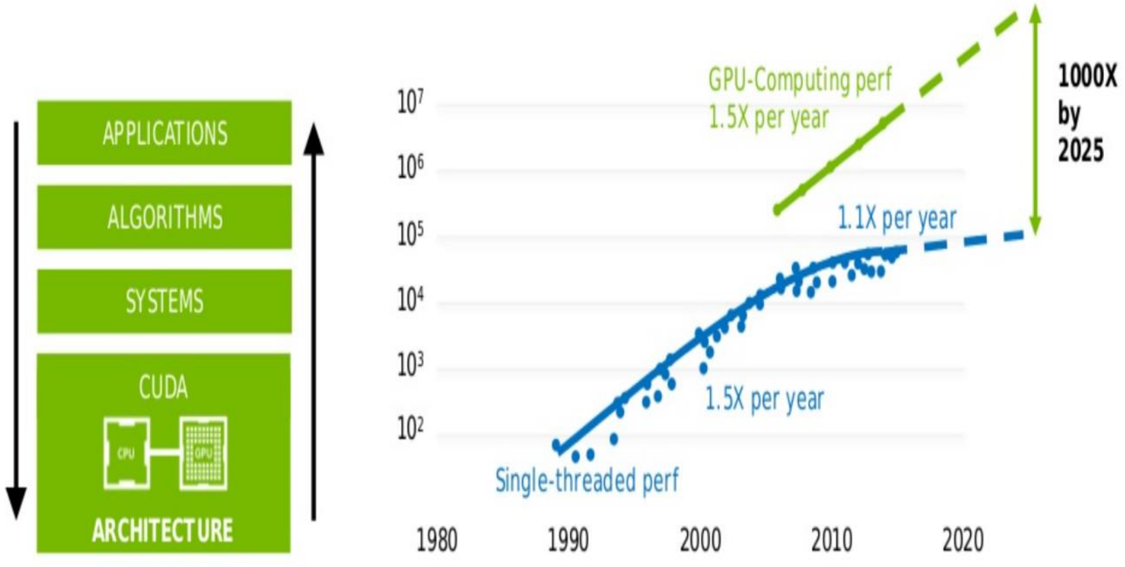


Figure 1.1: GPU Parallelism Compute Perf [7]

Shifting Trends towards GPU

- The Graphics Processing Unit (GPU) or the Virtual Processing Unit (VPU) as shown in Figure 1.2 is a specialized electronic circuit designed to rapidly manipulate or alter memory to accelerate heavy workloads in shorter intervals of time as compared to a CPU.

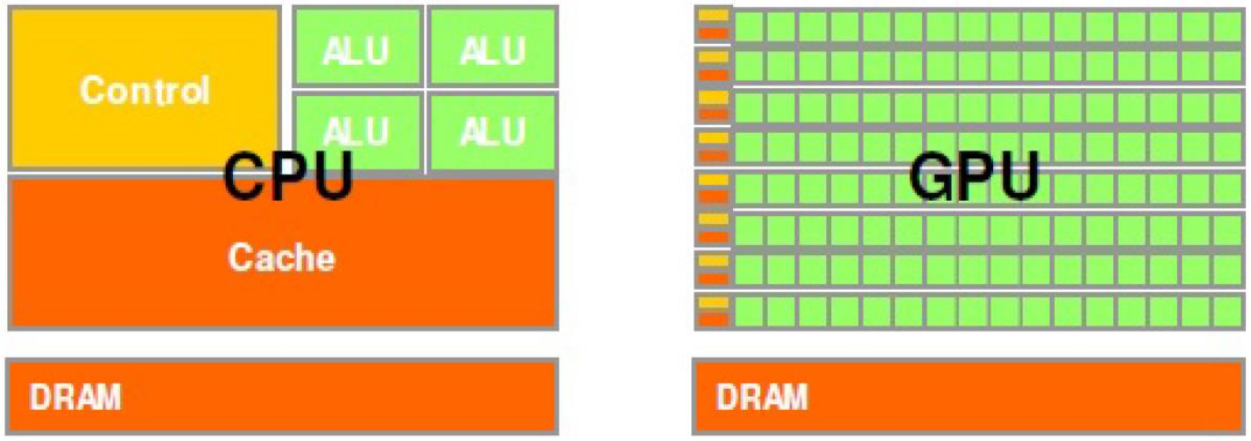
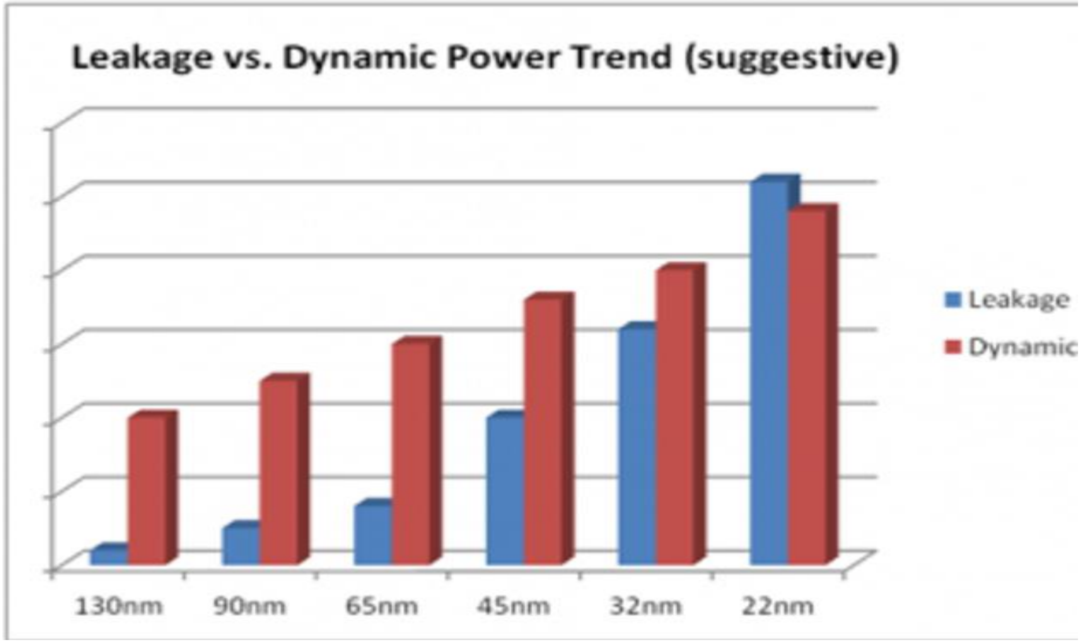


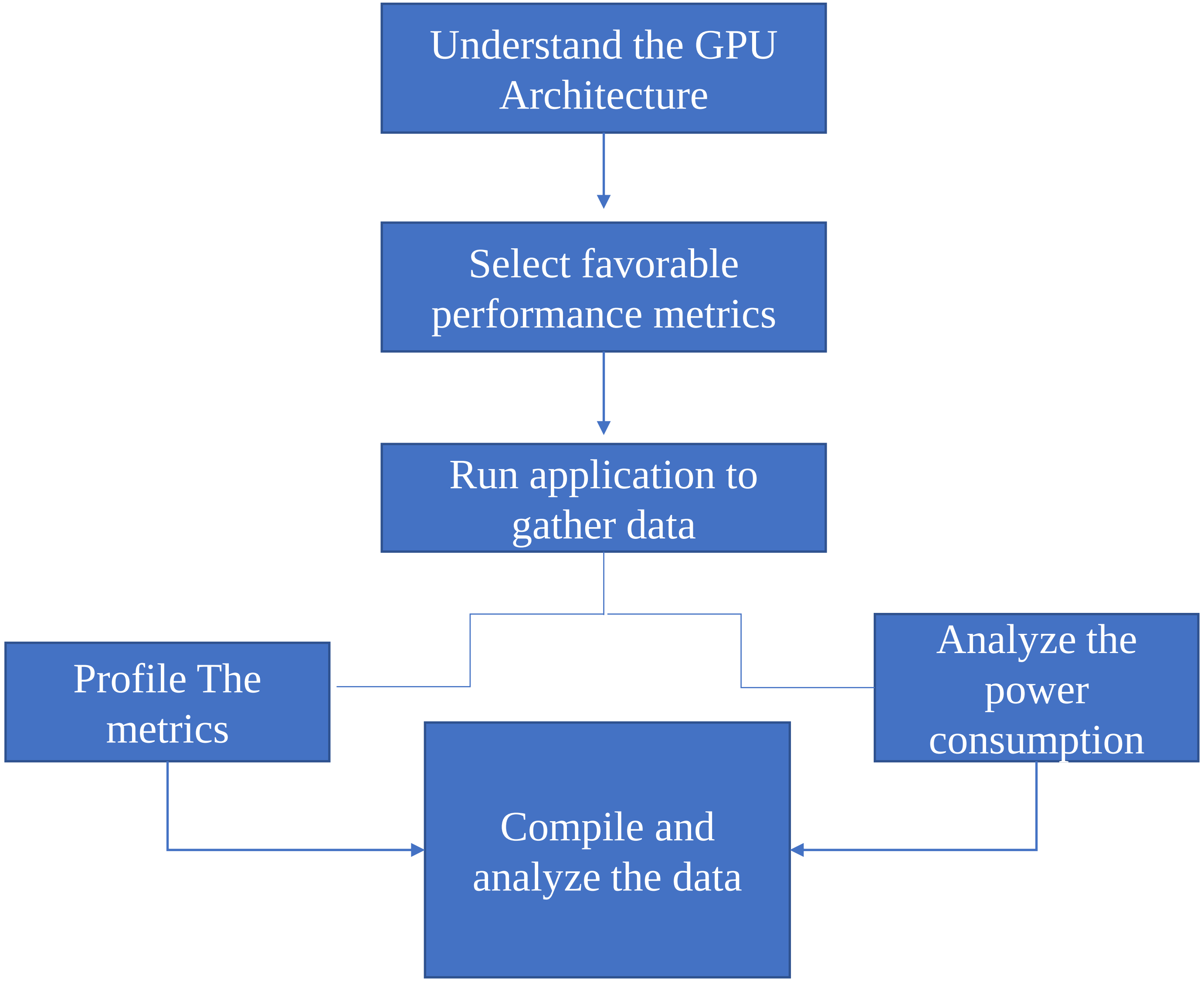
Figure 1.2: Parallelism between CPU and GPU architectures. [2]

Motivation

- Designing for low power as architectures scaled down in technology became essential as number of transistors on a dye kept on increasing to accommodate more cores.
- It is hence becoming increasingly difficult for speed to keep par with increasing number of transistors.



Working Methodology



Understand the GPU Architecture

Select favorable performance metrics

Run application to gather data

Profile The metrics

Analyze the power consumption

Compile and analyze the data

GPU BASICS

Parallelism in CPU’s

- The five essential steps required for an instruction to execute in a CPU RISC Architecture are:
  - Instruction Fetch (IF)
  - Instruction Decode (ID)
  - Instruction Execute (Ex)
  - Memory Access (Mem)
  - Register Write-Back (WB)

Pipelining

- Pipelining is a technique of executing multiple instructions in parallel per clock. Instruction Level Parallelism can fix the inefficiencies brought about by sequential execution like latency, resource utilization.

Instruction 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**

GPU Design

- A GPU comprises of a large number of cores (significantly greater than a CPU) each of which run at a clock speed significantly slower than a core of a CPU.

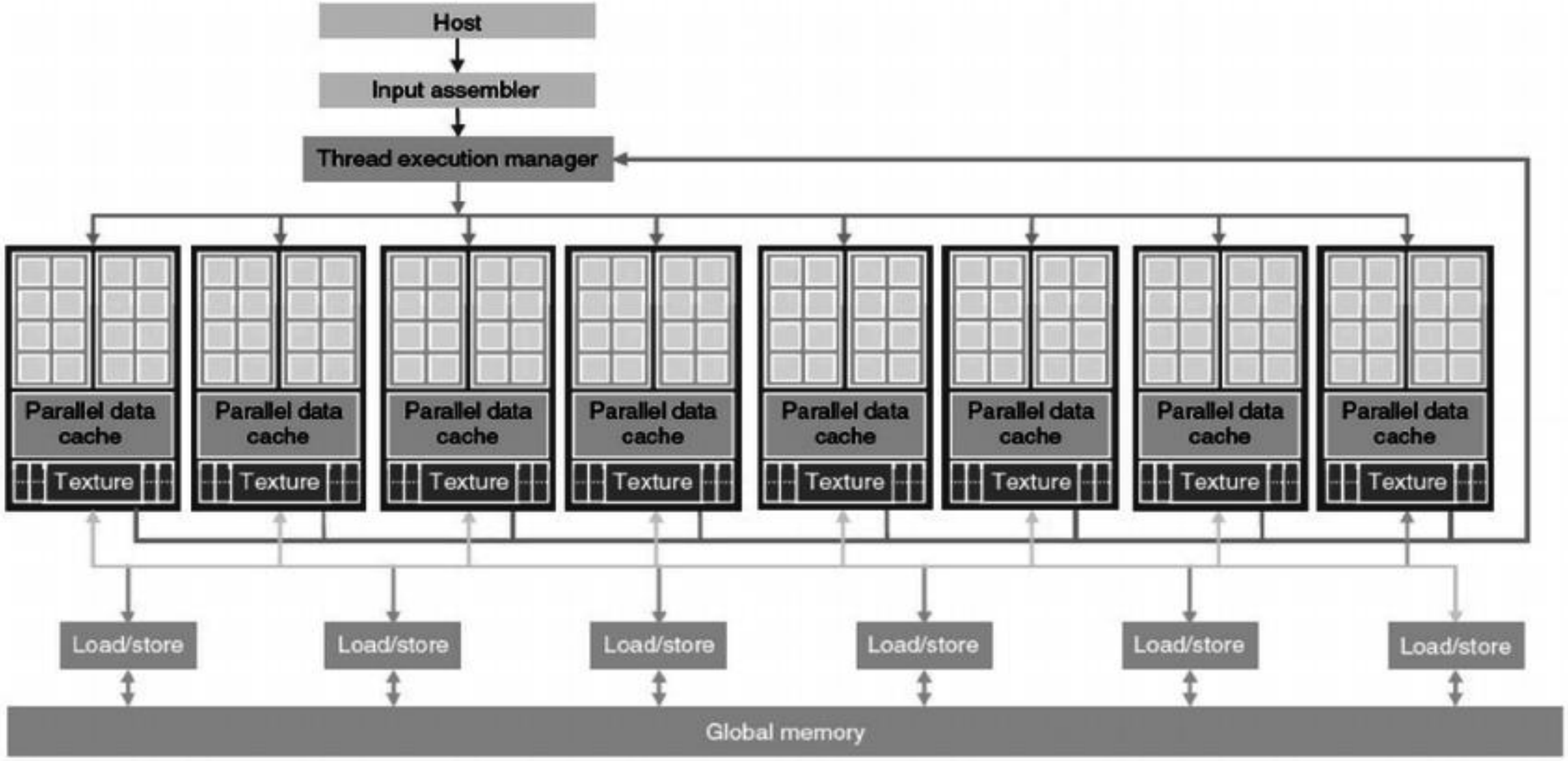
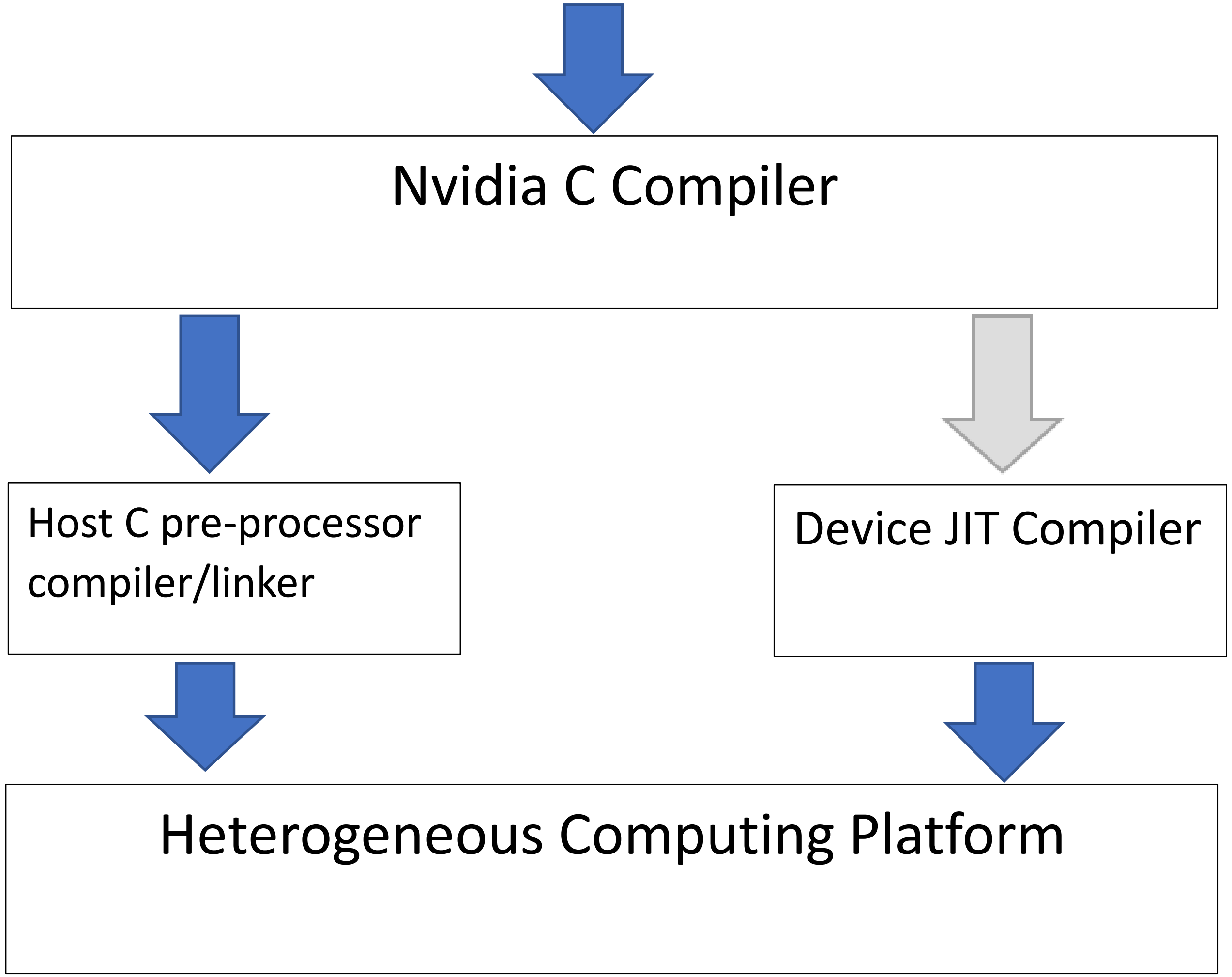


Figure 1.3 : GPU Architecture

GPU Execution

- A typical CUDA program has code intended for both the CPU and the GPU
- The NVCC or Nvidia C Compiler splits the Data intensive code from the program.
- The Host machine processes the less intensive code.
- The Device processes the more intensive code.
- Together it forms a heterogeneous computing platform.


Nvidia C Compiler

Host C pre-processor compiler/linker

Device JIT Compiler

Heterogeneous Computing Platform

CUDA Memories

- Following are the supported memory types on a CUDA supported GPU
  - R/W per-thread registers
  - R/W per-thread local memory
  - R/W per-block shared memory
  - R/W per-grid global memory
  - Read only per-grid constant memory.

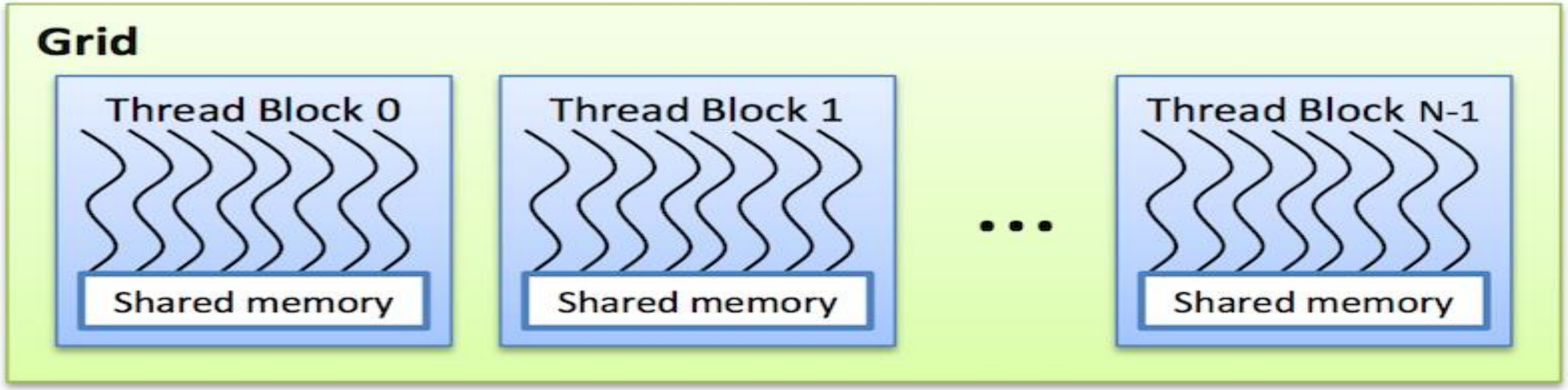


Figure 1.4: Thread Level Hierarchy

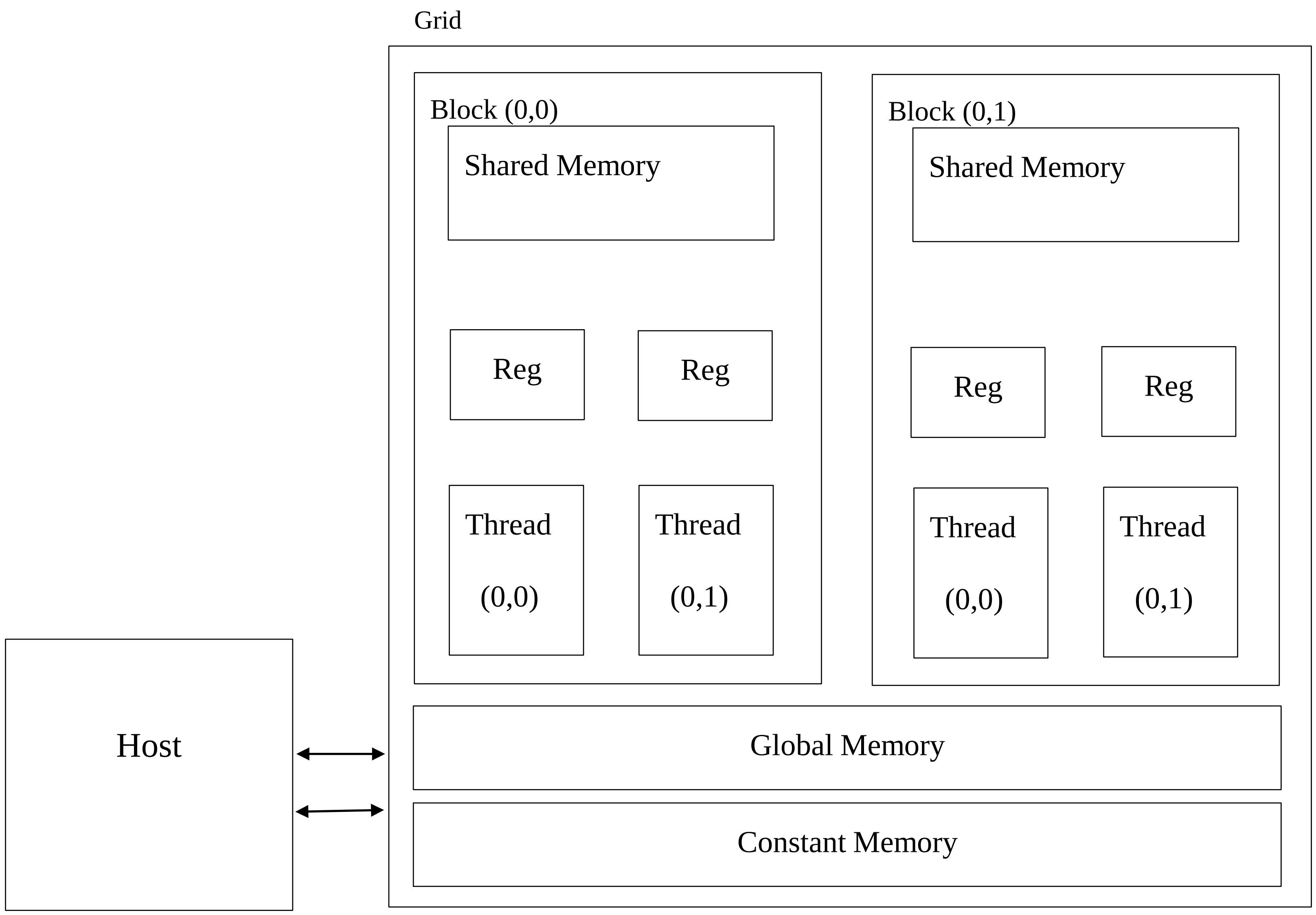


Figure 1.5: Memory Assignment

SILICON POWER

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 logics.

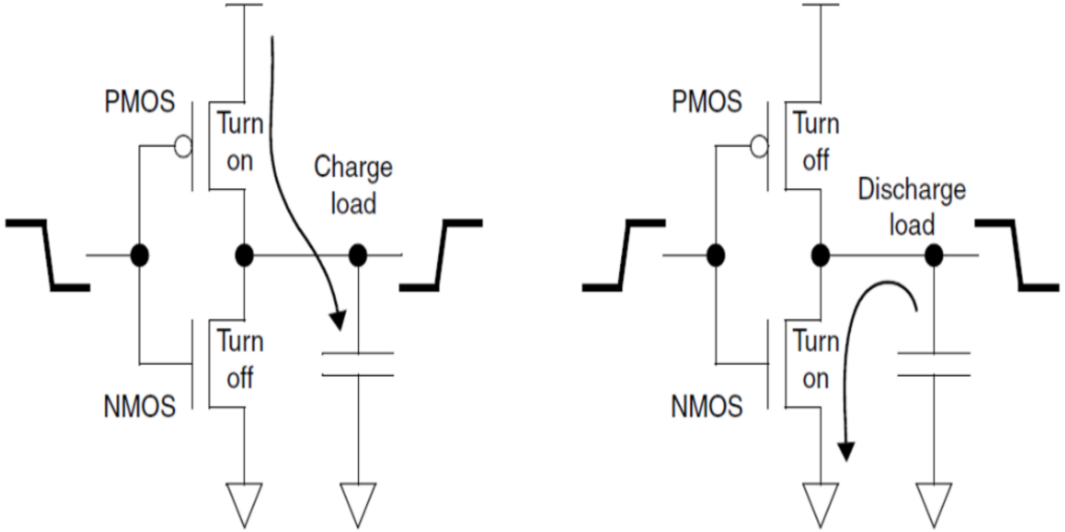


Figure 1.6: Logic transitions in an inverter

- 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 as shown in Figure 1.7 contributing to internal power.

$$P_{dyn} = \left(C_{eff}.V_{dd}{}^{2}.f_{clk}\right) + (t_{sc}.V_{dd}.I_{peak}.f_{clk}) \qquad (1.1)$$

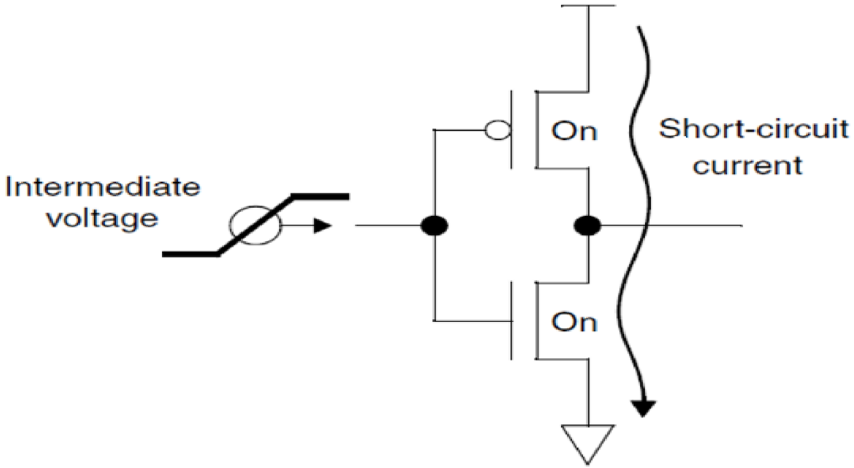


Figure 1.7: Short Circuit Current

Static (Leakage) Power

- Static power is the transistor leakage current whenever power is applied to the device. The main causes of leakage power as displayed in Figure 1.8 are
  - Reverse bias p-n junction diode leakage: Caused by minority carrier drift
  - Sub threshold leakage: Current which 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.

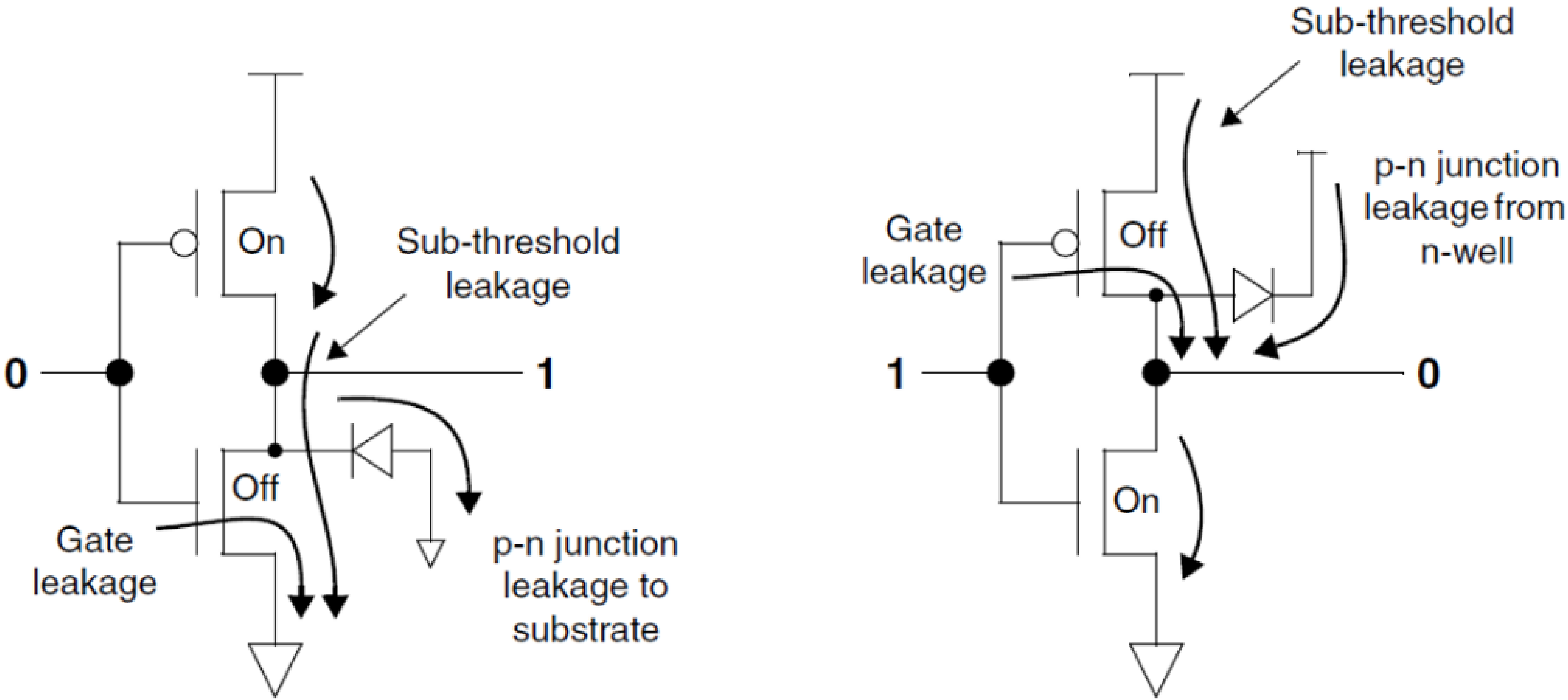


Figure 1.8: Static (Leakage) Power causes

RESULTS

Captured Data

- **Compute Metrics:** Metrics represent the counters that can be involved in a computation.
  - Number of additions.
  - Number of bytes transferred from the memory.
  - Number of instructions executed.
- **Power:** Core Voltage, Core Current, Memory Voltage, Memory Current.

Feature Selection

- Features represent the various groups of obtained data under the captured metrics.
- Reducing the feature count to minimize the complexity of the data, helps in efficient modelling.
- This can be achieved using largely available Machine Learning algorithms like PCA (or Principal Component Analysis) which helps eliminate the features from a set which do not contribute to large changes in the Target Variable (here, power) , i.e., feature prioritization.

Correlations in Data

- Multiple data points can be scaled versions of a data point.
- Such data points make data model complex and their removal will not affect the model in any way.

- Techniques to remove correlations:
  - Clustering.
  - Cosine Similarity.

Power Modelling

- Implemented Steps
  - Minimum number of features obtained.
  - Redundant data, representing similar data points removed.

1. **Total Power from running workload:** This is the power obtained from the GPU with operating workload.
2. **Computed Power from Statistical Models:** This is the computed power after Feature Selection and removing Correlations in data.
3. **Accuracy of Prediction**

- Implement Statistical Power model for energy consumed and obtain the convergence error.
  - $Error = Total\ Energy - Computed\ Energy$

References

[1] Siddhartha Sankar Mondal, “GPU: Power vs Performance”, Department of Information Technology, Uppsala Universitet, June 2014.

[2] S Sundar, M Panchatcharam, “GPU Basics”, Department of Electronics and Communication, Indian Institute of Technology, August 2014, Madras.

[3] 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.

[4] 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.

[5] 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.

[6] CUDA – Introduction to GPU, www.tutorialspoint.com/cuda

[7] GPU Fundamentals , https://slideshare/GPU

[8] 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.

[9] I. Paul, W. Huang, M. Arora et al., "Harmonia: Balancing compute and memory power in high-performance gpus," in *Proceedings of the 42nd Annual International Symposium on Computer Architecture, ser. ISCA '15*. New York, NY, USA: ACM, 2015, pp. 54–65.

[10] G. Wu, J. L. Greathouse, A. Lyashevsky et al., "Gpgpu performance and power estimation using machine learning," in *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*, Feb 2015, pp. 564–576.

[11] J. Coplin and M. Burtscher, "Energy, power, and performance characterization of gpgpu benchmark programs," in *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW),* May 2016, pp. 1190–1199.

[12] 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.

[13] R. Ge, R. Vogt, J. Majumder et al., "Effects of dynamic voltage and frequency scaling on a k20 gpu," in *2013 42nd International Conference on Parallel Processing, Oct 2013*, pp. 826–833.

- [14] J. Guerreiro, A. Ilic, N. Roma et al., “Multi-kernel auto-tuning on gpus: Performance and energy-aware optimization,” in *2015 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, March 2015, pp. 438–445.
- [15] 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.