Computer Power and Integrated Circuits

CCS5 – Computer Architecture and Organization

Agenda

- Power, Energy, and Integrated Circuits
- Quantitative Principles of Design and Measurement: *Modules and Integrated Circuits*

Power, Energy, and Integrated Circuits

Computer Power and Integrated Circuits

Power and Energy

- When discussing power for processors, it tends to have two meanings.
- The first is power in terms of **overall processing power** such as GHz.
- The second meaning of power is **power consumption** expressed in Watts.

Power and Energy

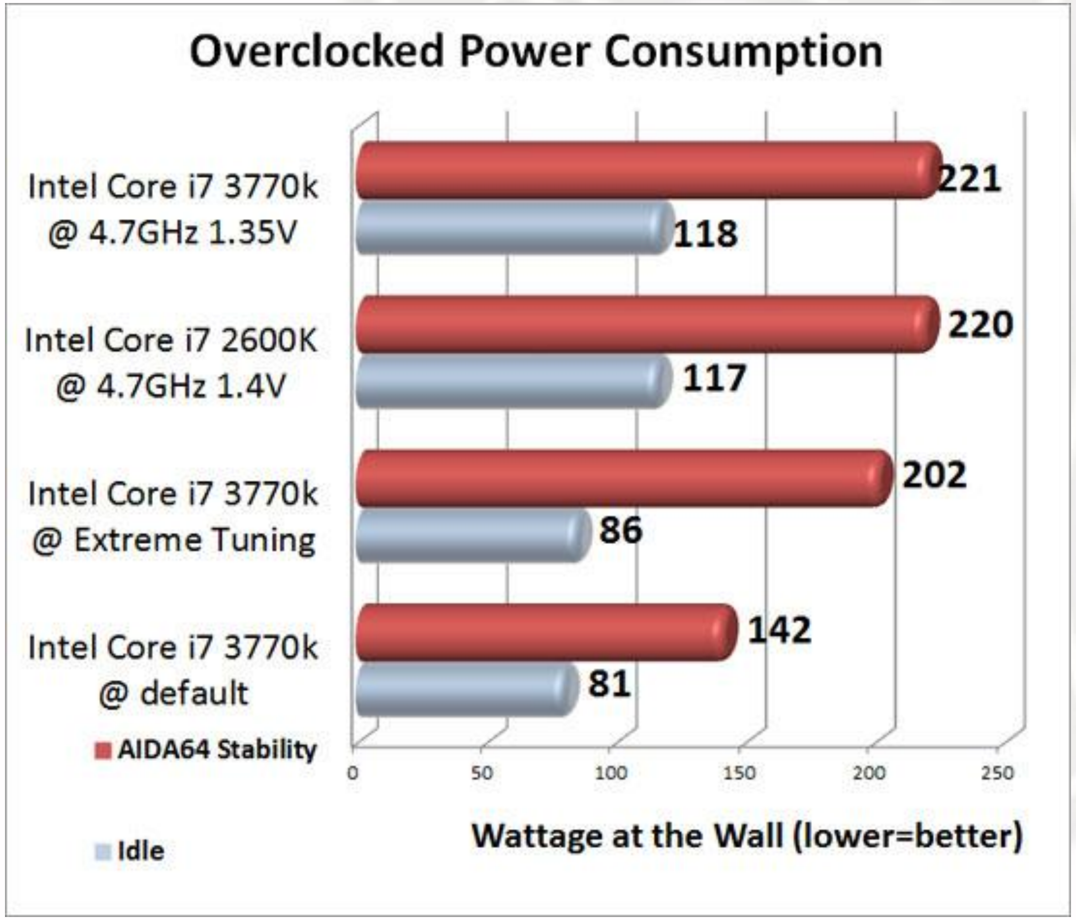
- Power can be represented using these equations:
  - ***Power*** = *Energy / Time (e.g., Watt)*
  - ***Energy*** = *Voltage*
  - ***Watt*** = *Joules / Second*
- The focus of managing power for processors is getting power in and getting power out.

Power and Energy

- There are 3 commonly used metrics for measuring power consumption for a given processor:
  - ***Peak Power:*** *This refers to the maximum power that a processor can handle.*
  - ***Thermal Design Power (TDP):*** *This represents the sustained power consumption of a given processer.*
  - ***Average Power:*** *This refers to the mean power used between surges of activity and inactivity.*

Power and Energy

- Take note that **frequency or clock speed/clock rate** greatly affects power consumption.



Power and Energy

- The limit of the frequency of a given processor depends on many factors.
- These factors can be the quality of the silicon, the cooling solution and the power being fed into it.
- **Overclocking:** This is a process of increasing the frequency of a CPU beyond the recommended speed.

Dynamic Power and Energy

- **Dynamic Power** refers to the power dissipated in circuits and contributes to peak power.
- It is represented using the given equation:
    - *½ Capacitive Load x Voltage*^{2} *x Frequency Switched*

Dynamic Power and Energy

- **Dynamic Energy** refers to the energy stored in the capacitor by virtue of it being charged to *V* volts.
- It is represented using the given equation:
  - *Transistor switch from 0 -> 1 or 1 -> 0*
  - *½ Capacitive Load x Voltage*^{2}

Dynamic Power and Energy

- When reducing the clock rate for a given system, it ends up reducing the **energy consumption** of that given system.
- When determining the dynamic power and energy used by a system, we start with **dynamic energy**.

Dynamic Power and Energy

Let us look at some examples of how dynamic energy and dynamic power are used:

- Microprocessors these days are digitally designed to have voltages that can be adjusted depending on various situations. For instance, there are processors that experience frequencies reduced up to 20% when the voltage is reduced by 25%.

Dynamic Power and Energy

Let us look at the effect on Dynamic Energy:

Dynamic Energy $= \text{Voltage}^2$

$= 1 – 0.25 = 0.75$

$= 0.75^2$

$= 56.25\%$

The processor has reduced energy to about 56.25% of the original.

Dynamic Power and Energy

Let us look at the effect on Dynamic Power:

Dynamic Power = Voltage^2 * Frequency Switched

= 0.5625 * 0.8

= 45%

The processor has reduced power to about 45% of the original.

Power

- **Processors** are becoming faster over time but also consume more power than ever before.
- Cooling modern processors requires more sophisticated cooling solutions to keep them cool.
    - *Static Power Consumption: Current (Static) x Voltage*

Power

- The static power consumption scales with number transistors.
- This value is reduced through **power gating**.
- This is a technique used to reduce power consumption, by shutting off current blocks of the circuit not in use.

Quantitative Principles of Design and Measurement: Modules and Integrated Circuits

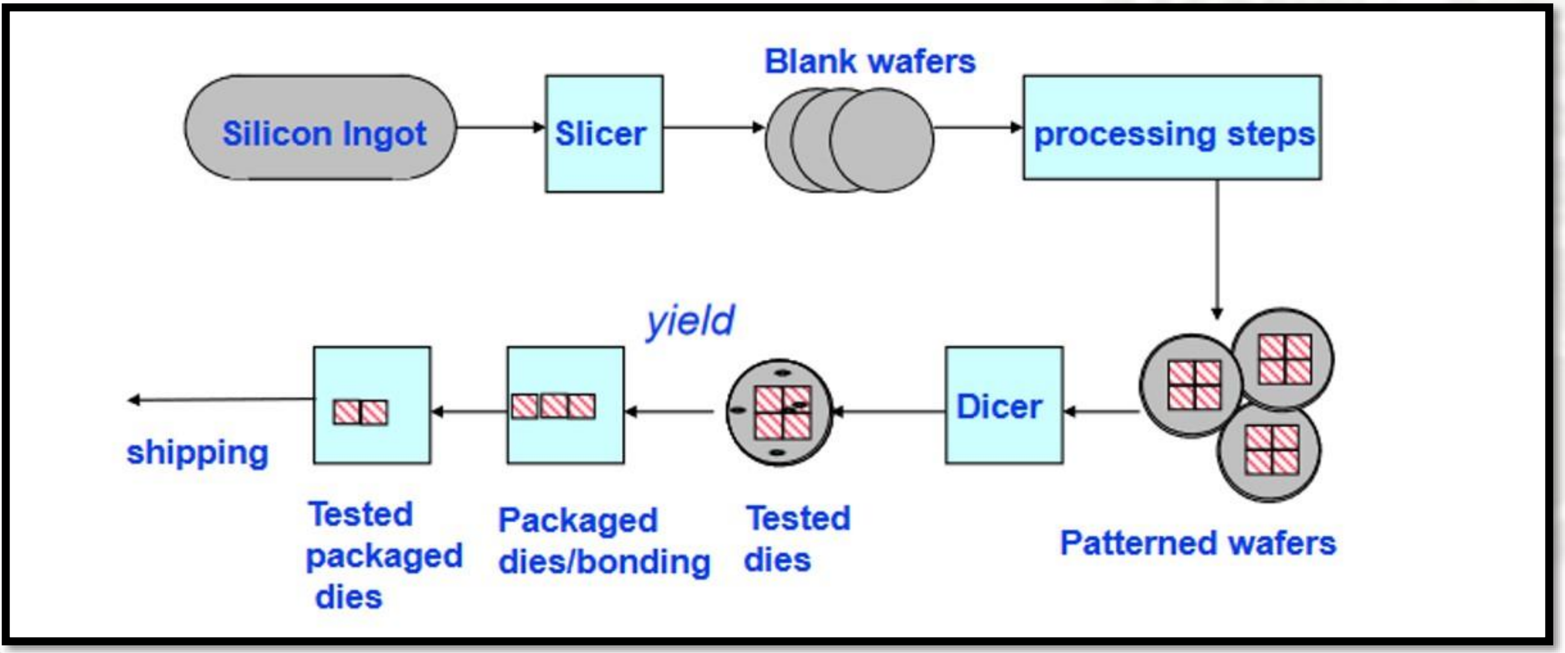
Computer Power and Integrated Circuits

Integrated Circuit Cost

- When manufacturing and selling **integrated circuits (IC)**, a big factor that is looked at is **price**.
- The manufacturers will price the ICs depending on their output and whether a buyer will purchase the ICs at cost.
- Cost for a given IC is not fixed and will depend on the wafers that are used to create them.

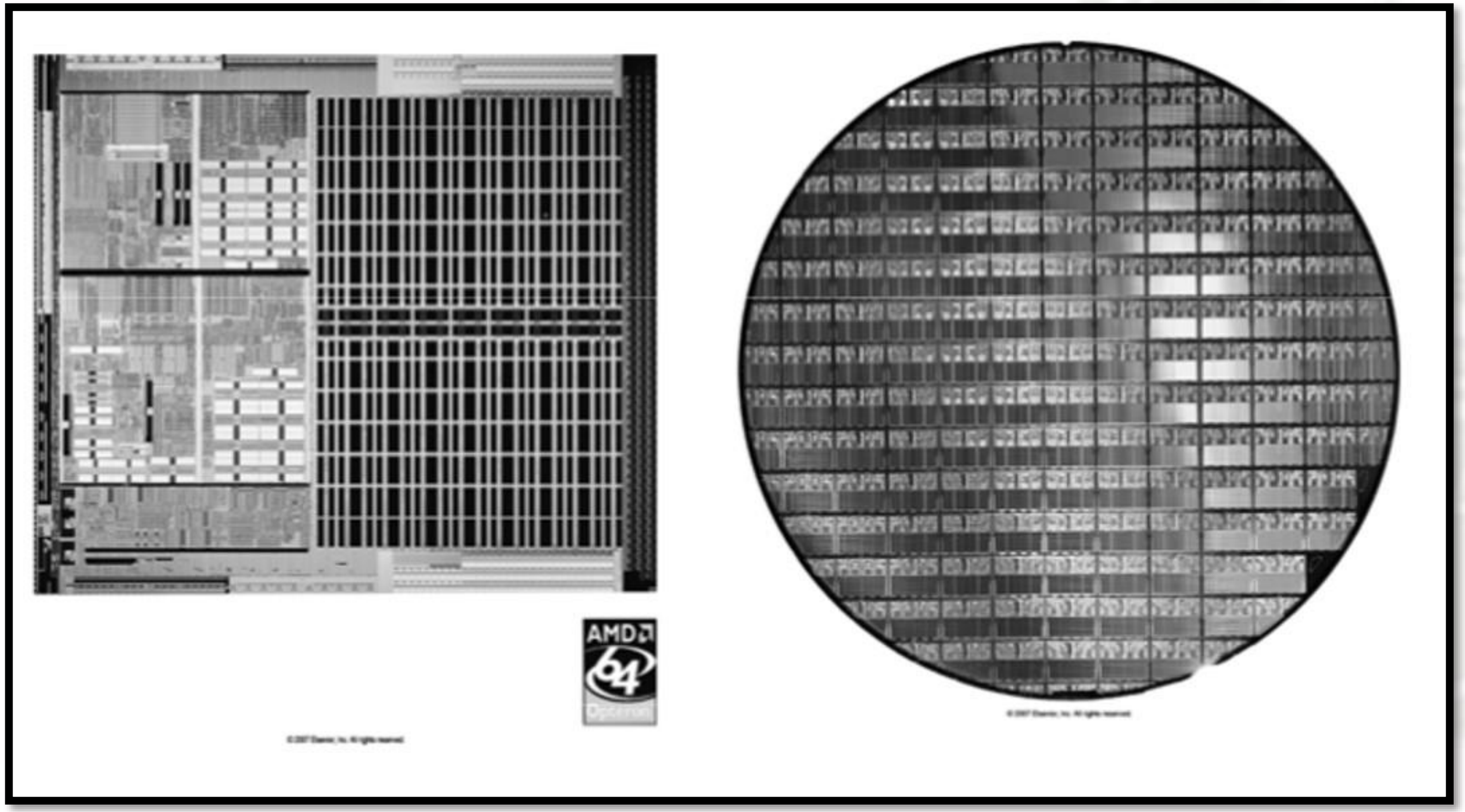
Integrated Circuit Cost

- **Wafers** are a thin slice of semiconductor, used for creating integrated circuits.
- Shown below is a diagram of the IC creation process:



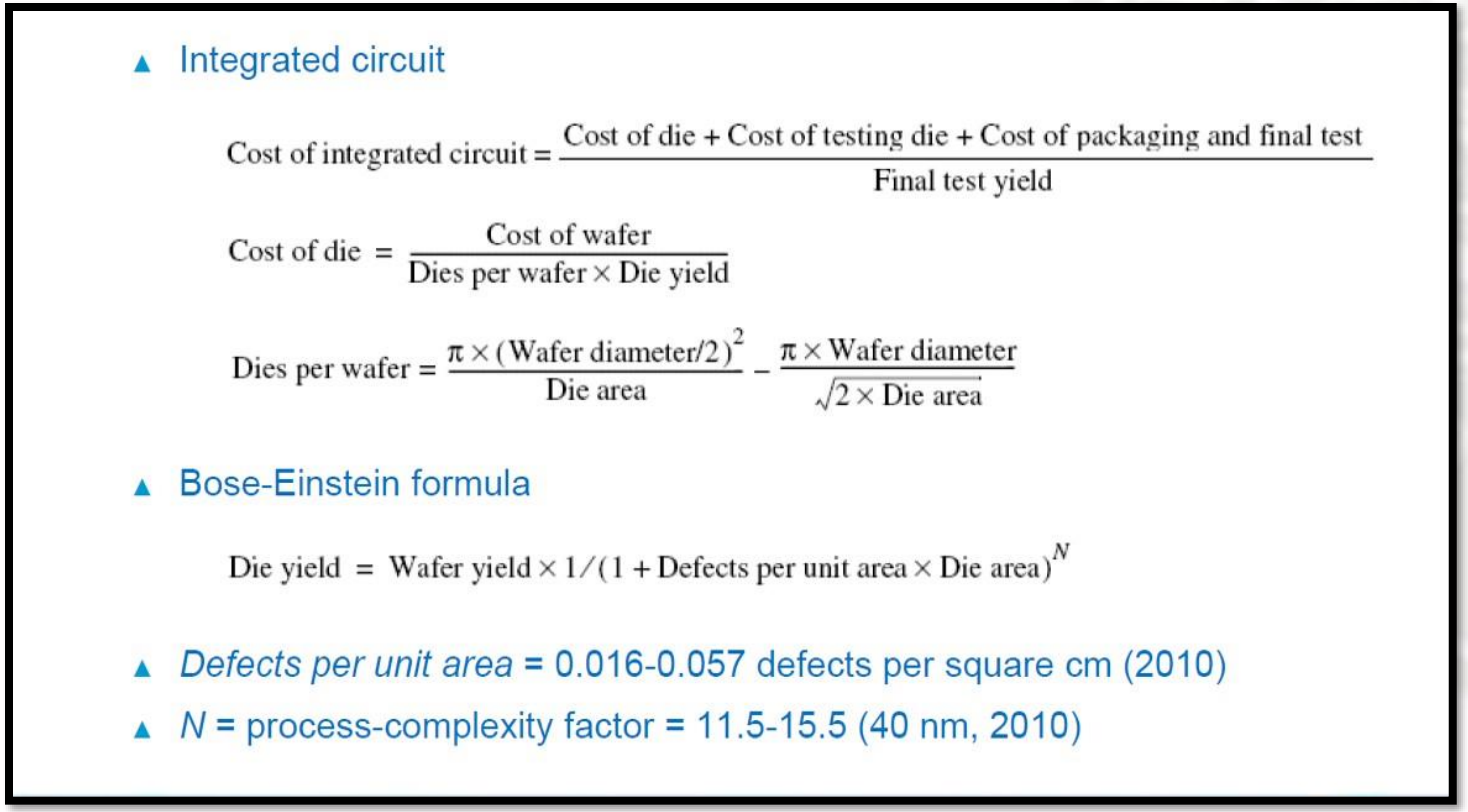
Integrated Circuit Cost

- Here is a picture of how a wafer typically looks like:



Integrated Circuit Cost

- There are also many ways to compute for the given cost of an IC, as shown below:



- Integrated circuit

$$\text{Cost of integrated circuit} = \frac{\text{Cost of die} + \text{Cost of testing die} + \text{Cost of packaging and final test}}{\text{Final test yield}}$$

$$\text{Cost of die} = \frac{\text{Cost of wafer}}{\text{Dies per wafer} \times \text{Die yield}}$$

$$\text{Dies per wafer} = \frac{\pi \times (\text{Wafer diameter}/2)^2}{\text{Die area}} - \frac{\pi \times \text{Wafer diameter}}{\sqrt{2 \times \text{Die area}}}$$

- Bose-Einstein formula

$$\text{Die yield} = \text{Wafer yield} \times 1/(1 + \text{Defects per unit area} \times \text{Die area})^N$$

- *Defects per unit area* = 0.016-0.057 defects per square cm (2010)
- *N* = process-complexity factor = 11.5-15.5 (40 nm, 2010)

Integrated Circuit Cost

Let us try determining the cost for a given wafer using the equations shown in the previous slide:

- Let us say we would like to find the maximum number of dies per 30 cm diameter for a wafer for a die that is 1.5 cm on a side.

Integrated Circuit Cost

Here are our solutions:

Wafer Diameter = 30 cm

Die Radius = 1.5 cm

Die Area = 1.5 cm^2 = 2.25 cm

Dies per wafer = ((π * (Wafer Diameter/2)^2)/Die Area) – ((π * Wafter Diameter)/ $\sqrt{}$(2 * Die Area))

= ((π * (30/2)^2)/2.25) – ((π * 30)/ $\sqrt{}$(2 * 2.25))

= 269.730 or about 269 dies/wafer

Module Dependability and Reliability

- Another important factor to keep in mind with regards to ICs and computer hardware is their **reliability**.
- **Mean Time to Failure (MTTF):** Measures the length of time a hardware device is expected to last in operation.
    - *This is mainly used* ***for non-repairable products***
    - *It is the average of fail times*
    - *It is the reciprocal of* ***Failure Rate***
    - ***MTTF*** *= (Operating Time (Cycles) / Number of Failures)*

Module Dependability and Reliability

- **Mean Time Between Failure (MTBF):** Measures the elapsed time between system breakdowns.
  - *This is mainly used* ***for repairable products***
  - *It is also the average of fail times*
  - *It is also the reciprocal of* ***Failure Rate***
  - ***MTBF*** *= (Operating Time (Cycles) / Number of Failures)*

Module Dependability and Reliability

- **Failure Rate:** Measures how often a system fails.

  Failure Rate *= (Number of Failures / Operating Time (Cycles))*

  = Failures per Hour

  Operating Time (Cycles)

  *= Total Units Tested * Expected Operating Time / Sum of Non-Operating Time of Failed Units*

Module Dependability and Reliability

Let us see how we can compute for the reliability and failure rate for a given device:

- Let us say we have a system that has the following parts and individual MTTF, let us try to compute the entire systems' MTTF. We generalize that every failure is independent.

Component	Quantity	Expected Operating Time	Failure Time
CPU	1	45000	n/a
GPU	1	65000	38000
SSD	1	85000	67000
RAM	1	35000	21000
PSU	1	95000	n/a

Module Dependability and Reliability

- To determine our systems' overall reliability, we start by computing for the failure rate:

Failure Rate = *(Number of Failures / Operating Time (Cycles))*

= *3 / ((45000 + 95000) +(38000 + 67000 + 21000))*

= *3 / (140000 + 126000)*

= *3 / 266000*

= 1.13 x $\boldsymbol{10^{-5}}$

= 0.0000113 Failures Per Hour

Module Dependability and Reliability

- The failure rate result does not tell us much initially, but we can use the reciprocal value of MTTF to get more data:

MTTF *= (Operating Time (Cycles) / Number of Failures)*

= ((45000 + 95000) +(38000 + 67000 + 21000)) / 3

= (140000 + 126000) / 3

= 266000 / 3

= 88666.67 Hours to Failure

= ~10 Years to Failure

Measuring Performance of Integrated Circuits

- There are many methods that can be used to help measure the performance of a given IC.
- The most common metric used for users is **execution time or response time.**
- This refers to the time between the submission of a request until the request is received.

Measuring Performance of Integrated Circuits

- For administrators, it tends to be **throughput**, which is the amount of data that passes through a system.
- For ICs, we make use of **execution time**:
    - *n = Execution time of B / Execution time of A*

Measuring Performance of Integrated Circuits

- Take note that **performance** is inversely proportional to execution time:
  - *n = Performance of A / Performance of B where Performance = 1 / Execution time*
  - *If given that Processor A is faster than Processor B, that means execution time of A is less than Execution time of B.*
  - ***Therefore, performance of Processor A is greater than that of performance of Processor B.***

Measuring Performance of Integrated Circuits

- Execution time can be defined in different ways:
  - ***Wall-clock time:*** *The latency to complete a task, which includes disk accesses, memory accesses, I/O and OS overhead.*
  - ***CPU time:*** *Refers to the time where the CPU is computing, which excludes time waiting for I/O and executing other programs.*

Measuring Performance of Integrated Circuits

- There are also other methods for benchmarking a given processor:
  - ***Synthetic Benchmarking:*** *This is the least accurate type of benchmarking and mimics the process and resource requirements of real programs.*
  - ***Toy Programs:*** *These are small applications (10-100 lines of code) or programs that run in different machines as benchmarks.*

Measuring Performance of Integrated Circuits

- These are a few more methods for benchmarking a given processor:
    - ***Kernel Programs:*** *These are parts of an actual application that are run to isolate the performance of individual features.*
    - ***Benchmark Suites:*** *These are sets of applications or programs to set up the benchmarking of software and hardware.*

Principles of Computer Design

Parallel Processing:

- Describes a class of techniques which enables the system to achieve simultaneous data-processing tasks.
- A parallel system can carry out simultaneous data-processing to achieve faster execution time.
- The purpose of this is to enhance the computer processing capability and increase throughput.

Principles of Computer Design

- **Locality Principle:** Programs tend to reuse data and instructions near those they have used recently.
- **Temporal Locality:** This means recently referenced items are likely to be referenced soon.
- **Spatial locality:** This means that items with nearby addresses tend to be referenced close together in time.

Principles of Computer Design

Amdahl's Law:

- Presented by **Gene Amdahl** in 1967 and deals with parallel processing for a given system.
- It is a formula used to find the **performance improvement** from a parallel computing system.
- This has a limiting factor on program speedup such that adding more processors may **not** speed up a program.

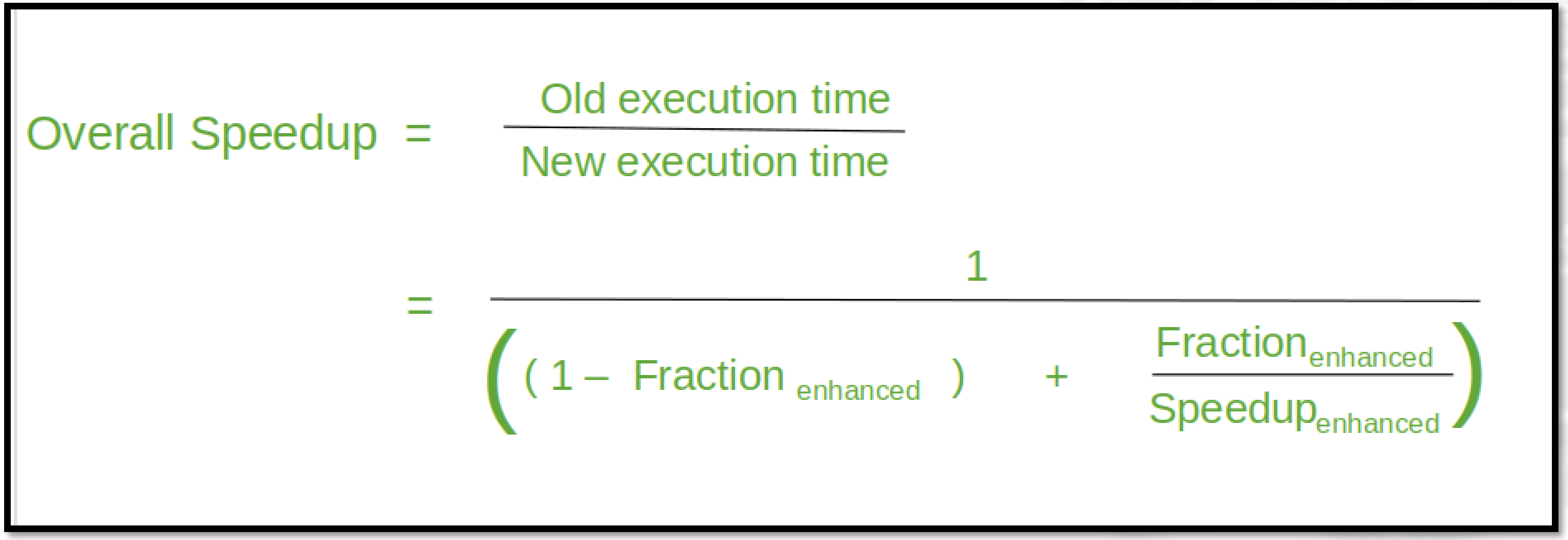
Principles of Computer Design

Amdahl’s Law:

- It presents a formula that gives the **theoretical speedup** in latency of the execution of tasks at a fixed workload.
- Amdahl’s Law states that you always go to the “common case”.
- This refers to the maximum improvement possible by improving the most frequently used part of the system.

Principles of Computer Design

- The formula for **Amdahl's Law** dictates how much gain you will get from improving one component of the system:



$$\text{Overall Speedup} = \frac{\text{Old execution time}}{\text{New execution time}}$$

$$= \frac{1}{\left((1 - \text{Fraction}_{\text{enhanced}}) + \dfrac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}} \right)}$$

Principles of Computer Design

- Let us look at the parts of the formula used for Amdahl’s Law:
  - ***Speedup***: *This is defined as the ratio of performance for the entire task.*
  - ***Speedup enhanced:*** *This can be simply thought of as the improvement on the specific component.*
  - ***Fraction enhanced:*** *This can be thought of as how often this component is used.*

Principles of Computer Design

Let us have a look at an example of how Amdahl's Law is used:

- Let's assume that you're currently considering to improve the processor your company uses for providing cloud services. You have computationally tested that a processor made by an independent chip maker B.E.N. is 10x faster than your original processor. Your manager asks you to give an accurate computation of overall speed up based on Amdahl's Law for him to consider the upgrade. Given that your original processor has 40% constant usage and 60% I/O waiting time, what will your answer be?

Principles of Computer Design

Let us see how we solve this problem using Amdahl's Law:

Speedup Enhanced = 10
Fraction enhanced = 0.4
Overall Speedup = 1/((1-0.4) + 0.4/10)
= 1/(0.6 + 0.04)
= 1/0.64
= 1.56

The new processor provides an overall speedup of 1.56x