Lecture 1: class slides

CE/CZ 3001:
Advanced Computer Architecture

(Module 1: Introduction and Background)

Dr Smitha K. G.
School of Computer Science
And Engineering

Pre Video- Summary

- 5- basic hardware of computers
- Von Neumann and Harvard Architecture
- Need of high performance computing
- Design goals and constraints
  - Performance and Power consumption

What is performance?

- Performance indicator- $Execution\ time$
- Execution time (CPU time) is the time to execute a program
  - Less the execution time → better is the performance
  - $Performance = \frac{1}{Execution\ Time}$

Example 1

Consider a program comprising of 100 instructions which runs in two different machines M_1 and M_2. Each instruction takes 2 clock cycles in M_1 and 1 clock cycle in M_2. The clock period of M_1 and M_2 are 20 nanosecond (ns) and 30 ns, respectively. Find the execution times of the program in both machines. Which machine has better performance?

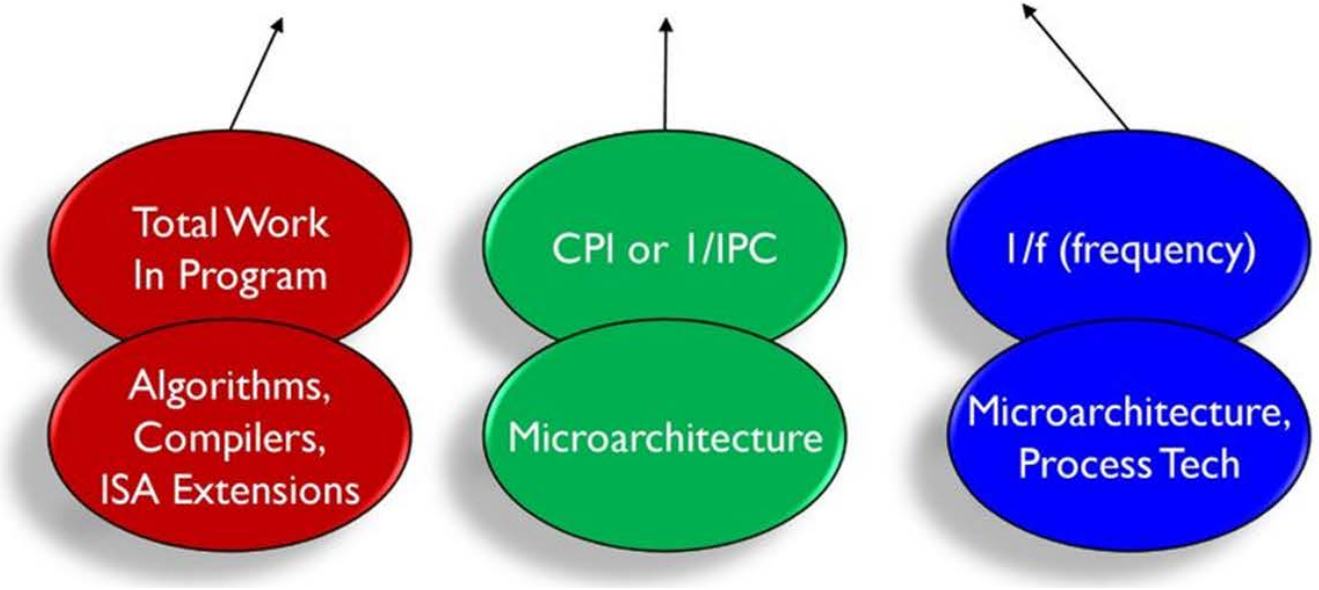
Execution time on M_1 = 100 x 2 x 20 ns = 4000 ns
Execution time on M_2 = 100 x 1 x 30 ns = 3000 ns

$$Performance = \frac{1}{Execution\ Time}$$

Hence Machine M_2 has better performance

The *Iron Law* of Processor Performance

$$\frac{Time}{Program} = \frac{Instructions}{Program} \times \frac{Cycles}{Instruction} \times \frac{Time}{Cycle}$$



$$Execution\ time = IC \times CPI \times T$$

Decrease in one may lead to increase in other two.

Example 2

A Processor executed a program comprising of 3 billion instructions and processor spends two cycles on each instruction. The processor clock is 3Ghz. Find the execution time?

Exe time= (3*10^9 *2)/(3*10^9)= 2s

Example 3

A program has 50 million instructions running at a frequency of 4MhZ

- 10 million branches(CPI= 4)
- 15 million loads(CPI=2)
- 5 million store (CPI=3)
- Rest Add(CPI=1)

Find the execution time

(10 *10^6*4 +15*10^6*2+5*10^6*3+20*10^6*1)/(4*10^6)

(40+30+15+20)/4= 26.25s

What is speedup?

Case A: Consider two computers: computer-A and computer-B speedup of computer-A over computer-B can be computed as

$$Speedup = \frac{Perf_A}{Perf_B} = \frac{Time_B}{Time_A}$$

Case B: Suppose computer-B is an enhanced version of computer-A, achieved by some specific performance enhancement technique, then the speedup achieved due to the enhancement technique can be expressed as

$$Speedup = \frac{T_{unenhanced}}{T_{enhanced}} = \frac{T_{original}}{T_{enhanced}}$$

Amdahl's law: speedup via parallelism is limited by that component of an application which cannot be enhanced (the sequential component)

$$\text{Speed up} = \frac{Time\ original}{T\ enhanced} = \frac{1}{[(1-E)+E/S]}$$

How to prove that limit of parallelism?

Example: accountants adding the total amount of invoices

Add the total amount in m=1024 invoices as quickly as possible by n accountants.

Find the time (T) required to add these numbers?

Serial Computation

The following assumptions can be used:

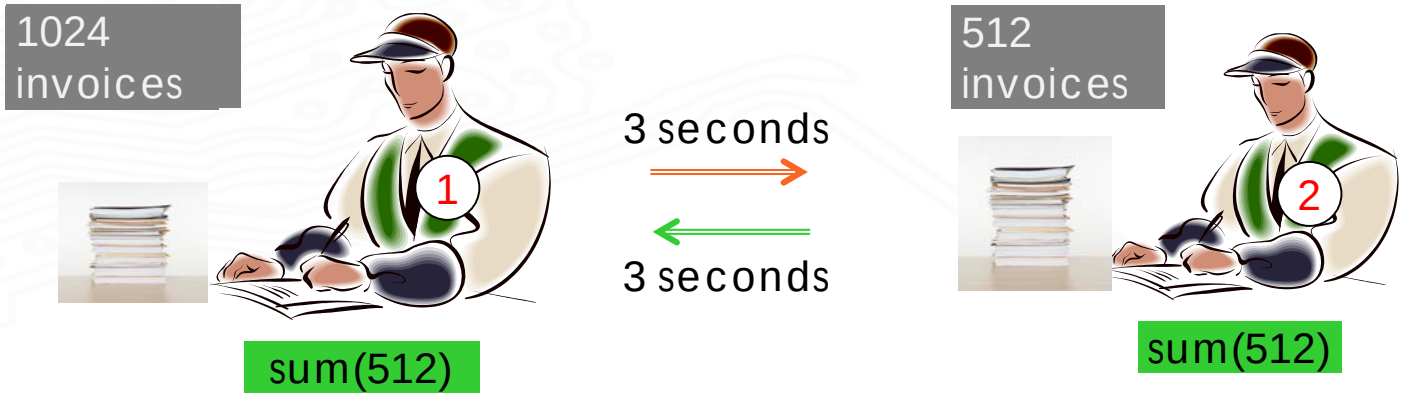
1. A stack of 1024 invoices, is initially given to accountant #1. The total amount of all the invoices are to be added.
2. Any accountant takes 1 second to add two numbers. (Counting time = 1 sec)
3. An accountant will require 3 seconds to send any number of invoices or computed results to any other accountant. (Transfer time = 3sec)
4. Accountant #1 computes the final result.

case-1: m=1024, *number of accountants (n)*=1:
***T*=1023 seconds**
only one accountant: this is an *example of serial computation.*

Distributed Parallel Computation (Part 1/2)

case-2: *m=1024*, *n*=2 (two accountants)

3 sec to transfer the invoices to Acc_2
511 sec for adding 512 invoices by Acc_1 and Acc_2
3 sec to transfer of partial results back to Acc_1
1 sec for sum (512) of Acc_1 and Acc_2



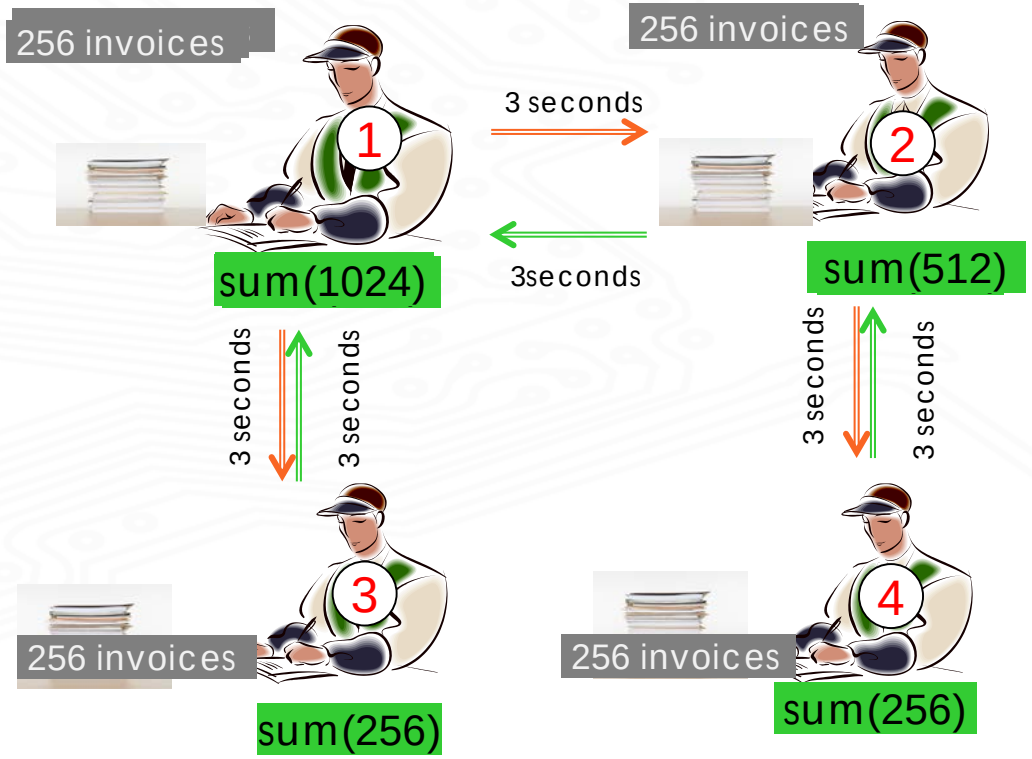
511 × 1 seconds to add 512 numbers

Time to complete the task,

$$T = 3 \times 1 + 511 \times 1 + 3 \times 1 + 1 \times 1 = 3 + 511 + 3 + 1 = 518\, seconds$$

Distributed Parallel Computation (Part 2/2)

Case-4: *m=1024*, *n*=4 (four accountants)



3 sec to transfer the invoices to Acc_2

3 sec to transfer the invoices to Acc_3 and Acc_4

255 sec for adding 256 invoices by all Accountants

3 sec to transfer of partial results back to Acc_1 and Acc_2

1 sec for sum(256) by Acc_1 and Acc_2

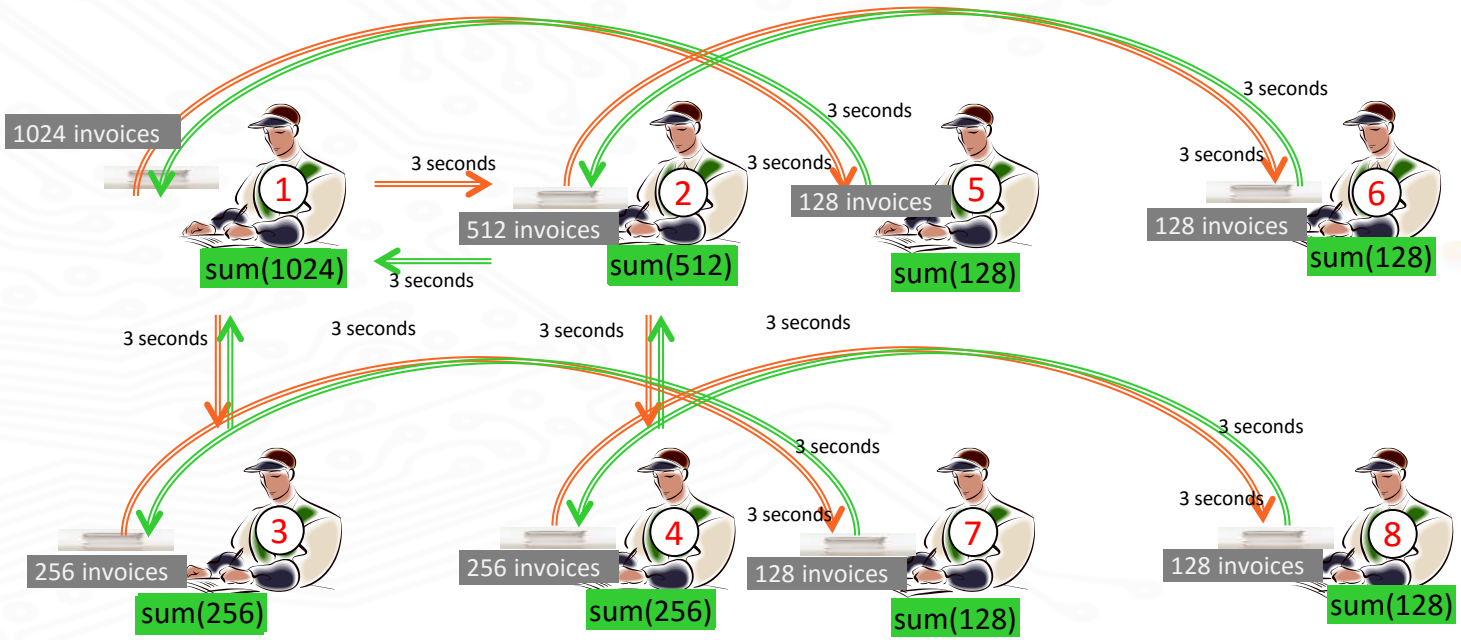
3 sec to transfer of partial results back to Acc_1

1 sec for sum(512) by Acc_1

Time to complete the task,

$$T = 3 \times 2 + 255 \times 1 + 3 \times 2 + 1 \times 2 = 6 + 255 + 6 + 2 = 269\ seconds$$

Case-8: *m=1024*, *n*=8 (eight accountants)



127*1 seconds to add 128 numbers
3x3=9 second to
transfer the invoices
3x3=9 sec to transfer
partial results
1 sec for sum(256)
1 sec for sum(512)
1 sec for sum(1024)

Time to complete the task,

$$T = 3 \times 3 + 127 \times 1 + 3 \times 3 + 1 \times 3 = 9 + 127 + 9 + 3 = 148\ seconds$$

Amdhals law

Amdahls law: speedup via parallelism is limited by that component of an application which cannot be enhanced (the sequential component)

4 stages- 16 people

=4 x 3 + 4 x 3+ (1024/16 -1) x 1+ 4x1= 91

5 stages- 32 people

=5 x 3 + 5 x 3+ (1024/32 -1) x 1+ 5x1= 66

6 stages- 64 people

=6 x 3 + 6 x 3+ (1024/64 -1) x 1+ 6x1= 57

7 stages- 128 people

=7 x 3 + 7 x 3+ (1024/128 -1) x 1+ 7x1= 56

8 stages- 256 people

=8 x 3 + 8 x 3+ (1024/256 -1) x 1+ 8x1= 59

No use in increasing the stages beyond 7 as per the performance calculation (limit mentioned by Amdhals law)

Generalized exp.=2(Number of stages (k) x communication time) +(k x addition time)+ (total value/2^K -1) x addition time

Example 4

A processor is running a program which has 50 billion instructions and the clock frequency of the processor is 2Ghz. The CPI and % of instruction types in program are given in the following table. Calculate the overall speed up when CPI of branch is improved from 4 to 2.

Instruction Type	% of instruction types in program	CPI
R-type	40	1
branch	20	4
Load	30	2
Store	10	3

Speed up = $\frac{Time\ original}{T\ enhanced}$

Execution time before enhancement=
(0.4*1+0.2*4+0.3*2+0.1*3)*50 * 10^9/(2*10^9)
(0.4+0.8+0.6+0.3)*25= 52.5

Execution time after enhancement=
(0.4*1+0.2*2+0.3*2+0.1*3)*50 * 10^9/(2*10^9)
(0.4+0.4+0.6+0.3)*25= 42.5

Speed up=52.5/42.5=1.24

Summary

- 5- basic hardware of computers
- Von Neumann and Harvard Architecture
- Need of high performance computing
- Design goals and constraints
  - Performance
    - **Iron law**
    - **Speed up using Amdahl's law**

Lecture 2: class slides

CE/CZ 3001:
Advanced Computer Architecture

(Module 1: Introduction and Background)

Performance enhancement and Power dissipation

Dr Smitha K. G.
School of Computer Science
And Engineering

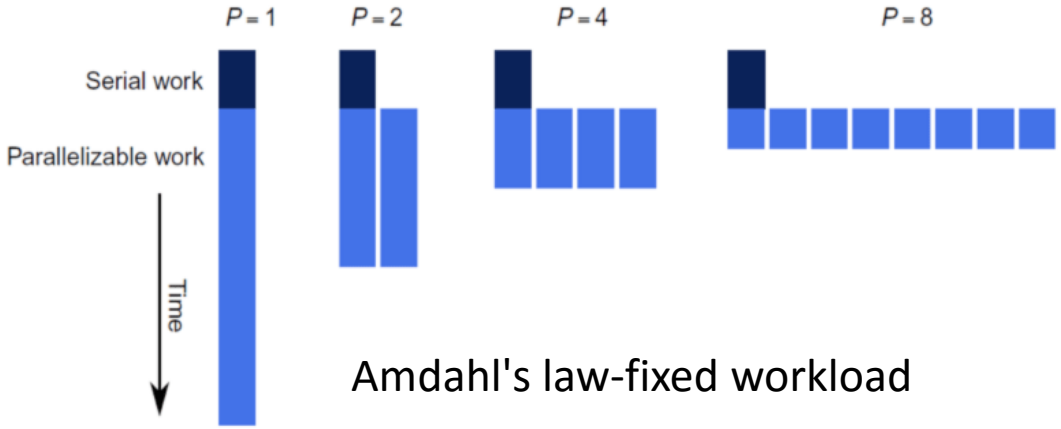
Pre Video- Summary

- Performance enhancement
  - Gustafson's Law
  - Other Performance metrics
    - MIPS and MFLOPS
- Power dissipation in processors
- Power metrics
- Low-power design techniques

Embarrassingly Parallel Problems

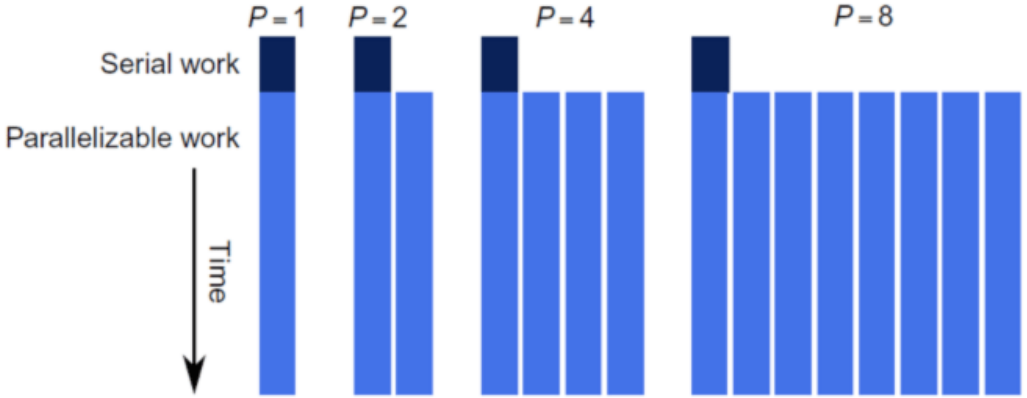
- Amdahl's law states a limit for parallelism
- Targets **embarrassingly parallel problems**

Gustafson's law



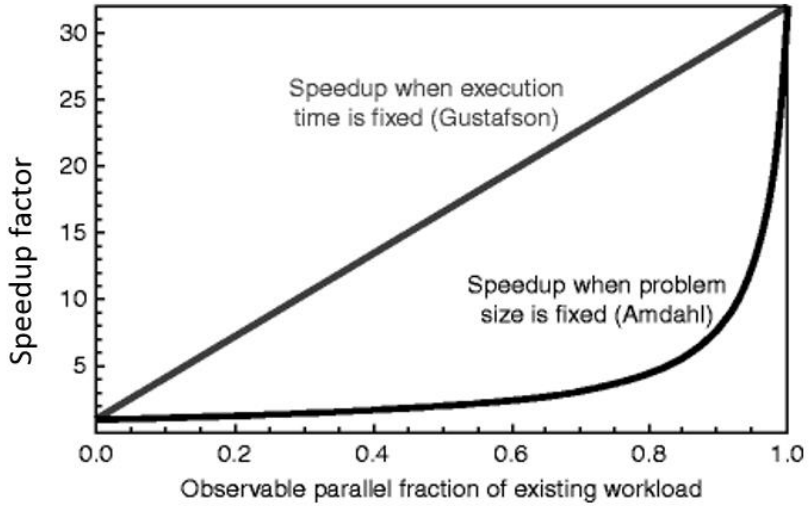
Amdahl's law-fixed workload

Gustafson's law-fixed time



Gustafson's Law

Instead of running the same size problem for all N, we can also consider running larger problems with better code or greater resources, which leads to Gustafson's law



$$Speedup(U,n) = \frac{T_{original}}{T_{enhanced}} = \frac{T_s + n.T_p}{T_s + T_p} = n - U(n-1)$$

Application in everyday computer systems

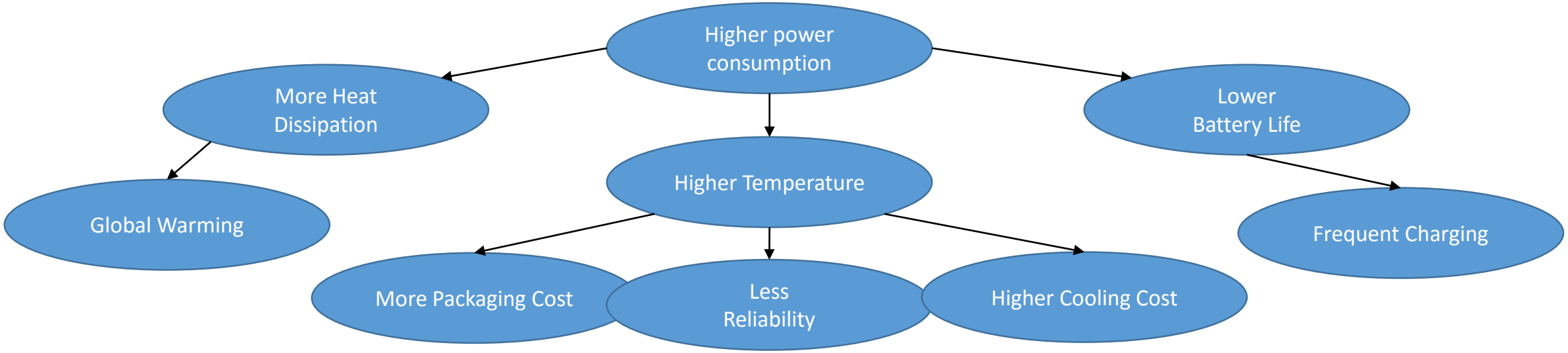
Amdahl's Law reveals a limitation in, for example,

- The ability of multiple cores to reduce the time it takes for a computer to boot to its operating system and be ready for use.
- Assuming the boot process was mostly parallel, quadrupling computing power on a system that took one minute to load might reduce the boot time to just over fifteen seconds.
- But greater and greater parallelization would eventually fail to make bootup go any faster, if any part of the boot process were inherently sequential.

Gustafson's law argues that

- A fourfold increase in computing power would instead lead to a similar increase in expectations of what the system will be capable of.
- If the one-minute load time is acceptable to most users, then that is a starting point from which to increase the features and functions of the system.
- The time taken to boot to the operating system will be the same, i.e. one minute, but the new system would include more graphical or user-friendly features.

Power dissipation in processors, power metrics, and low-power design techniques

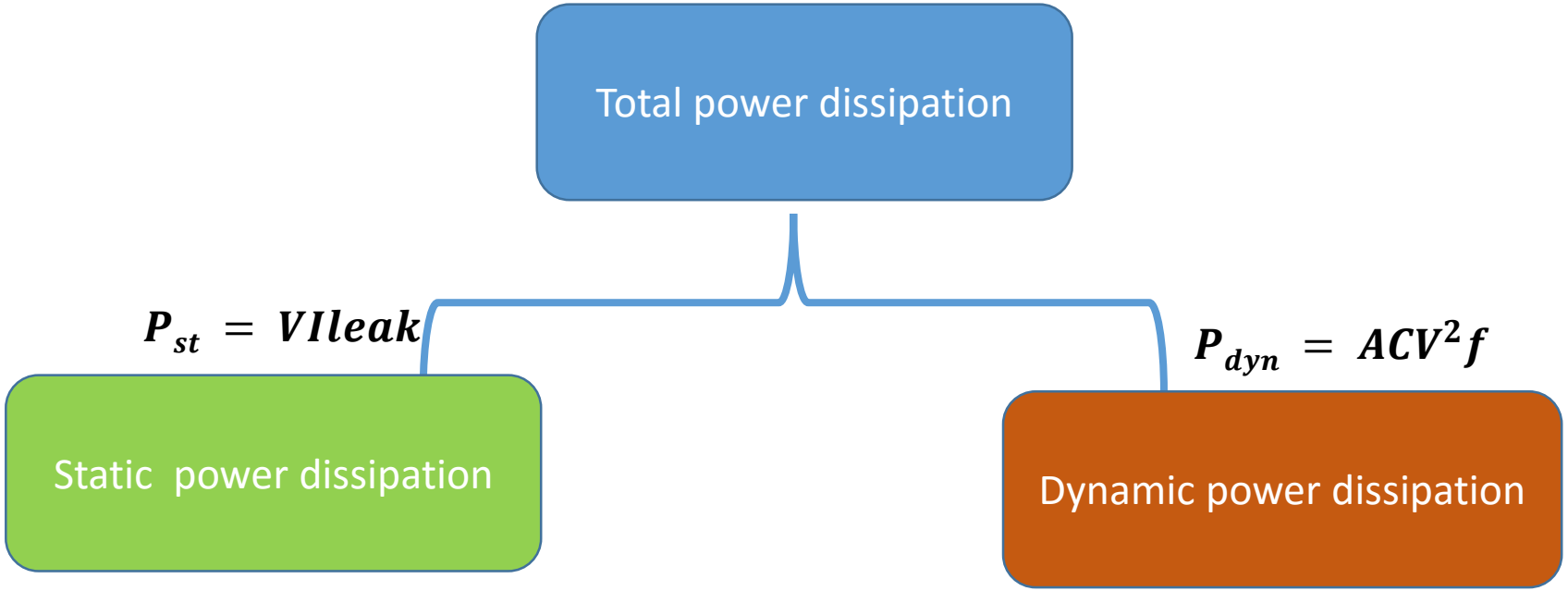


- Power → temperature rise → temperature induced effects in device functionalities.

- Increasing number of battery operated devices like hand phone and tablets. As energy consumption increases battery life decreases.

High- performance with less power consumption is required for all kinds of computers (just not portable devices).

Components of power dissipation in processor



Lower operating voltage V

Minimize I_{leak} (Leakage current)

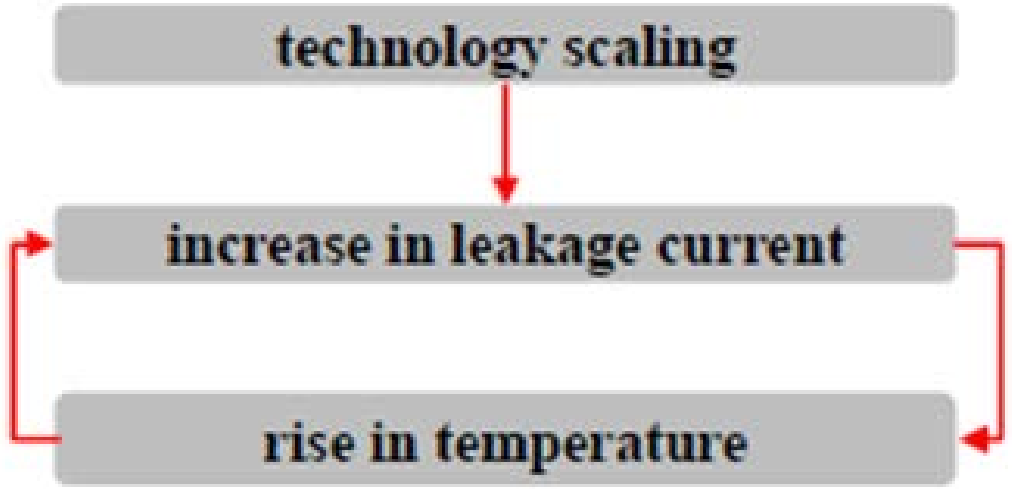
Lower Operating frequency

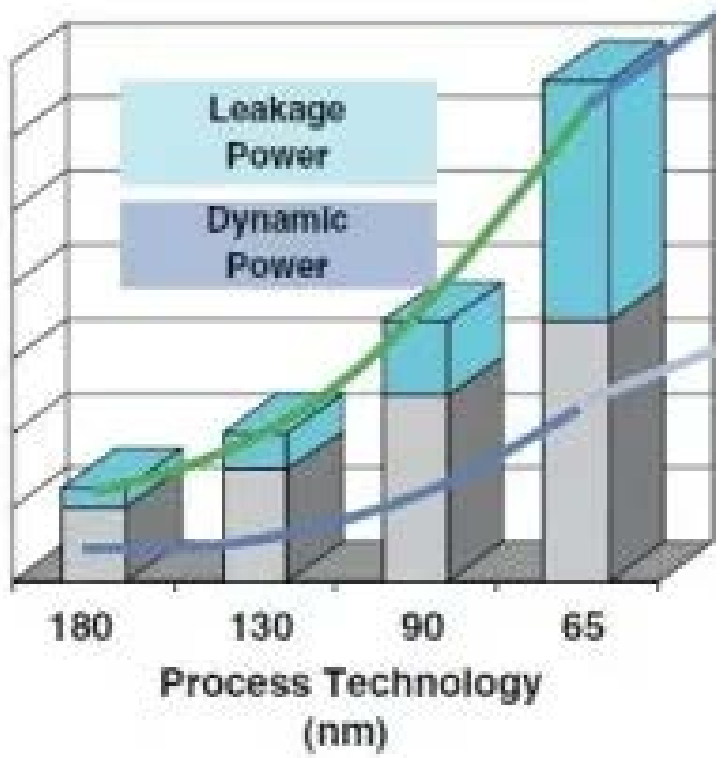
degrade performance

Less switching activity

Major Issues?

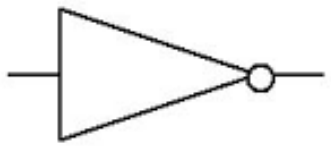
Minimize I_{leak} (Leakage current)





Why technology scaling increases the leakage Current?

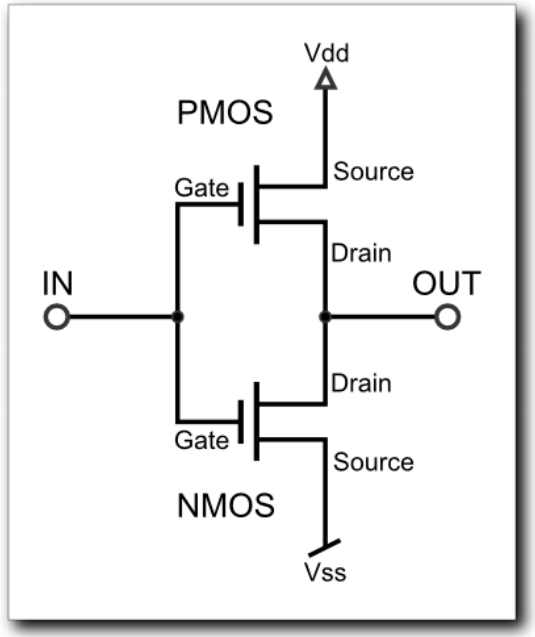
NOT



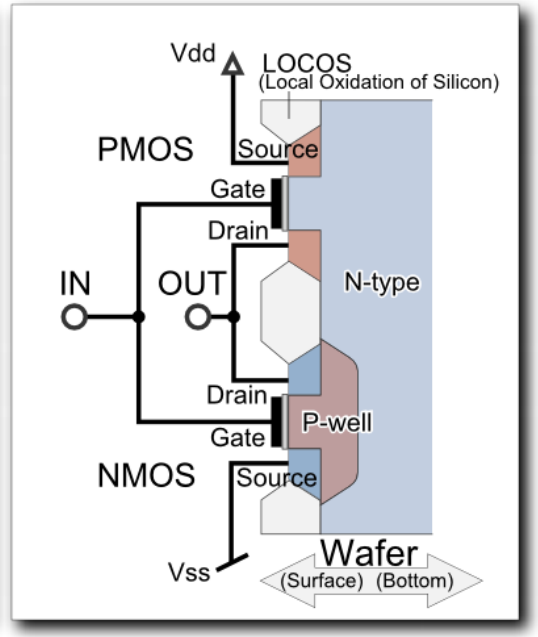
Input	Output
0	1
1	0

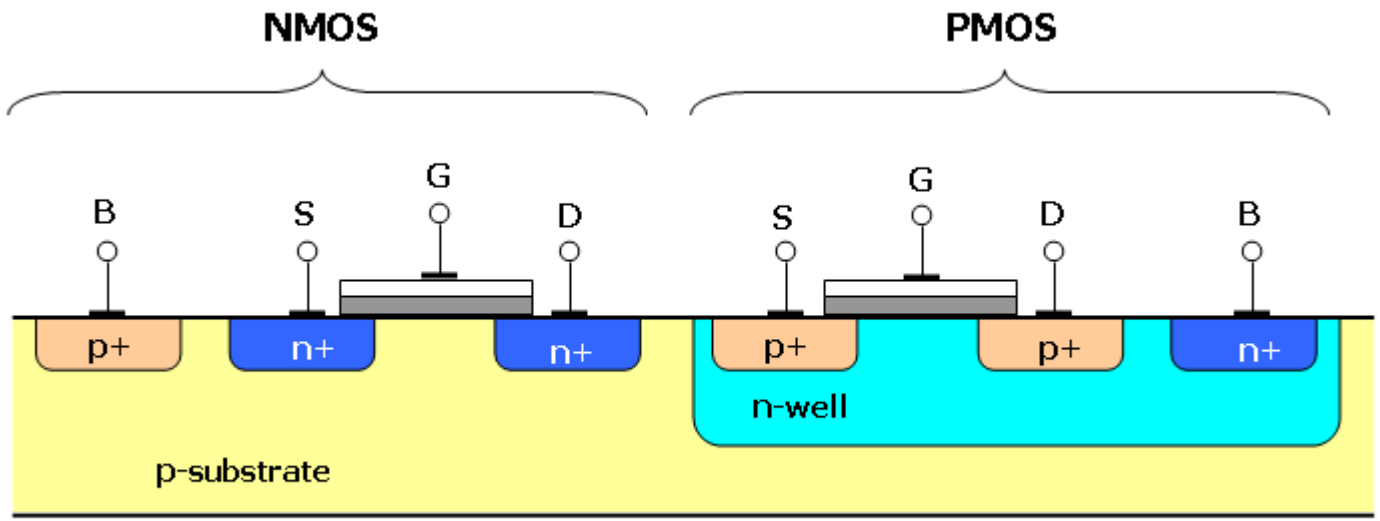
CMOS inverter

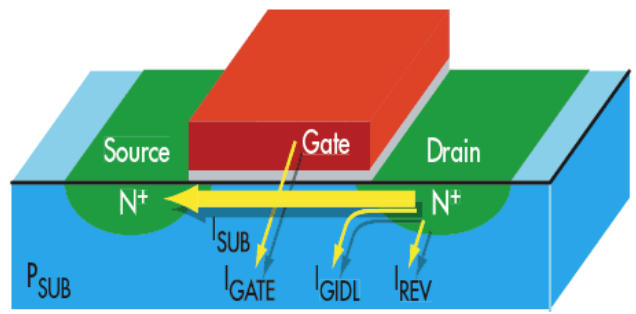
Model chart



Silicon wafer







Maximum clock frequency and operating voltage

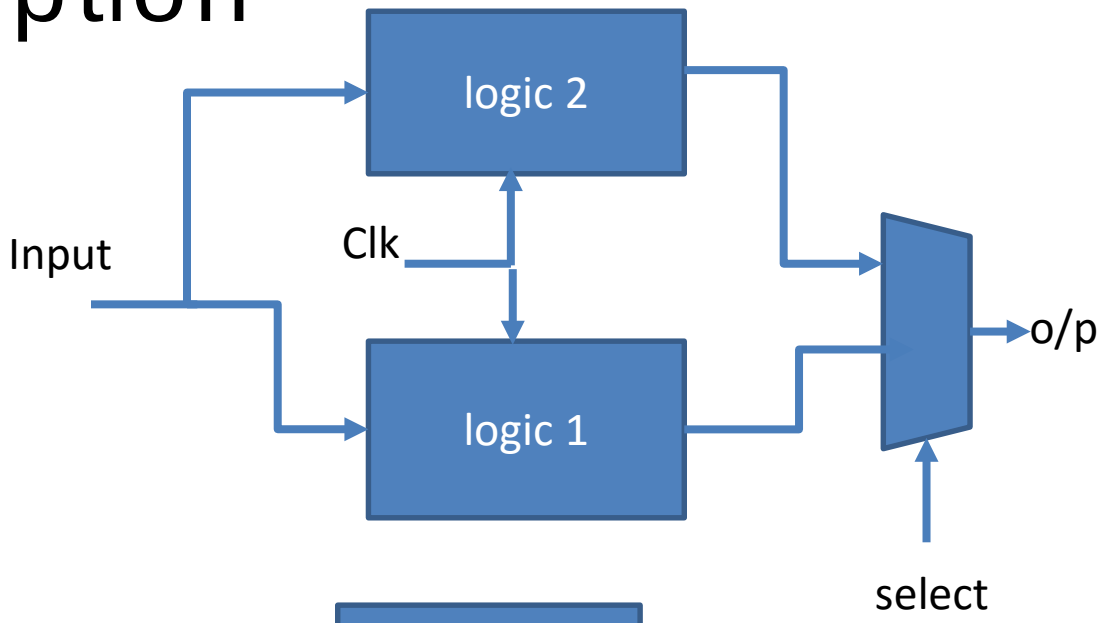
$$F_{max} \ \alpha \ (V-V_{th})^2/V$$

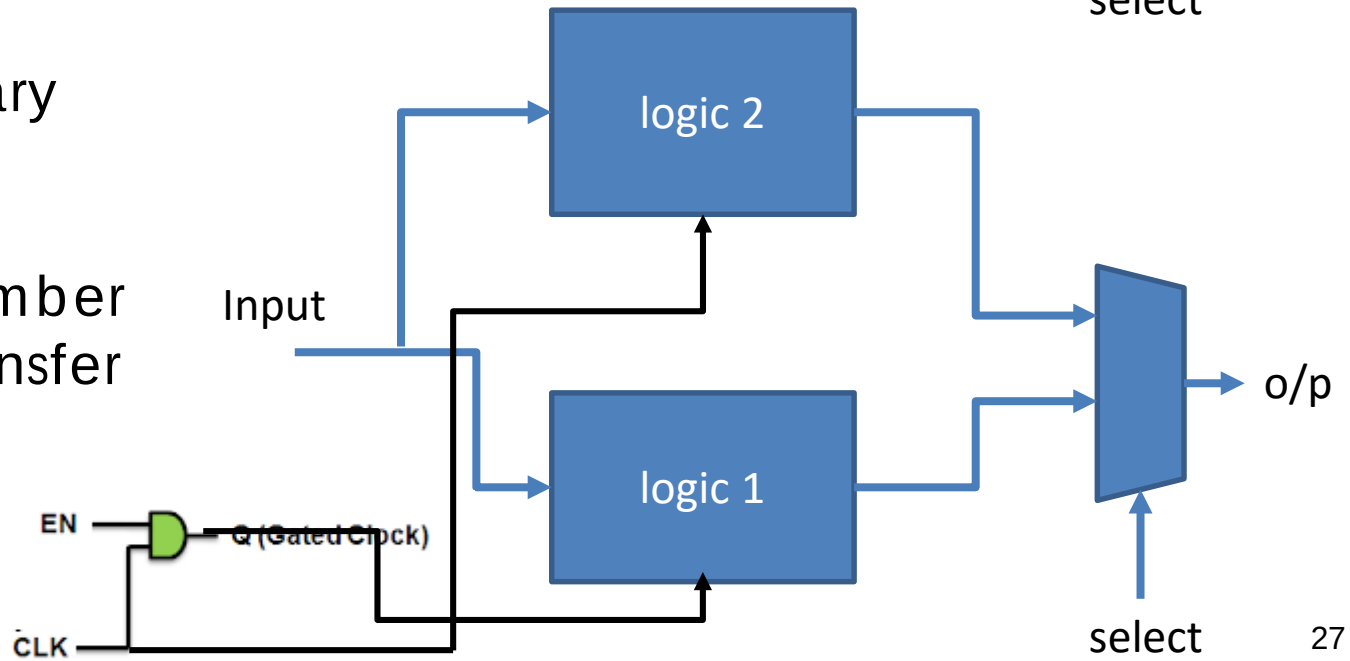
V_{th} is the threshold voltage , the gate-source voltage at which the transistor just starts conducting

If V_{th} is small compared to V, then $F_{max} \ \alpha \ V$

Reducing power consumption

- Efficient cache and memory hierarchy design
- Power gating: shutting down the unused components (enable signals)
- Clock gating: to reduce unnecessary switching
- Reducing the data movement, number of memory access, and register transfer





Example

Consider a processor while working at its maximum operating clock frequency of 500 MHz consumes 80 Watt dynamic power and 10 Watt static power. It consumes 540 kJ of energy for a given computation. If the leakage current is decreased by 10% by reduction of temperature and operating clock frequency is reduced to half what will be the energy consumptions due to static power and dynamic power consumptions.

Total power consumption = 80 + 10 = 90 Watts.
Execution time = 540 K/90 = 6000 seconds.
New static power = 10 -1 = 9 Watts.
New dynamic power = 80/2 = 40 Watts.
New execution time = 12000 seconds.
Energy consumption due to new static power = 9 × 12000 Joules = 108 kJ.
Energy consumption due to new dynamic power = 40 × 12000 Joules = 480 kJ.

Module 1 Summary

- Introduction
  - Von Neumann and Harvard Architecture
  - Need of high performance computing
  - Design goals and constraints
- Performance enhancement
  - Iron law
  - Amdhals Law
  - Gustafson's Law
  - Other Performance metrics
    - MIPS and MFLOPS
- Power dissipation in processors power metrics and low power design technics