Digital Design

Week 11: Optimizations and Trade-offs

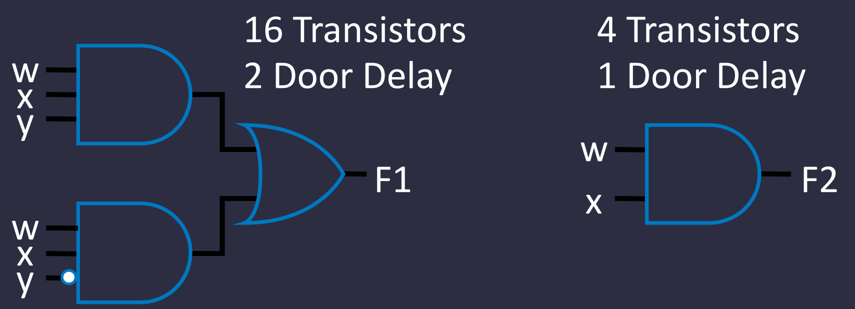
Fenerbahce University

Lesson plan

- Optimizations and Trade-offs

Optimizations and Trade-offs

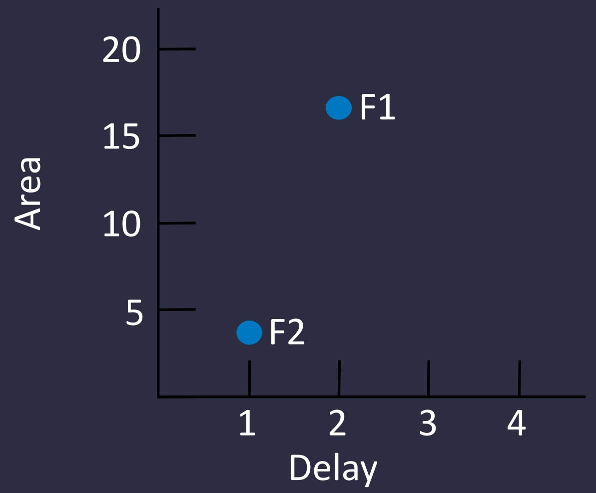
- We know how to build digital circuits
  - How to build better circuits?
- Two major constraints
  - ***Delay*** (Latency) – Elapsed time between input to output
  - ***Area*** – Number of transistor



F1 = wxy + wxy'

F2 = wx

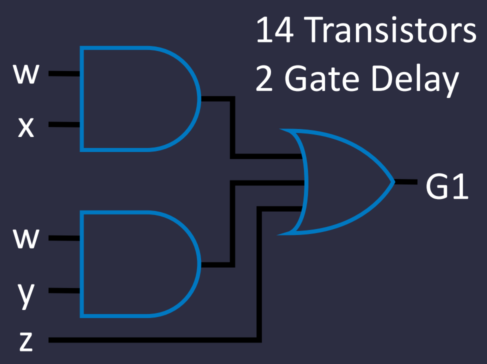
= wx (y+y ') = wx



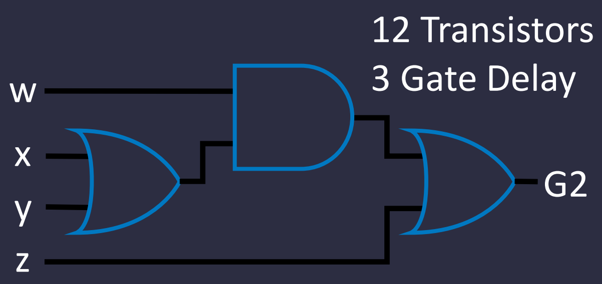
Note: Assume the gate delays are equivalent

Optimizations and Trade-offs

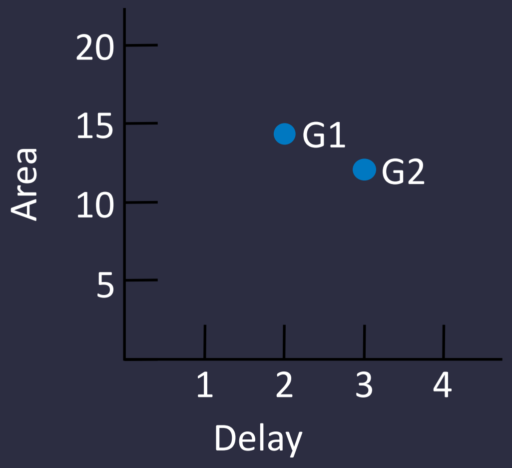
- Tradeoff
  - When 1 criterion improves, the other criterion worsens.
  - Design is made by giving priority or balancing according to a criterion.



G1 = wx + wy + z



G2 = w(x+y) + z

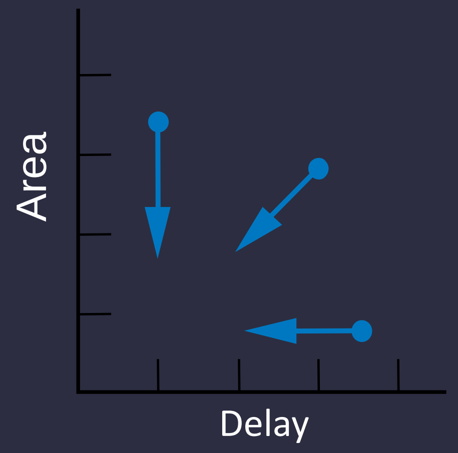


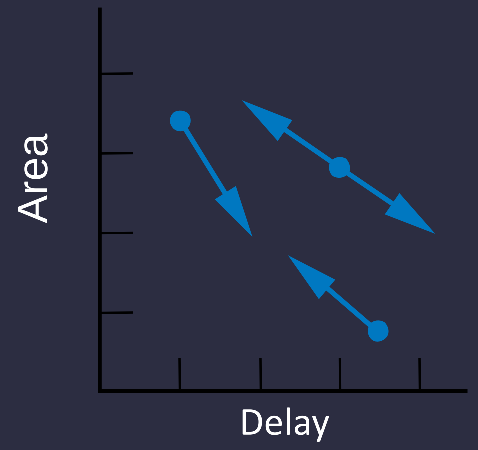
Note: Assume the gate delays are equivalent

Optimizations and Trade-offs

Optimization

Improvement of all criteria





trade-off

Some criteria improve while others worsen.

- If possible, the design is made by optimizing, otherwise making a choice according to the requirements and making tradeoffs.
  - It is designed according to the target, such as how it is not the most comfortable, the most fuel-efficient and the fastest car.

Optimizations and Trade-offs

- Space optimization
- Express the problem as a function
  - Simplify the equation with

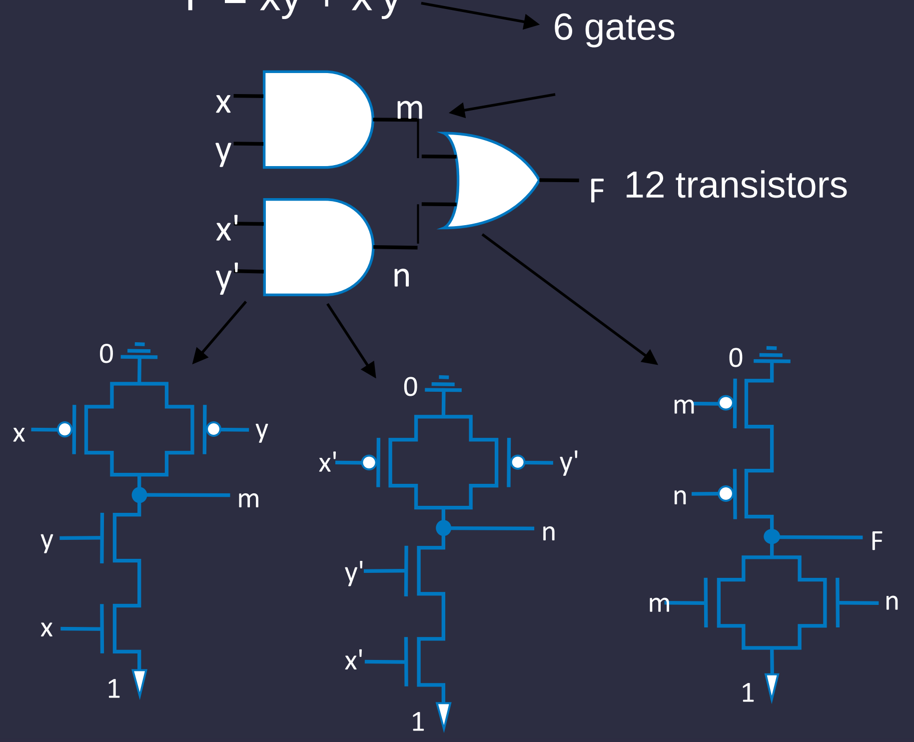
Sample

$F = xyz + xyz' + x'y'z' + x'y'z$

$F = xy(z + z') + x'y'(z + z')$

$F = xy*1 + x'y'*1$

$F = xy + x'y'$



Karnaugh Maps

- Moore 's rules, it is very possible to miss expressions that can be simplified.
- ***Karnaugh Map (K-map)***
  - Provides a graphical overview for simplification
  - Simplification is d1 by grouping expressions.
    - Simplification is made within the rectangle groups formed by

K-map

F x \ YZ	00	01	11	10
0	x'y'z'	x'y'z	x'yz	x'yz'
1	xy'z'	xy'z	xyz	xyz'

F = x'y'z + xyz + xyz' + x'y'z'

F x \ YZ	00	01	11	10
0	1	1	0	0
1	0	0	1	1

F x \ YZ	00	01	11	10
0	1	1	0	0
1	0	0	1	1

x'y' xy

F = x'y' + xy

F = xyz + xyz ' + x'y'z ' + x'y'z
F = xy (z + z') + x'y '(z + z')
F = xy *1 + x'y '*1
F = xy + x'y '

K-maps

- If there are 4 neighbor 1s, that value can be eliminated.
  - In other words, if the output is 1 in all cases, it means that it has no effect on the output.

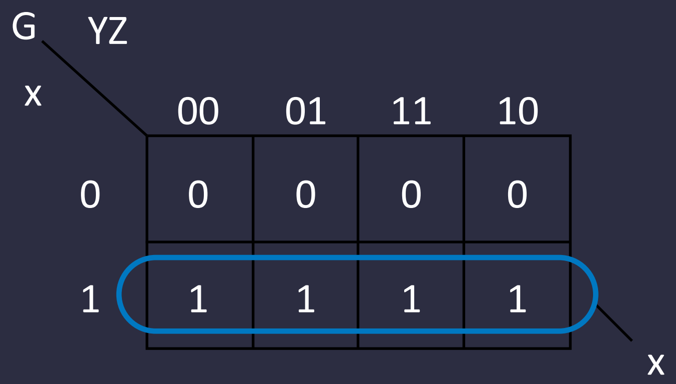
G = xy'z ' + xy'z + xyz + xyz '

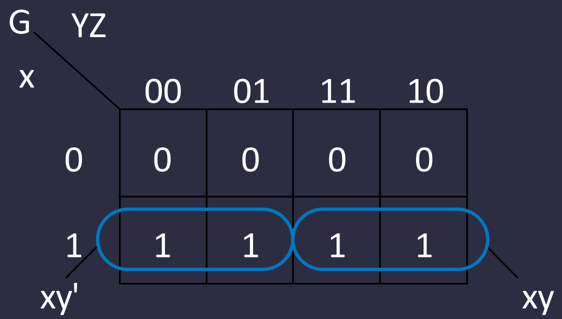
G = x(y'z '+ y'z + yz + yz ')

G = x(y'(z'+z) + y(z+z '))

G = x(y'+y)

G = x





K-maps

- Examples

J (x \ YZ)

x \ YZ	00	01	11	10
0	1	1	0	0
1	0	1	1	0

Groups: x'y', y'z, xz

$J = x'y' + y'z + xz$

$H = x'y'z + x'yz + xy'z + xyz$

H (x \ YZ)

x \ YZ	00	01	11	10
0	0	1	1	0
1	0	1	1	0

Group: z

$H = z$

I (x \ YZ)

x \ YZ	00	01	11	10
0	0	1	0	0
1	1	1	1	1

Groups: y'z, x

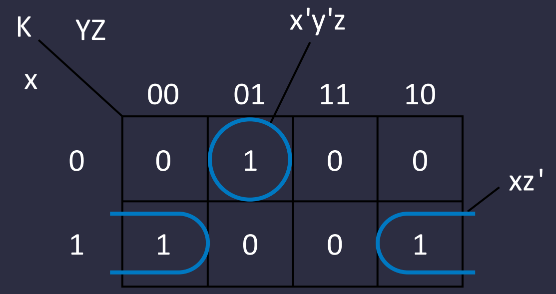
$I = x'y'z + xy'z' + xy'z + xyz + xyz'$

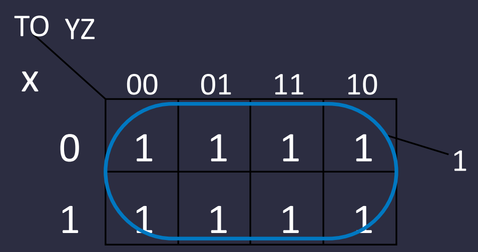
$I = x'y'z + xy'z + xy'z' + xy'z + xyz + xyz'$

$I = (x'y'z + xy'z) + (xy'z' + xy'z + xyz + xyz')$

$I = (y'z) + (x)$

- Rectangles can emerge from Kmap's boundaries from the opposite side.
- Rectangles must have multiples of 1, 2, 4, 8 …





K-maps

- can be calculated by the same principle in Kmaps with 4 values
- 5 and 6 value maps
  - However, it is very difficult to use

F (y \ z)

y \ z	0	1
0		
1		

F (wx \ YZ)

wx \ YZ	00	01	11	10
00	0	0	1	0
01	1	1	1	0
11	0	0	1	0
10	0	0	1	0

w'xy '

YZ

F=w'xy'+yz

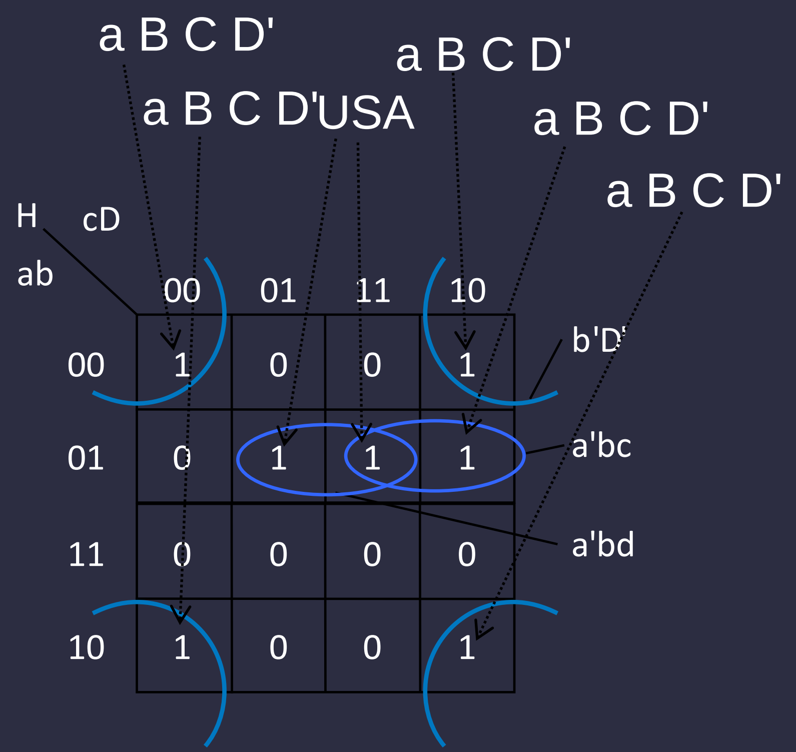
G (wx \ YZ)

wx \ YZ	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	0	1	1	0
10	0	1	1	0

z

G=z

- Example :
  - H = a'b '(cd' + c'd ') + ab'c'd ' + ab'cd ' + a'bd + a'bcd '

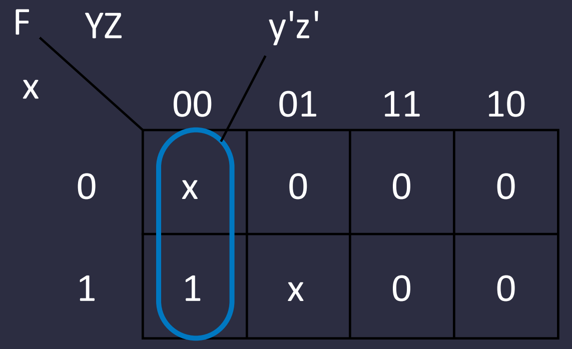


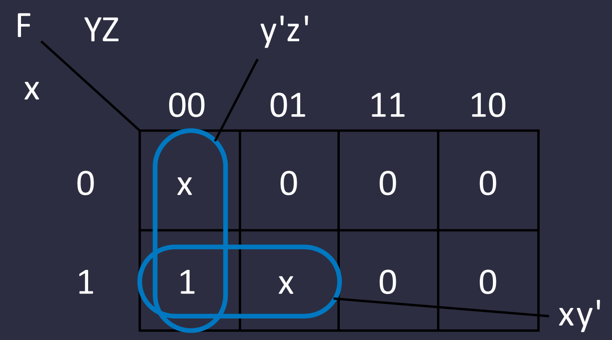
H (ab \ cD)	00	01	11	10
00	1	0	0	1
01	0	1	1	1
11	0	0	0	0
10	1	0	0	1

H = b'd ' + a'bc + a'bd

Do n't Care Entries

- Some input combinations may not be entered at all
  - x, y, and z will never be 0 at the same time for the formula F = xy'z ' and x 1, y 0, z 1 will not be simultaneously, they can be written as negligible.
  - It can be thought of as 1 or 0 to help simplify the function
- in kmaps
  - X is written instead of ignored
  - Not to be confused with variable X

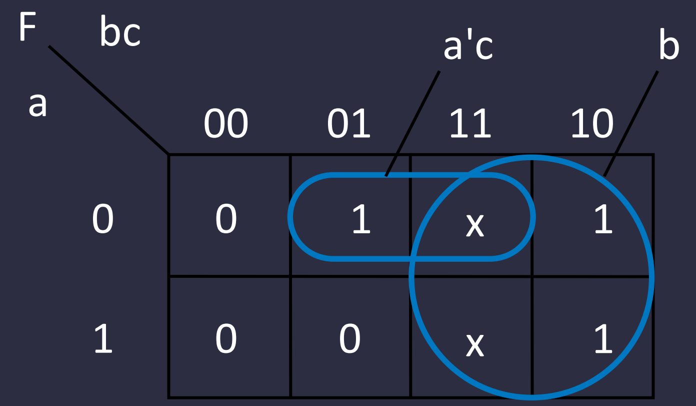




Brings too many terms in unnecessary use

dont Minimization

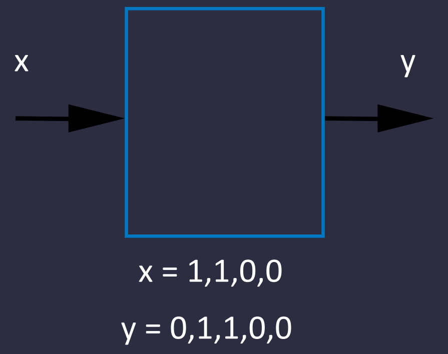
- Example :
  - F = a'bc ' + abc ' + a'b'c
  - Ignored entries :
    - a B C
    - a B C

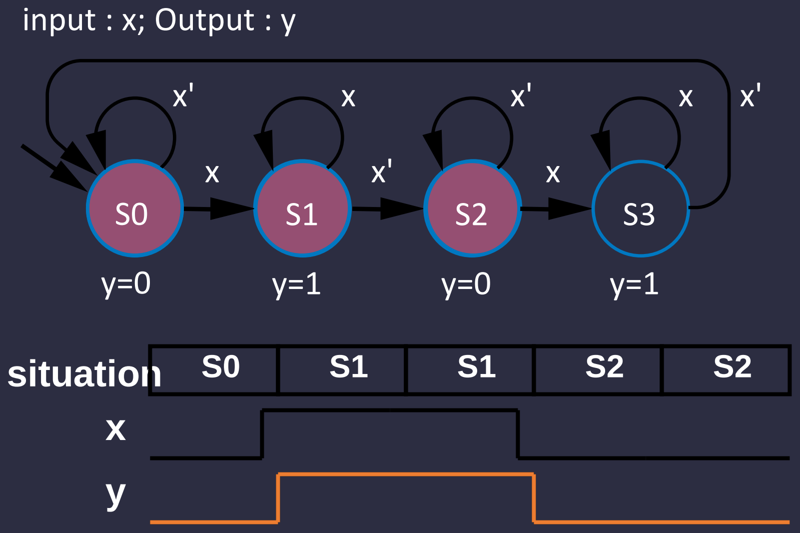


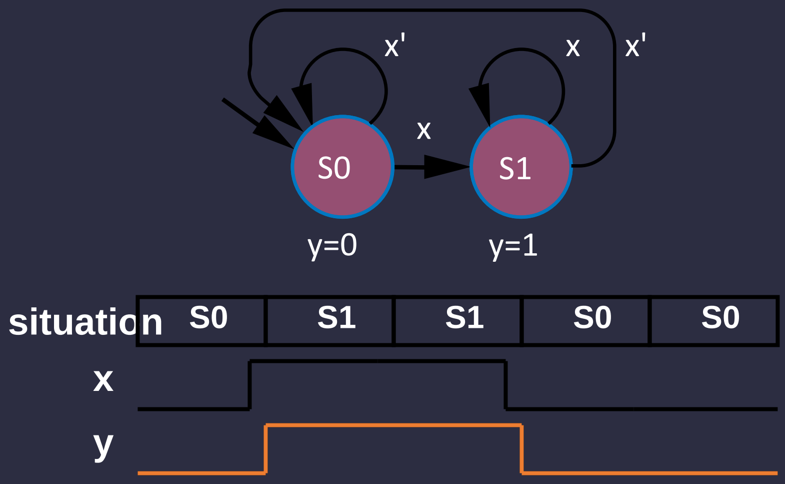
F = a'c + b

Condition Mitigation

- **Reducing the number of states without changing the behavior of**
  - Fewer states will likely result in less footprint
- FSMs for







Condition Mitigation

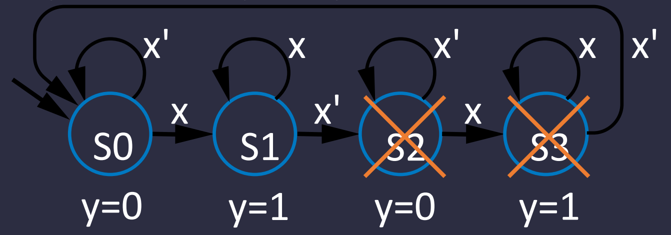
If the two conditions are equivalent

If they assign the same value to the output
- So S0 and S2 are throwing 0 to the output ,
- **S1 and S3 throw 1 to the output**

If starting from equivalent states and producing the same outputs when the same inputs come,
- for example *x* =1,1,0,0,...
  - **S1 start** , *y* =1,1,0,0,...
  - **S3 initial** , *y* =1,1,0,0,...

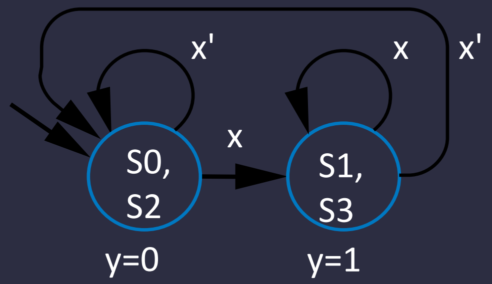
Statuses can be used in common.

Input : x; Output : y



State S0 and S2 equivalent

Case S1 and S3 equivalent

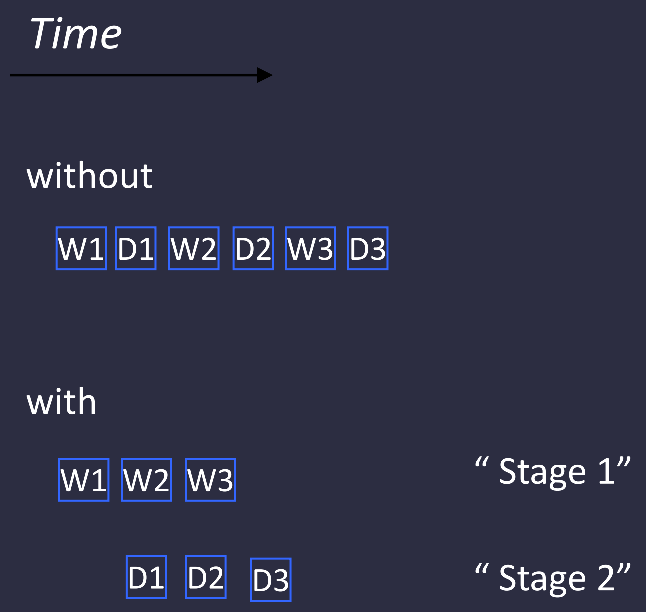


Pipelining

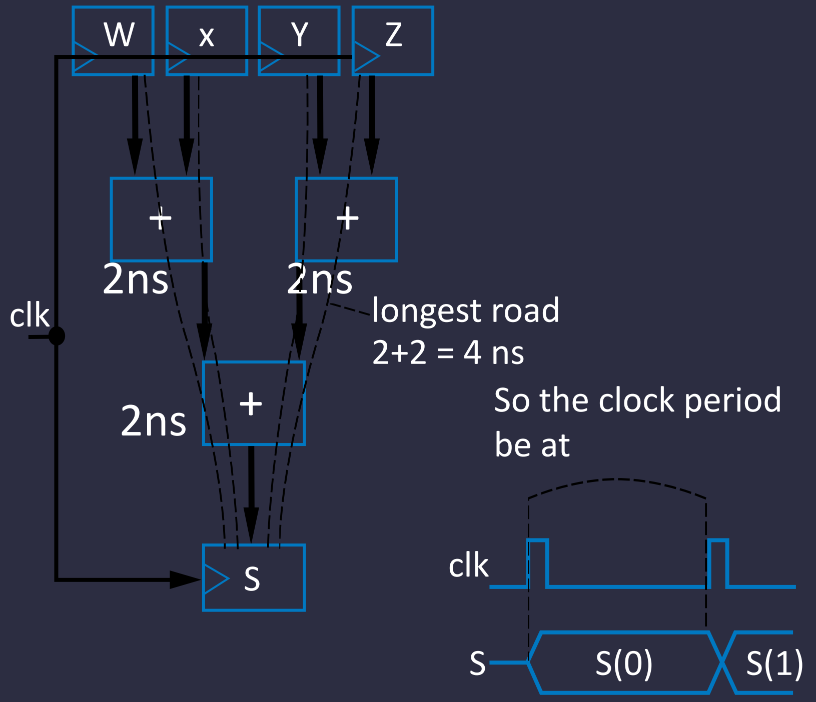
Pipelining: It is a structure that divides a job into stages, where the stages feed data to each other and can work in parallel.

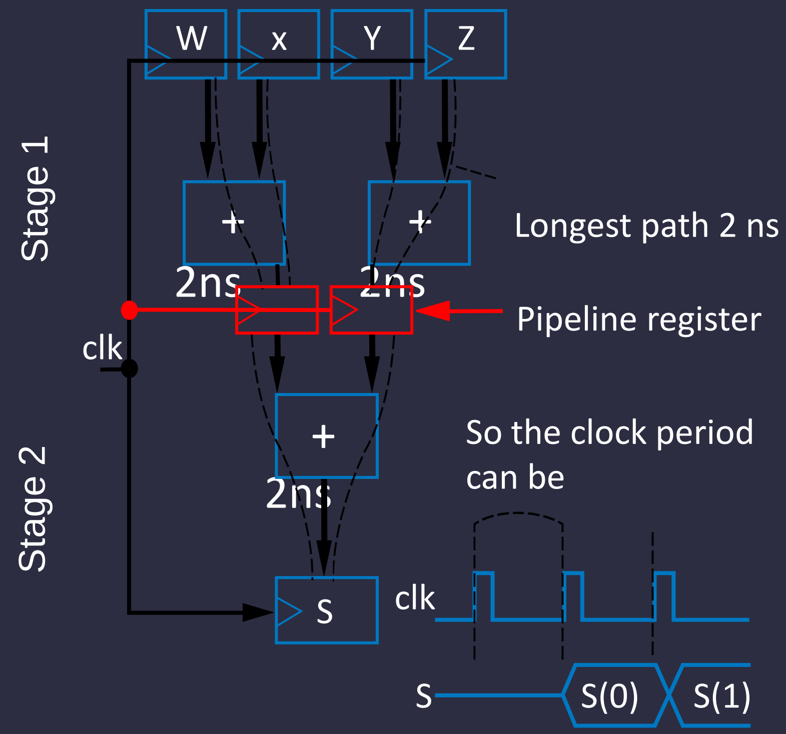
Sample;

- You are washing the dishes, your friend is setting up.
  - 1 plate washed
  - While 1 plate is dried , *2 plates are washed*
  - While 2 dishes are being dried , 3 dishes are being washed…
  - You don't wait for your friend to finish drying the plate.



Pipelining Example



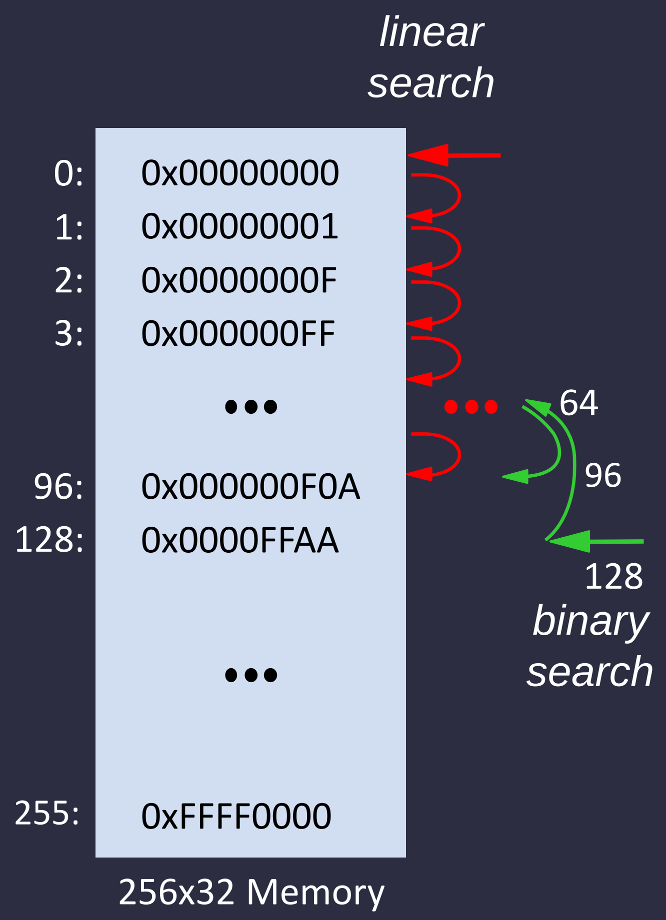


- S = W+X+Y+Z
- The left-hand circuit can receive inputs as fast as 4 ns .
- The circuit on the right can receive input in 2 ns as the fastest . Speed doubled.

Algorithm Selection

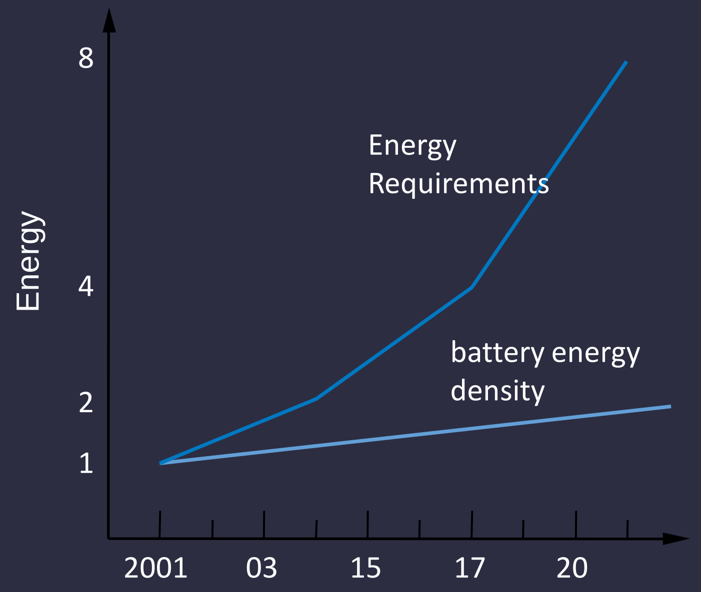
- The chosen algorithm has great influence

- Example : Searching for an element in a 256-address memory
  - Algorithm 1: “ Linear search”
    - Each element is checked individually M[0] , M[1], M[2], ...
    - Worst case 256 attempts
  - Algorithm 2: “ Binary search ”
    - Only takes 8 worst tries



Power Optimization

- **Strength** Another important design criterion is
  - Watts (Volts * Current)
- It is an important parameter as space consumption.
  - Especially important on mobile devices
  - With the developments in batteries, the required energy need does not increase at the same level.
  - Therefore, more power efficient designs are required.
  - CMOS technology, switching from 0 to 1 (Dynamic Power, Dynamic power)
    - $P = k * CV^2 f$
      - k: constant ;
      - C: capacitance of cables ;
      - V: voltage ;
      - f: Switching frequency

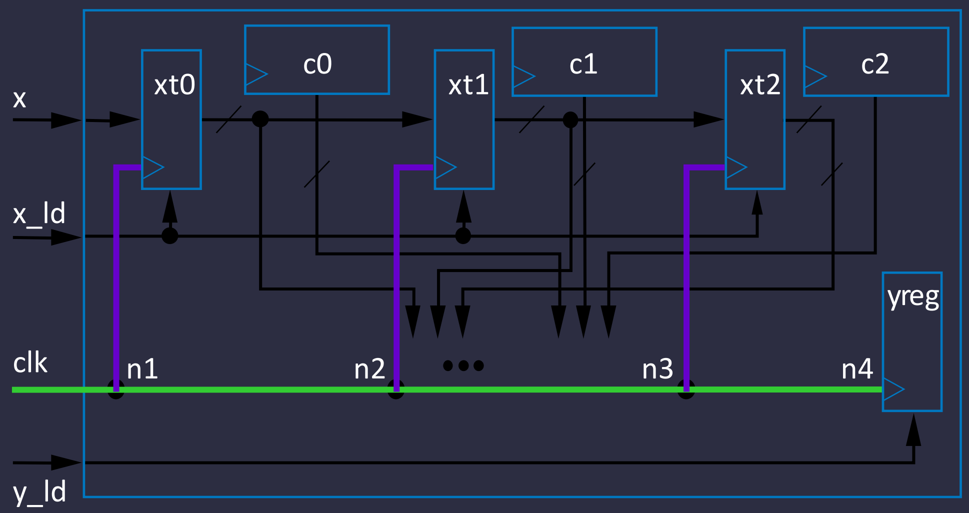


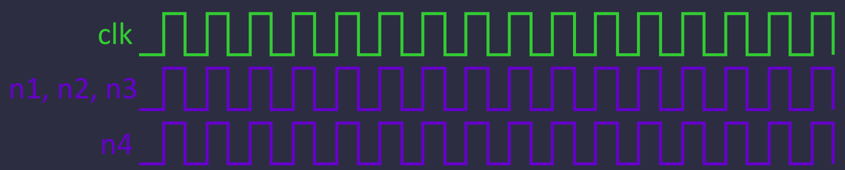
Power Optimization (Clock Gating)

- Exchange of signals within the chip increases power consumption

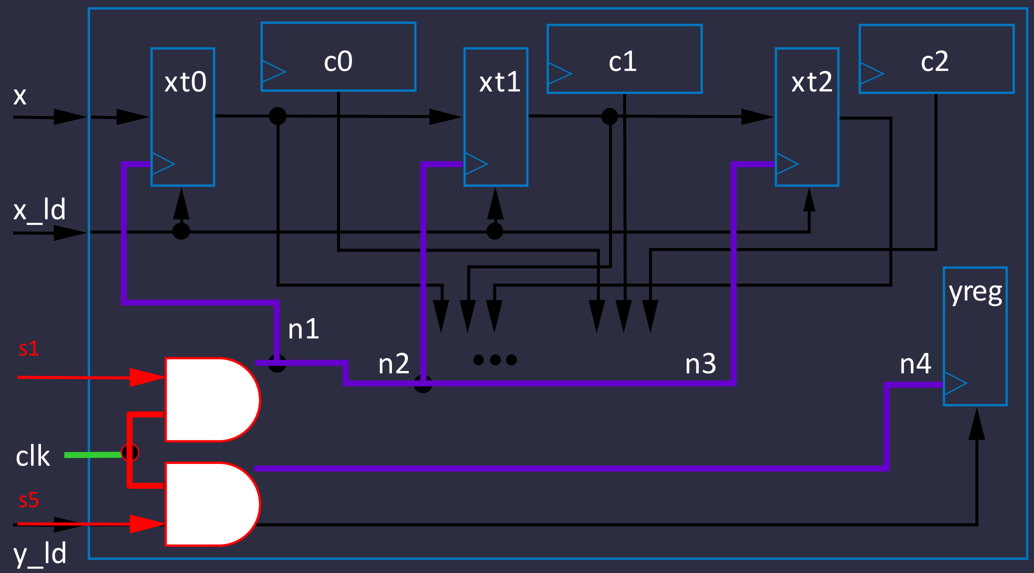
- The solution to this is to stop unused registers in certain situations.
    - Ande gate

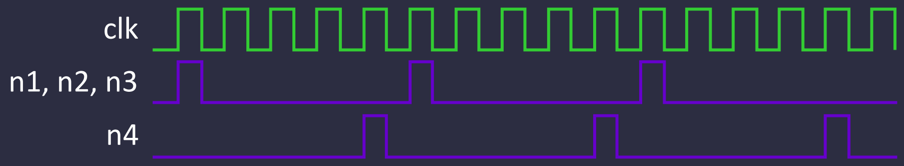
Power Optimization (Clock Gating)





You heavily have a





switching is reduced

Power Optimization (Low Power Gates)

- Low Power Gates
  - There can be multiple versions of doors performing the same task.
    - fast/high power consumption slow/low power consumption
  - critical path as slow/low power consumption, the power can be reduced without affecting the total delay.