Computer Architecture

Yuqiao Meng

2022-1-28

Contents

1 Overview

Intel Xeon Processor

$13012

$400

- **Xeon Platinum 8380HL**
  - **28: Cores**
  - **56: Threads**
  - **Base Frequency: 2.9 GHz**
  - **Max Turbo Frequency: 4.3 GHz**
  - **Cache: 38.5MB**
  - **Max Memory Size: 4.5 TB**
  - **Memory Channels: 6**
  - **Max. Memory Speed: 3200MHz**
  - **Package Size: 77.5 x 56.5mm**
- **Intel i9-10850K**
  - **10: Cores**
  - **20: Threads**
  - **Base Frequency: 3.6 GHz**
  - **Max Turbo Frequency: 5.2 GHz**
  - **Cache: 20MB**
  - **Max Memory Size: 128GB**
  - **Memory Channels: 2**
  - **Max. Memory Speed: 3200MHz**
  - **Package Size: 37.5 x 37.5mm**

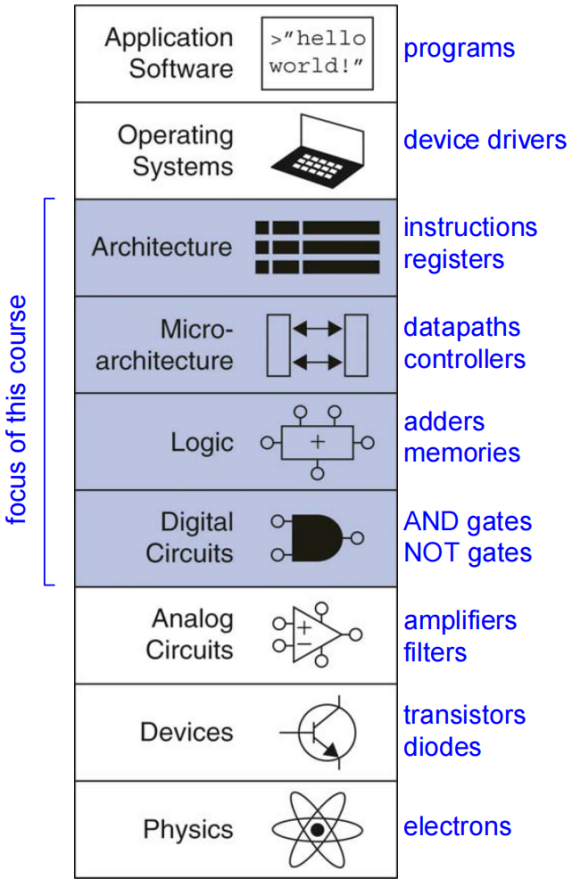
1.1 Why the prices of two processors differs?

1. The prices increase exponentially as the number of Cores and Threads increase. That's because every core has a bad possibility, so the difficult of making a processor with many cores much harder.

2. Significant: The memory Chaneels of XEON is three times by i9, which means it has three times as many pins as i9 has.

2 Digital Design

2.1 Abstraction



2.2 The Digital Abstraction

- Most physical variables are continuous
  - Voltage on a wire
  - Frequency of an oscillation
  - Position of a mass
- Digital abstraction considers discrete subset of values

2.3 Digital Discipline: Binary Values

- Two discrete Values:
  - 1's and 0's
  - 1, true, high
  - 0, false, low
- 1 and 0: voltage levels, rotating gears, fluid levels
- Digital circuits use voltage levels to represent 1 and 0

2.4 Decimal to Binary Conversion

Method: repeatedly divided by 2, remainders goes in next most significant bit

53_{10} =

53/2 = 26 R1
26/2 = 13 R0
13/2 = 6 R1
6/2 = 3 R0
3/2 = 1 R1
1/2 = 0 R1

$= \mathbf{110101_2}$

2.5 Signed Binary Numbers

2.5.1 Sign/Magnitude Numbers

Problems

- Has two 0 values, positive 0 and negative 0
- Addition of a negative and a postive will fail

2.5.2 Two's Complement Numbers

Conversion from positive to negative:

- Invert every bit
- Add 1

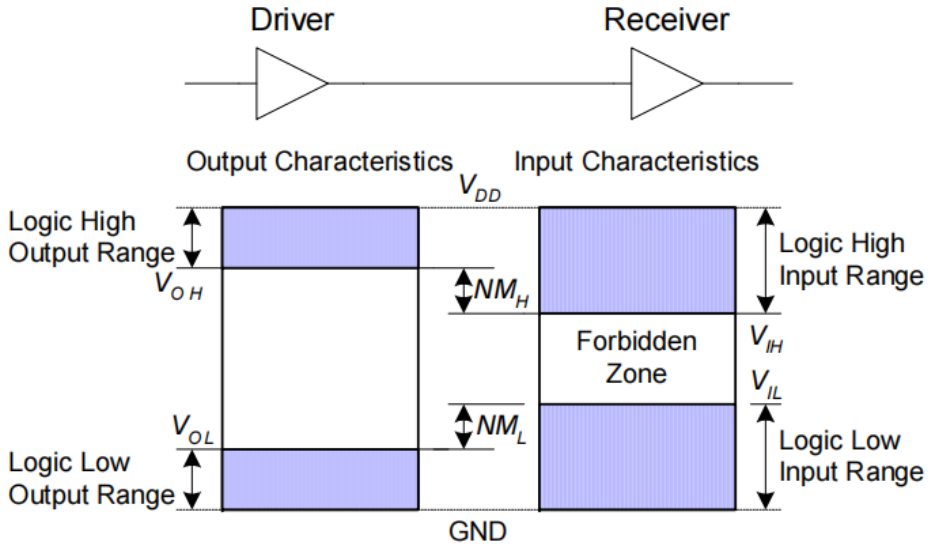
2.6 Logic Gates

- AND
- OR
- XOR
- NAND
- NOR
- XNOR

2.7 Logic levels

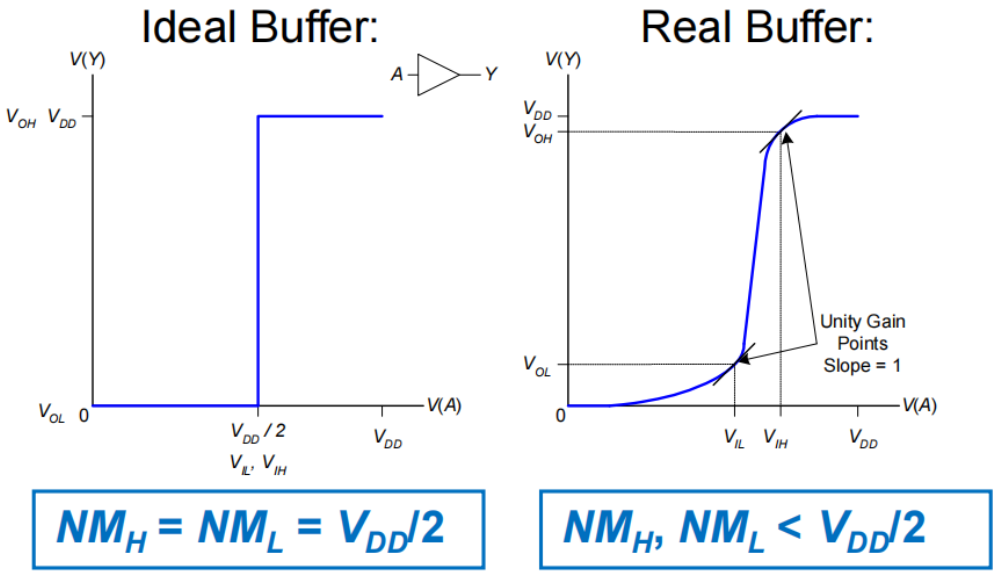
Discrete voltages represent 1 and 0, but there is noise that will degrade the signal

2.7.1 Noise Margins



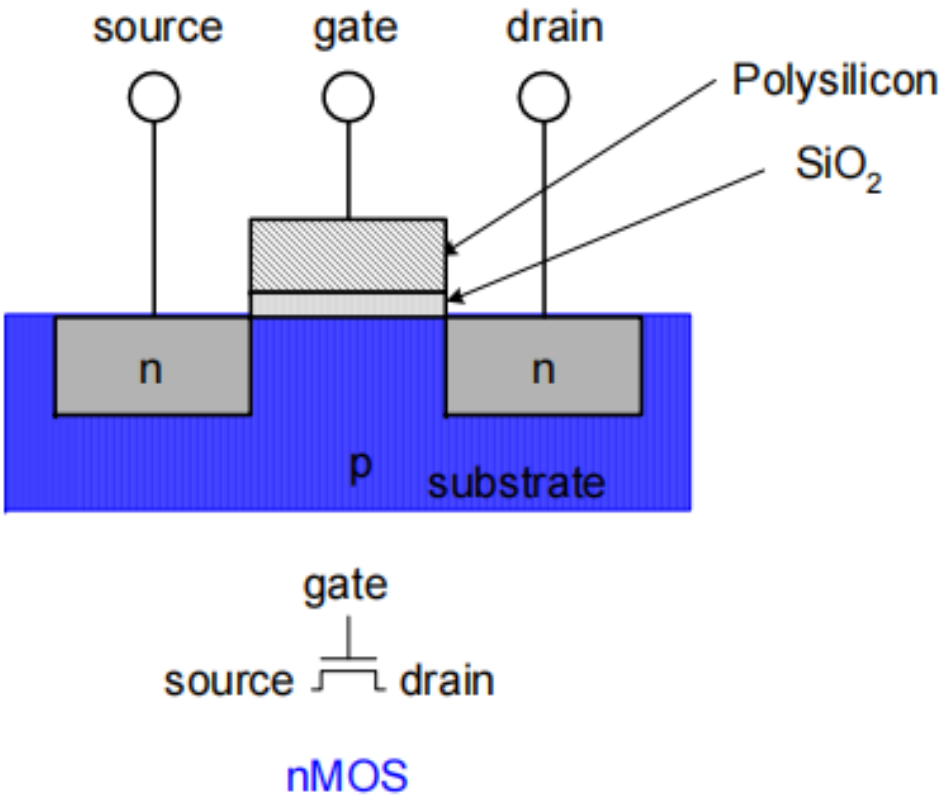
- High Noise Margin: $NM_H = V_{OH} - V_{IH}$
- Low Noise Margin: $NM_L = V_{OL} - V_{IL}$

2.7.2 Transfer Characteristics



2.8 MOS Transistor

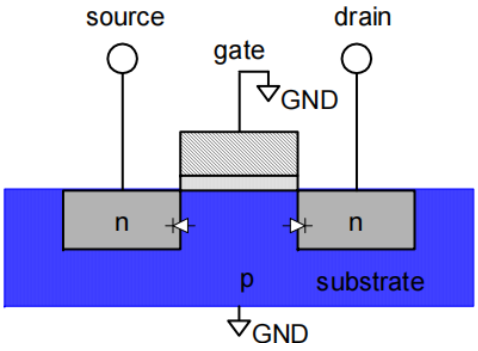
- Polysilicon Gate: control the transistor
- Oxide insulator: keep electrons to form a path for electricity
- Doped Silicon



2.8.1 nMOS

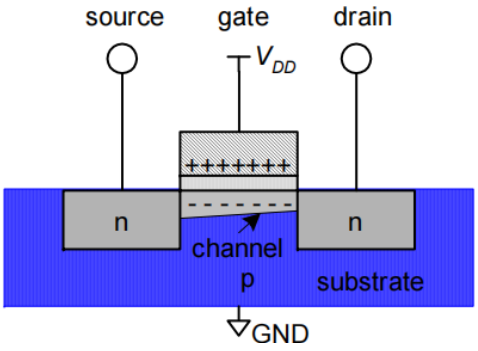
Gate = 0

OFF (no connection between source and drain)



Gate = 1

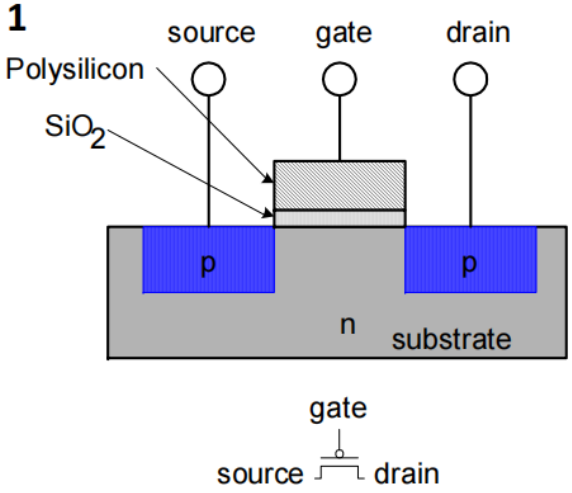
ON (channel between source and drain)



2.8.2 pMOS

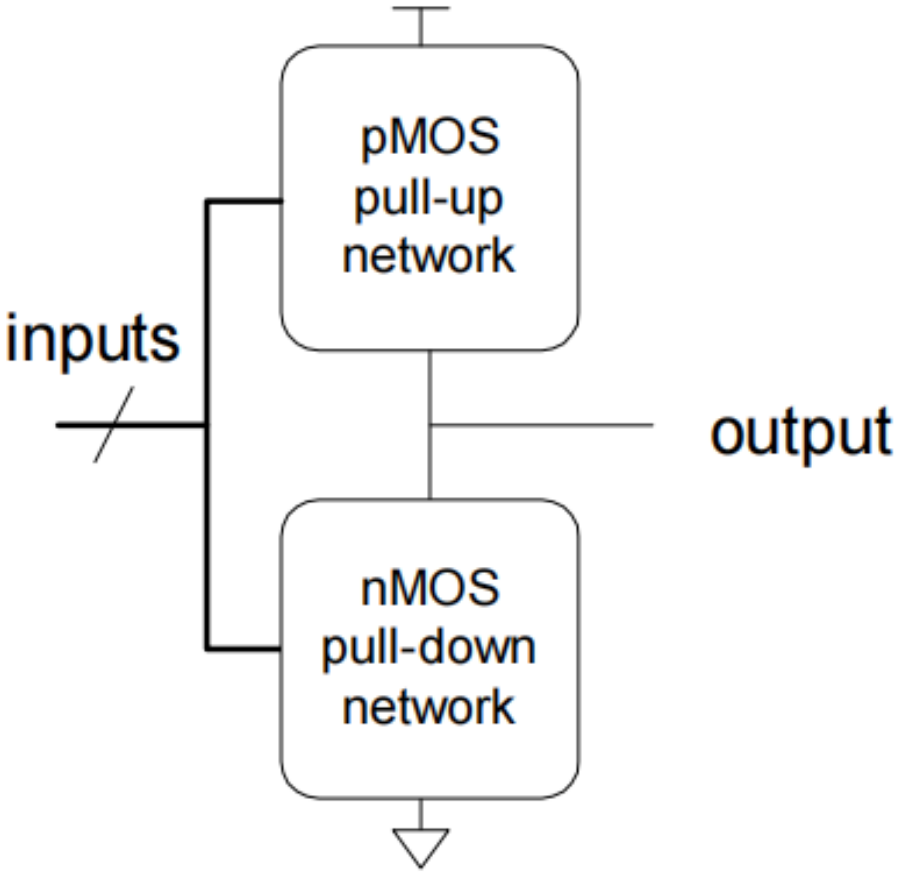
pMOS transistor is opposite

- **ON** when **Gate = 0**
- **OFF** when **Gate = 1**

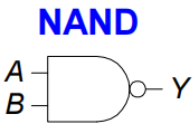


2.8.3 Why do we need two types of MOS?

- nMOS: pass good 0's, so connect source to GND
- pMOS: pass good 1's, so connect source to VDD

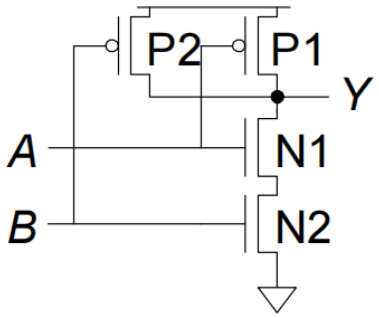


2.9 cMos NAND Gate



$Y = \overline{AB}$

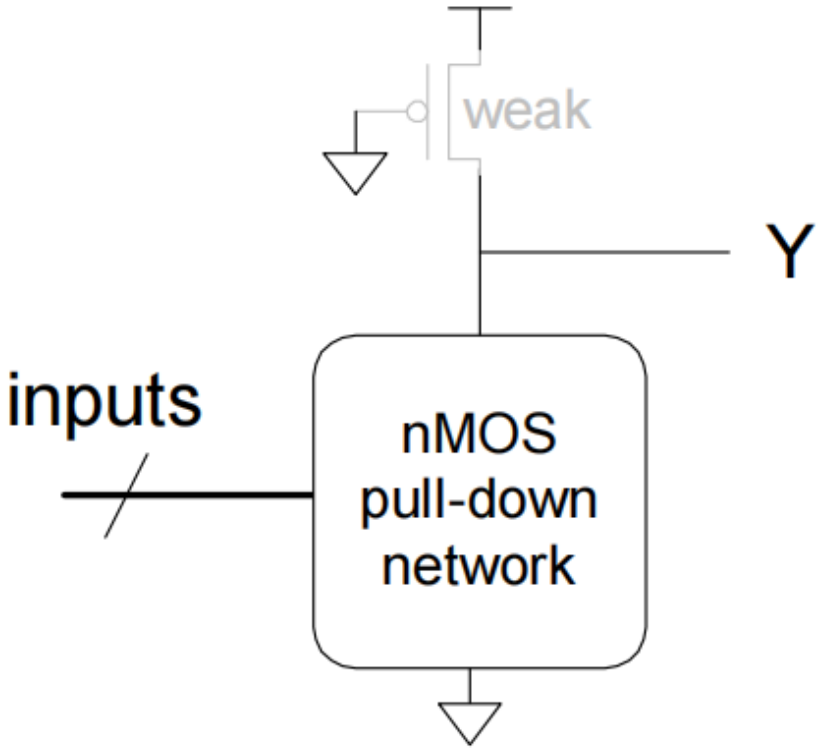
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0



A	*B*	P1	P2	N1	N2	*Y*
0	0	ON	ON	OFF	OFF	1
0	1	ON	OFF	OFF	ON	1
1	0	OFF	ON	ON	OFF	1
1	1	OFF	OFF	ON	ON	0

2.10 Pseudo-nMOS Gates

- Replace pull-up network with weak pMOS transistor that is always on
- Tradeoff: need more energy



2.11 Power Consumption

- Dynamic power consumption: Power to charge transistor gate capacitances
  - $P_{Dynamic} = \frac{1}{2}CV_{DD}^2 f$
  - f: Frequency
  - $CV_{DD}^2 f$: energy to charge a capacitance
- Static power consumption
  - power consumed when no gates are switching
  - caused by quiescent supply current I_{DD}(there is more and more current leackage for transistor is smaller and smaller, and the distance between source and drain is shorter and shorter, so there is more and more quiescents leaked)

  - $P_{static} = I_{DD}V_{DD}$
- $P = \frac{1}{2}CV_{DD}^2 f + I_{DD}V_{DD}$

2.12 Circuits

2.12.1 Composition

- inputs
- outputs
- functional specification
- timing specification

2.12.2 Types

- Combinational
  - Memoryless
  - output $\rightarrow$ current value of inputs
- Sequential
  - has memory
  - output $\rightarrow$ previous and current value of inputs

2.12.3 Definitions

- **Complement**: variable with a bar over it
  $\bar{A}, \bar{B}, \bar{C}$
- **Literal**: variable or its complement
  $A, \bar{A}, B, \bar{B}, C, \bar{C}$
- **Implicant**: product of literals
  $AB\bar{C}, \bar{A}C, BC$
- **Minterm**: product that includes all input variables
  $AB\bar{C}, A\bar{B}\bar{C}, ABC$
- **Maxterm**: sum that includes all input variables
  $(A+\bar{B}+C), (\bar{A}+B+\bar{C}), (\bar{A}+\bar{B}+C)$

2.12.4 SOP Forms

- Each row has a minterm(products of literals)

A	B	Y	minterm	minterm name
0	0	0	$\bar{A}\,\bar{B}$	m_0
0	1	1	$\bar{A}\,B$	m_1
1	0	0	$A\,\bar{B}$	m_2
1	1	1	$A\,B$	m_3

$$Y = F(A, B) = \bar{A}B + AB = \Sigma(1, 3)$$

2.12.5 POS Forms

- Each row has a maxterm(sum of literals)

A	B	Y	maxterm	maxterm name
0	0	0	$A + B$	M_0
0	1	1	$A + \overline{B}$	M_1
1	0	0	$\overline{A} + B$	M_2
1	1	1	$\overline{A} + \overline{B}$	M_3

$$Y = F(A, B) = (A + B)(\overline{A} + B) = \Pi(0, 2)$$

2.12.6 SOP & POS Forms

SOP – sum-of-products

O	C	E	minterm
0	0	0	$\overline{O}\ \overline{C}$
0	1	0	$\overline{O}\ C$
1	0	1	$O\ \overline{C}$
1	1	0	$O\ C$

$$E = O\overline{C}$$
$$= \Sigma(2)$$

POS – product-of-sums

O	C	E	maxterm
0	0	0	$O + C$
0	1	0	$O + \overline{C}$
1	0	1	$\overline{O} + C$
1	1	0	$\overline{O} + \overline{C}$

$$E = (O + C)(O + \overline{C})(\overline{O} + \overline{C})$$
$$= \Pi(0, 1, 3)$$

2.13 Boolean

2.13.1 Axiom

Number	Axiom	Dual	Name
A1	B = 0 if B ≠ 1	B = 1 if B ≠ 0	Binary Field
A2	$\overline{0} = 1$	$\overline{1} = 0$	NOT
A3	0 • 0 = 0	1 + 1 = 1	AND/OR
A4	1 • 1 = 1	0 + 0 = 0	AND/OR
A5	0 • 1 = 1 • 0 = 0	1 + 0 = 0 + 1 = 1	AND/OR

Dual: Replace: • with +
0 with 1

2.13.2 Theorem

#	Theorem	Dual	Name
T6	B•C = C•B	B+C = C+B	Commutativity
T7	(B•C) • D = B • (C•D)	(B + C) + D = B + (C + D)	Associativity
T8	B • (C + D) = (B•C) + (B•D)	B + (C•D) = (B+C) (B+D)	Distributivity
T9	B • (B+C) = B	B + (B•C) = B	Covering
T10	(B•C) + (B•$\overline{C}$) = B	(B+C) • (B+$\overline{C}$) = B	Combining
T11	(B•C) + ($\overline{B}$•D) + (C•D) = (B•C) + ($\overline{B}$•D)	(B+C) • ($\overline{B}$+D) • (C+D) = (B+C) • ($\overline{B}$+D)	Consensus

Warning: T8' differs from traditional algebra:
OR (+) distributes over AND (•)

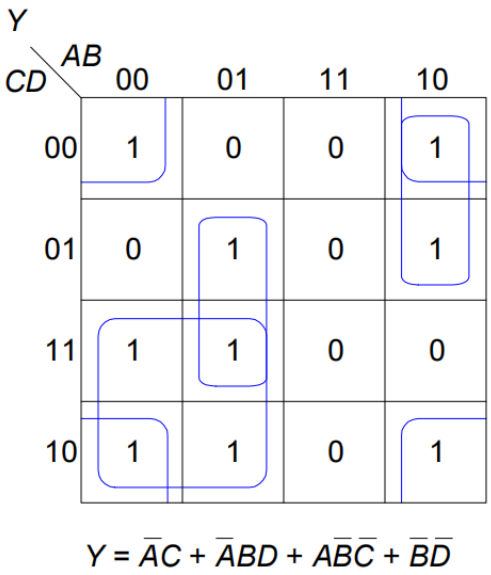
2.13.3 Simplification methods

- **Distributivity (T8, T8')** $B\,(C+D) = BC + BD$; $B + CD = (B+C)(B+D)$
- **Covering (T9')** $A + AP = A$
- **Combining (T10)** $P\overline{A} + PA = P$
- **Expansion** $P = P\overline{A} + PA$; $A = A + AP$
- **Duplication** $A = A + A$
- **"Simplification" theorem** $\overline{PA} + A = P + A$; $PA + \overline{A} = P + \overline{A}$

2.14 K-Maps

$PA + P\overline{A} = P$

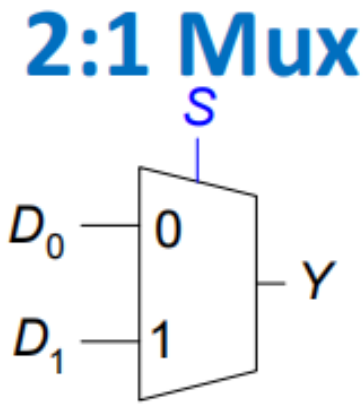
A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0



2.15 Combinational Building Blocks

2.15.1 Mutiplexers

$\lg_2^N$bit select input



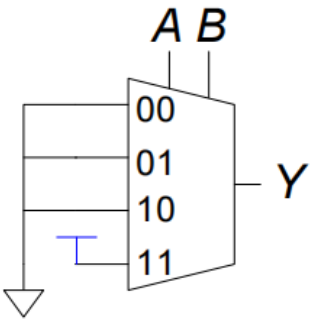
S	D_1	D_0	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

S	Y
0	D_0
1	D_1

- Logic use

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

$Y = AB$



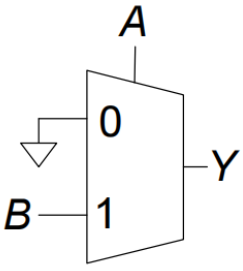
2.15.2 Decoders

N inputs, 2^N outputs

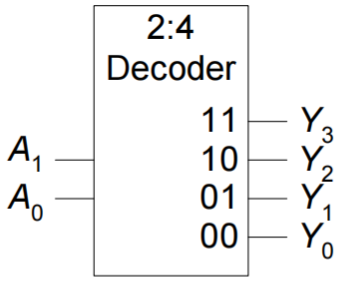
$Y = AB$

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

A	Y
0	0
1	B



- Logic use

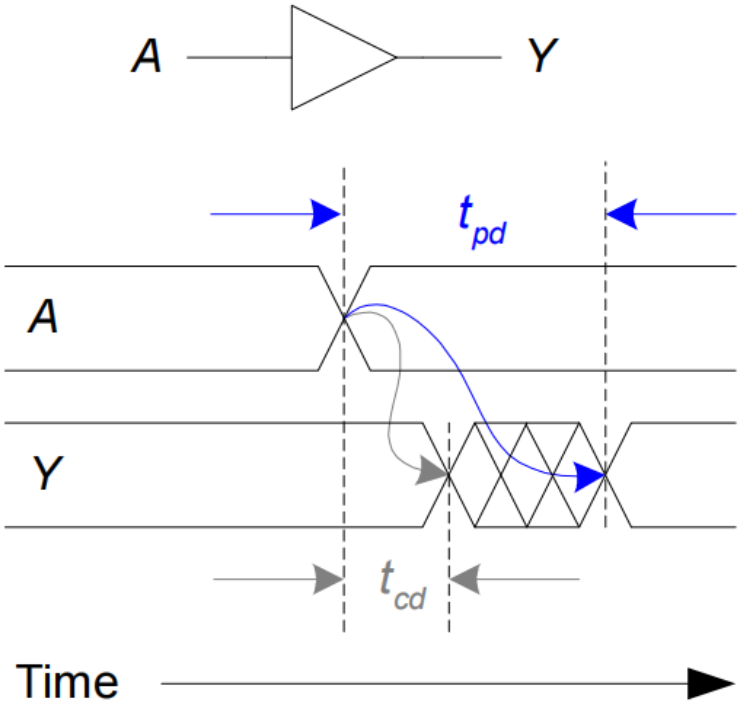


A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

2.16 Timing

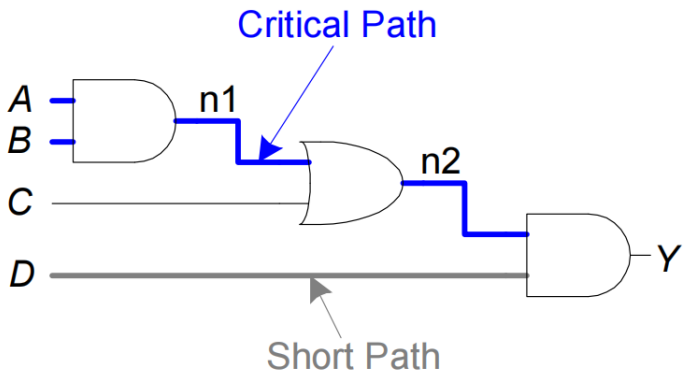
Because of wire resistance and we need time to fully charge the capacity

- t_{pd}Propagation delay: the time from input starting to change to output becoming stable
- t_{cd}Contamination delay: the time from input starting to change to output starting to change



- Different rising and falling delays
- multiple inputs and outputs
- circuits slow down when hot

2.17 Critical paths



Critical (Long) Path: $t_{pd} = 2t_{pd\_AND} + t_{pd\_OR}$

Short Path: $t_{cd} = t_{cd\_AND}$

2.18 Glitches

When a single input changes causes an output to change several times

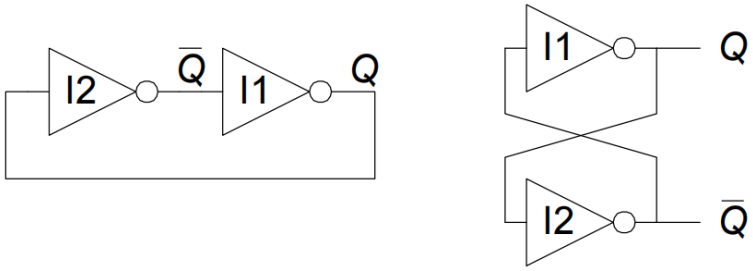
2.19 Sequential Logic

2.19.1 Definitions

- State: influence the future behavior
- Latches and flip-flops
- Synchronous sequential circuits

2.20 State elements

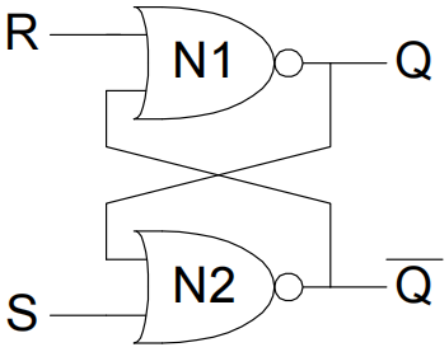
2.20.1 Bistable Circuit



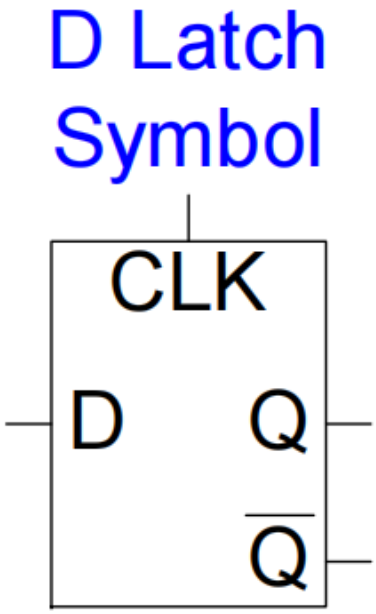
2.20.2 SR Latch

S: set, R: reset

- S: 0, R: 0: previous
- S: 1, R: 0: 1
- S: 0, R: 1: 0
- S: 1, R: 1: invalid

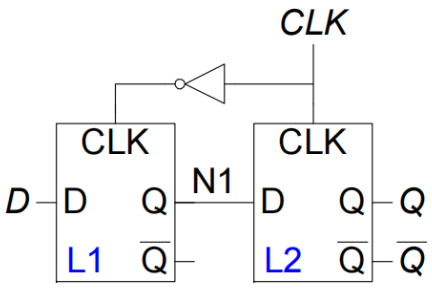


2.20.3 D Latch



- CLK: controls when the output changes
- D: controls what the output changes to

2.20.4 D Flip-Flop

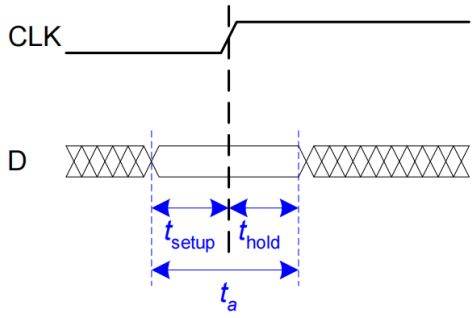


Samples D on rising edge of CLK

2.21 Timing

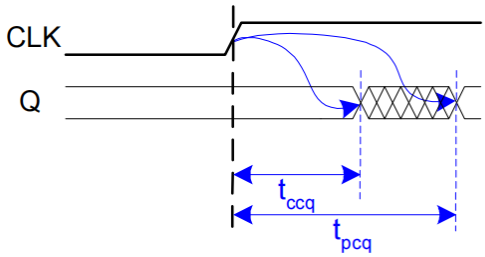
2.21.1 Input timing constraint

- t_{setup}: time before clock edge data must be stable
- t_{hold}: time after clock edge data must be stable
- Aperture timet_a: time around clock edge data must be stable



2.21.2 Output timing constraint

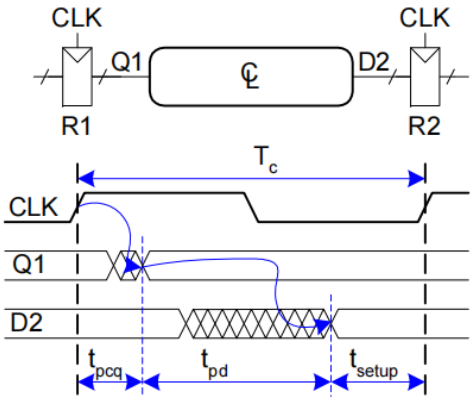
- t_{pcq}: time after clock edge that the output Q is guaranteed to be stable
- t_{ccq}: time after clock edge data might be unstable



2.21.3 Setup Time Constraint

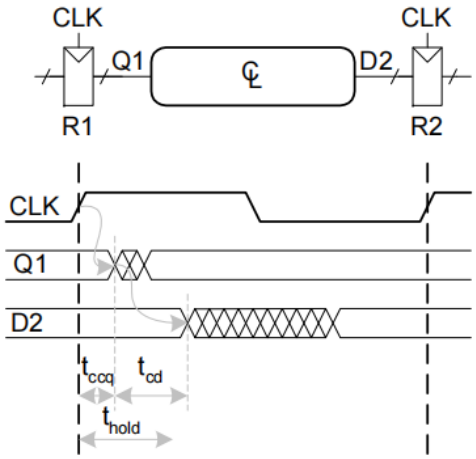
- input to register R2 must be stable at least t_{setup} before clock edge
- clock edge time ¿ Q be stable time + propagation time + set up time
- $T_c \geq t_{pcq} + t_{pd} + t_{setup}$

- $t_{pd} \leq T_c - (t_{pcq} + t_{setup})$



2.21.4 Hold Time Constraint

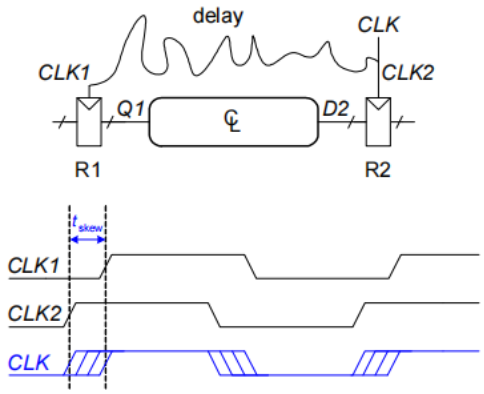
- input to register R2 must be stable at least t_{hold} after clock edge
- hold time ¡ Q1 starts to change time + D2 starts to change time
- $t_{hold} < t_{ccq} + t_{cd}$
- $t_{cd} > t_{hold} - t_{ccq}$



2.22 Clock Skew

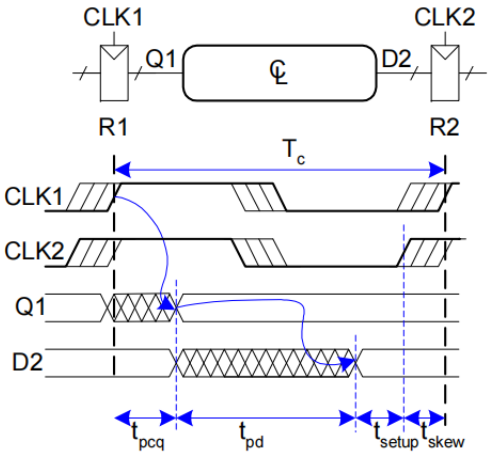
- Skew: difference between two clock edges

- Perform worst case analysis to guarantee dynamic discipline is not violated for any register



2.22.1 Setup Time Constraint with Skew

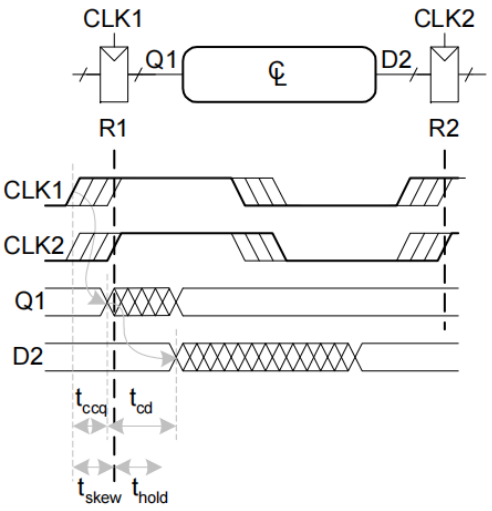
- in worst case, CLK2 is earilier than CLK1
- $T_c \geq t_{pcq} + t_{pd} + t_{setup} + t_{skew}$
- $t_{pd} \leq T_c - (t_{pcq} + t_{setup} + t_{skew})$



2.22.2 Hold Time Constraint with Skew

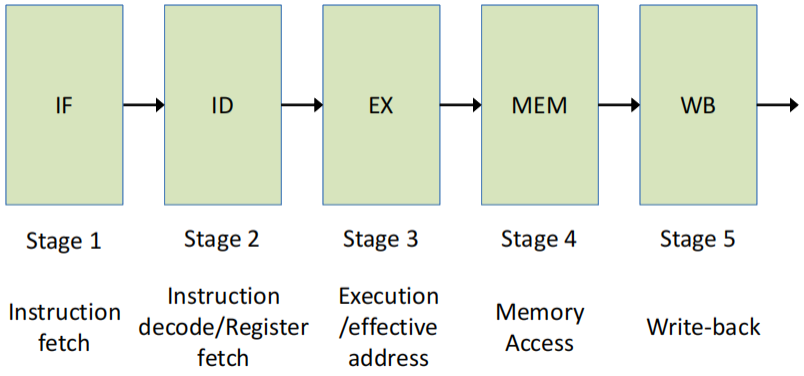
- in worst case, CLK2 is later than CLK1

- $t_{hold} + t_{skew} < t_{ccq} + t_{cd}$
- $t_{cd} > t_{hold} + t_{skew} - t_{ccq}$



3 Pipeline Overview

3.1 Simple Processor Pipeline



3.2 Five-Stage Pipeline for a RISC processor

- IF(Instruction Fetch)

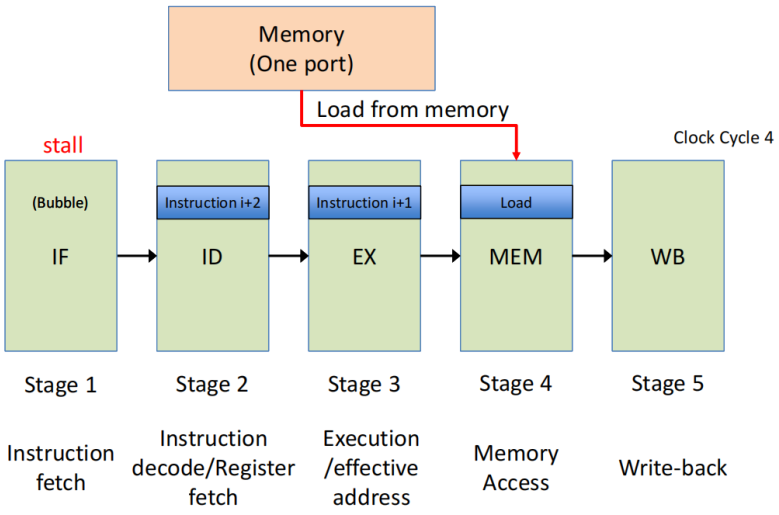
- ID(Instruction Decode/Register Fetch)
- EX(Execution/Effective address)
- MEM(Memorry access): store instruction
- WB(Write-back): write the result back into the REGISTER FILE

3.3 CPU Block Diagram

3.4 Hurdles in Pipelining

3.4.1 Structural Hazard: Resources

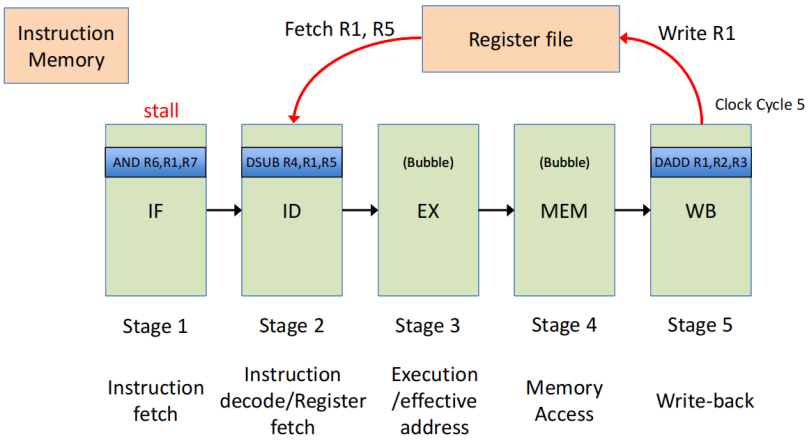
- All the overlapped instructions in pipeline requires pipelining of functional units and duplication of resources
- Resource conflict can happen when some combination of instructions cannot be accommodated.



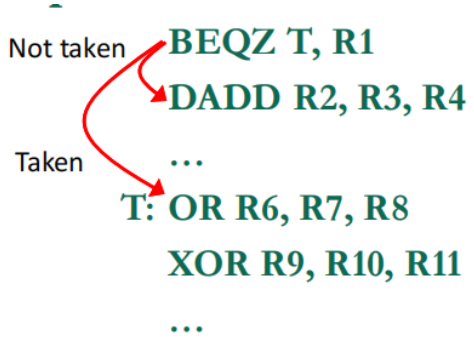
3.4.2 Data Hazard: Data dependency

```
DADD R1, R2, R3    // R2+R3 = R1
DSUB R4, R1, R5    // R1-R5 = R4
AND R6, R1, R7     // R1 & R7 = R6
OR R8, R1, R9      // R1 | R9 = R8
XOR R10, R1, R11   // R1 ^ R11 = R10
```

- All the instructions after the DADD use the result of DADD
- The value of R1 is ready at WB stage of DADD, but DSUB needs it at its ID stage



3.4.3 Control Hazard: Branch

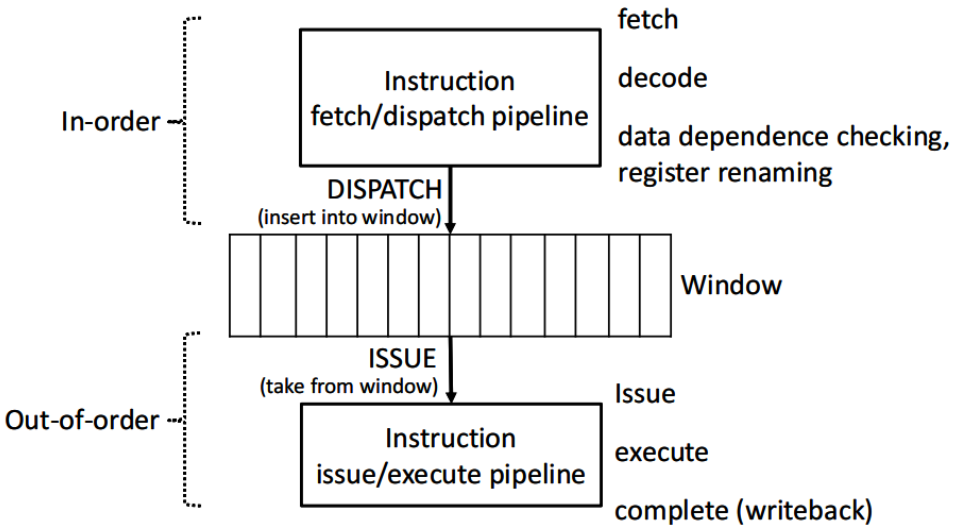


- Branch may change or not change the PC
- Which instruction to fetch after the branch?
- IF stalls until the branch condition is evaluated.

3.5 Data Dependency

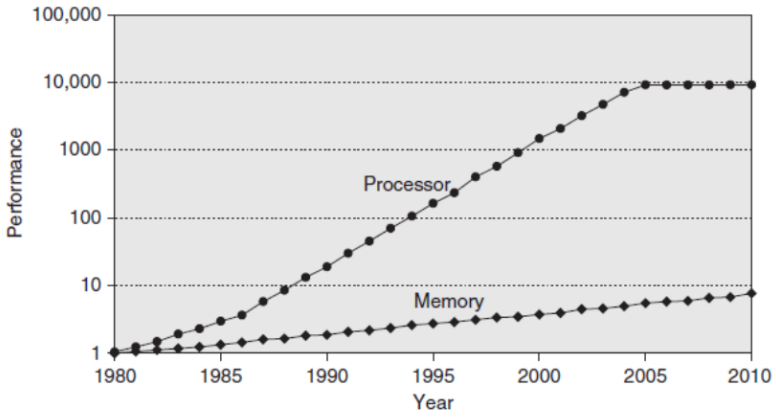
- True-dependence(pure-dependence, flow-dependence): an instruction depends on the result of a previous instruction
- Anti-dependence: an instruction requires a value that is later updated(because we are not executing instructions in order)
- Output-dependence: the ordering of instructions affects the final ouput result

3.6 Out-of-order pipelnine



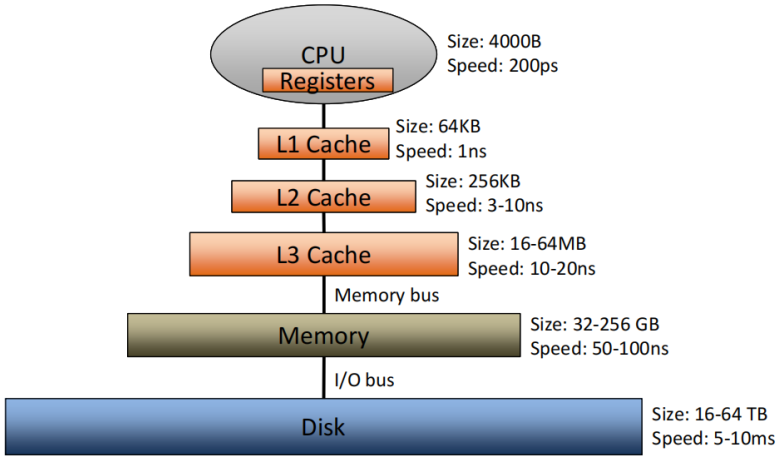
- Instruction fetching do its job in order.
- Then put these instructions into a window
- When a instruction in the window gets ready, execute pipeline executes it

3.7 Memory Wall



- Reason: The growing disparity of speed between CPU and main memory
- Given this trend, memory would become a performance bottleneck

3.8 Memory Hierarchy Design



- Large memory is slow
- processor wants large and fast memory
- Solution: take advantage of locality