Number Representation:

If N is (n+1) bit unsigned no.

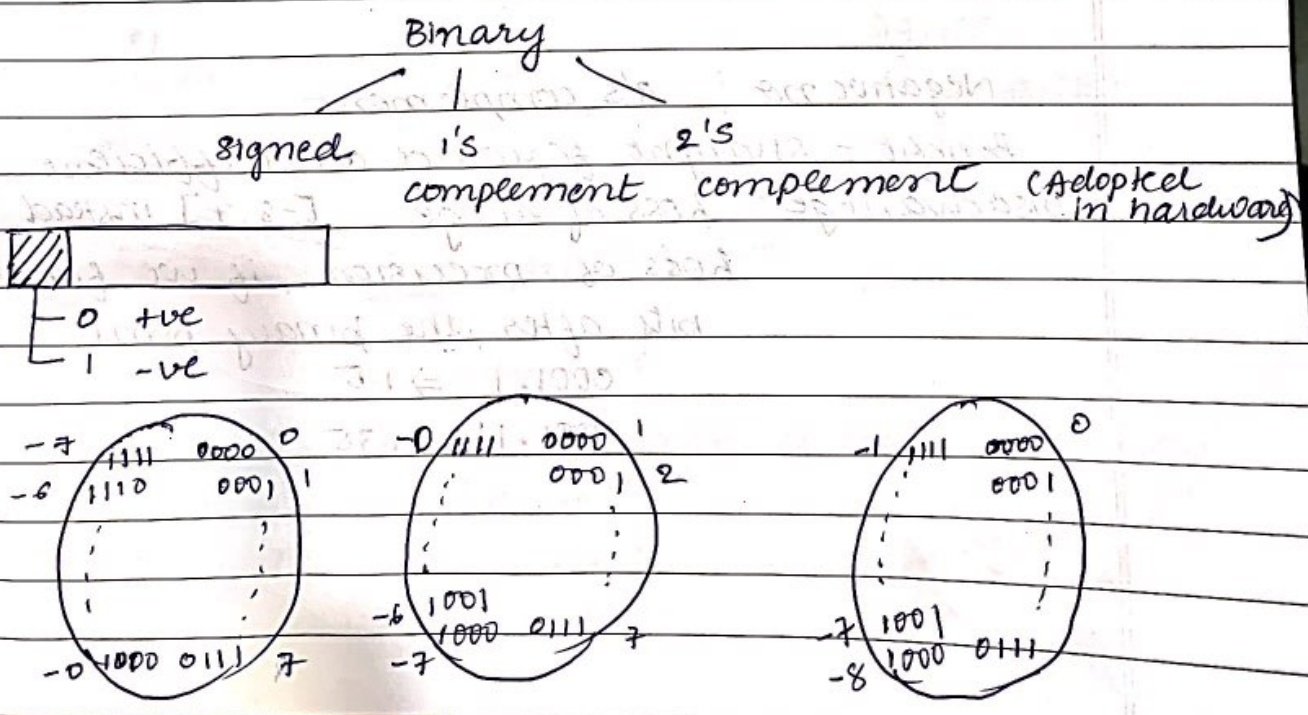
$$N = \sum_{i=0}^{n} x_i 2^i$$

$x_i \Rightarrow i^{th}$ binary bit of N.

$x_0 \Rightarrow$ LSB

$x_n \Rightarrow$ MSB

1. Decimal
2. Octal
3. Hexadecimal
4. Binary
5. BCD
6. 12 Decimal
7. Logarithmic.
8. Residual Number System.
9. CORDIC (Coordinate Rotation, Digital Computer)



Signed and 1st complement have 2 zeroes.
Thus, 2's complement is used for hardware.

eg 4 bits no.
Signed -7 to $+7$
Unsigned 0 to 15.

eg 8 bit
signed $-(2^7-1)$ to 2^7-1
unsigned 0 to 2^8-1

Binary system:

Shifting is the key.

2^2	2^1	2^0	.	2^{-1}	2^{-2}	2^{-3}

Binary point.

To define fixed point we need 2 parameters:
- width of the number representation
- Binary point position within the number

00010.110 = 2.75

Negative no: 2's complement
Benefit - straight forward and efficient
Disadvantage - loss of range [-8,7] instead [0,15]
loss of precision, if we fix the bits after the binary point
0001.1 ⇒ 1.5
0001.11 ⇒ 1.75

To avoid the compromise of range and precision with the fixed point (for scientific calculation / purpose) floating point number system is preferred

$\pi = 3.14159265 \quad \pi = 31.4159265 \times 10^{-1}$

$e = 2.71828 \quad e = 271.11828 \times 10^{-2}$

↳ normalized numbers / standered numbers

32 bit fixed point number, what is the full range? $(2^{32}-1) \sim 10^9 \times 4$

Precision? 2^{-32}

16 bits

Range: $2^{16}-1$

Precision: 2^{-16}

Floating Point Number

Bit No.	Size	Field Name
31	1	Sign(S)
23-30	8	Exponent(E)
0-22	23	Mantissa(M).

$f.No = -1^S \times (1 + mantiss) \times 2^E$

↓

Binary point

Overflow: Exponent is too large to be represented in Exp. field.

Underflow: no. is too small to be to be represented in Exp. field.

The standard expresses the exponent as a biased notation. (This is to accomadate layer binary no.s and -ve no. representable)

1 8 23.

1×2^{-1} = [0] [11111111] [000...000 - 1]

$1 \times 2^{+1}$ = [0] [00000001] [000...000]

2's complement

With the biased notation the FB number is ~~equal to~~ expressed as

$FB = (-1)^S \times (1+M) \times 2^{E-bias}$

where bias = 127.

-1 exponent will always be represented as

$-1+127 = 126$

$+1$

$+1+127 = 128$

$0+127 = 127$

Effectively it is 33 bit no.

This is a normalized number with 1 intrinsic to the number

IEEE 754 - 1985 standard

2 representation
- Single Precision (32 bit)
- Double Precision (64 bit).
- Extended bit (80 bit)

| S | Exponent | Fraction or Mantissa |

Single Precision: | 1 | 8 | 23 |

Double Precision: | 1 | 11 | 52 |

$x = (-1)^S \times (1 + \text{fraction}) \times 2^{(\text{Exponent} - \text{bias})}$

(1 + fraction) → significand

bias → 127

Floating point number

S = 0 +ve

S = 1 -ve

Normalize significant: eg 1.23×10^{-21}

$0.5 \times 10^{-11} \rightarrow 5 \times 10^{-12}$

$11.7 \times 10^{2} \rightarrow 1.17 \times 10^{1}$

| x x x . . . . . x x x |

31 22 0

$1.0 \leq |\text{significand}| \leq 2.0$

0.111.....11.

$2^{-1}\ 2^{-2} \ldots 2^{-23}$

0.5 + 0.25 + 0.125 + ...

0.5
0.25
0.125
⋮
0.999.....

Significand is fraction with "1." restored. always has a leading 1 bit. So, no need to represent it explicitly.

Exponent : excess representation.
actual exponent + bias.
This ensures that the exponent is unsigned.
Single precision = 127
Double precision = ~~+203~~ 1023

Single Precision range:

Exponents: 00000000 11111111
reserved

Smallest value of exponent : 00000001

Actual expo = 1 - bias
= 1 - 127
= -126

Fraction = 00...0000
Significand = 1.0

Smallest value : $\pm 1.0 \times 2^{-126} \simeq 1.2 \times 10^{-38}$

Largest value : expo = 11111110 = 254
actual expo = 254 - 127
= 127

Fraction = 11...1111
Significand $\simeq$ 2.0 $\pm 2.0 \times 2^{127} \simeq 3.4 \times 10^{38}$

eg

Double Precision:

Smallest value: actual expo = 1022

Fraction = 00...000

significand = 1.0

$\pm 1.0 \times 2^{-1022} \simeq \pm 2.2 \times 10^{-308}$

Largest value: actual expo = 1028 1111....110

fraction = 111...111

significand $\simeq$ 2.0

$\pm 2.0 \times 2^{1023} \simeq \pm 1.8 \times 10^{308}$

Precision: all fractional bits are significand.

Single Precision approx: 2^{-23}

Equivalent: $23 \log_{10} 2 = 23 \times 0.3$

$\simeq$ 6 decimal digits of precision

Double Precision approx: 2^{-52}

Equivalent: $52 \log_{10} 2 = 52 \times 0.3$

$\simeq$ 16 decimal digits of precision.

eg -0.75

$s = 1$

$0.11 = 0.75$

$0.11 \rightarrow 1.1 \times 2^{-1}$

$-0.75 = (-1)^1 \times (1.1) \times 2^{-1}$

$s = 1$

Fraction = 000...0001

exponent = $-1 + 127$

$= 126$

01111110

eg | 1 | 1000 0001 | 0100 0...000

↳ 129

s = 1

fraction =

significand = 1.0100....

exponent = 129 - 127 = 2

$(-1)^1 \times 1.25 \times 2^2$

-1.25×4

-5.0

Adder:

Arithmetic operations relating to input and output is

$$x + y + c_{in} = 2^n c_{out} + z$$

For n bit adder

Inputs:

$$\underline{x} = (x_{n-1}, x_{n-2}, \ldots x_0) \quad x_j \in \{0,1\}$$

$$\underline{y} = (y_{n-1}, y_{n-2}, \ldots y_0) \quad y_j \in \{0,1\}$$

$$c_{in} = \{0,1\}$$

Outputs:

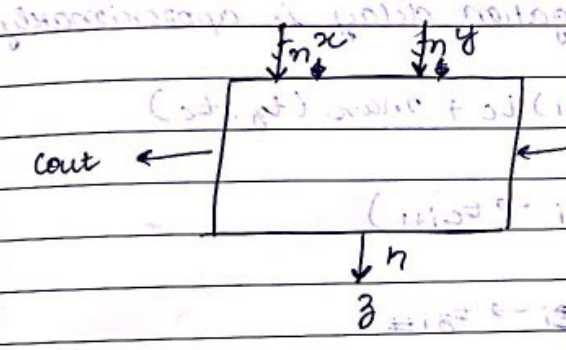
$$\underline{z} = (z_{n-1}, \ldots z_0) \quad z_j \in \{0,1\}$$

Functions: $z = (x + y + c_{in}) \bmod 2^n$

$$c_{out} = \begin{cases} 1 & \text{if } x+y+c_{in} > 2^n \\ 0 & \text{otherwise} \end{cases}$$

if $n=5$

x	y	cin	z	cout
11	12	1	$(11+12+1) \bmod 32 = 24$	0
20	15			



Ripple carry Adder:

i/p : $x_i, y_i, c_i \in \{0,1\}$.

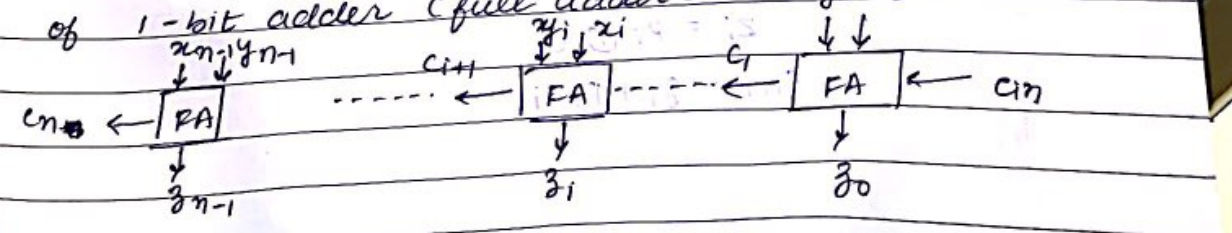
o/p : $z_i, c_o \in \{0,1\}$

Function :

$$z_i = (x_i + y_i + c_i) \bmod 2$$

$n=1$ $\quad c_{i+1} = \begin{cases} 1 & \text{if } x_i+y_i+c_i \geq 2 \\ 0 & \text{otherwise} \end{cases}$

For n bit ripple carry adder module, if it is a multilevel network formed by correct connection of 1-bit adder (full adder)



n bit adder (RCA)

Critical Path :

Begins in c_{in}, traverses all the module, terminates in C_{out} or z_{n-1}.

Delay :

Worst case propogation delay is approximately :

$$t(network) = (n-1)\,t_c + \max(t_z, t_c)$$

$$t_c = delay(t_{ci} \rightarrow t_{ci+1})$$

$$t_z = delay(t_{zi} \rightarrow t_{zi+1})$$

$$z_i = x_i \oplus y_i \oplus c_i$$

$x_i y_i' c_i' + x_i' y_i c_i' + x_i' y_i' c_i + x_i y_i c_i$

is 1 when $x_i + y_i = 2$ $(x_i y_i = 1)$,

or when $x_i + y_i = 1$ and $c_i = 1$

$$(x_i \oplus y_i).c_i = 1$$

$$c_{i+1} = x_i y_i + (x_i \oplus y_i).c_i$$

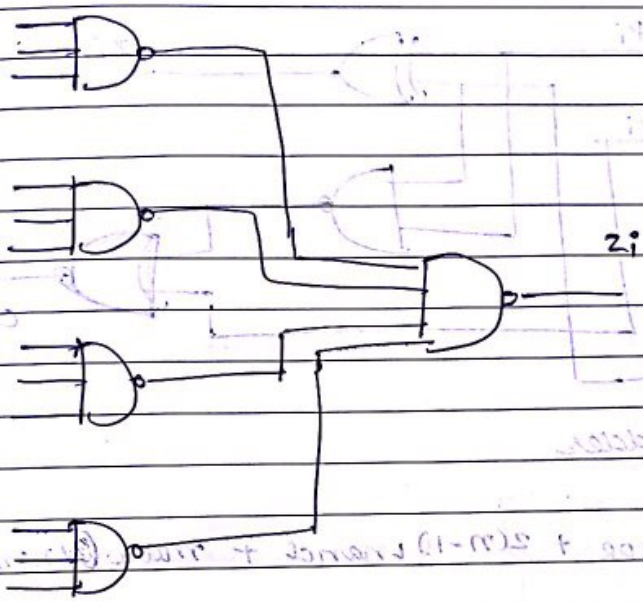
Variable

Propogate : $P_i = x_i + y_i$

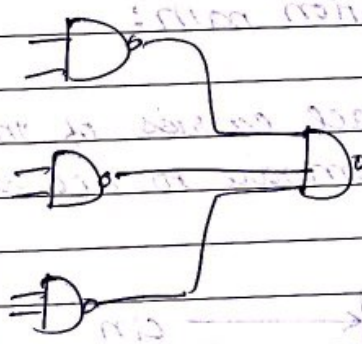
Generate : $g_i = x_i y_i$

$$z_i = P_i \oplus c_i$$

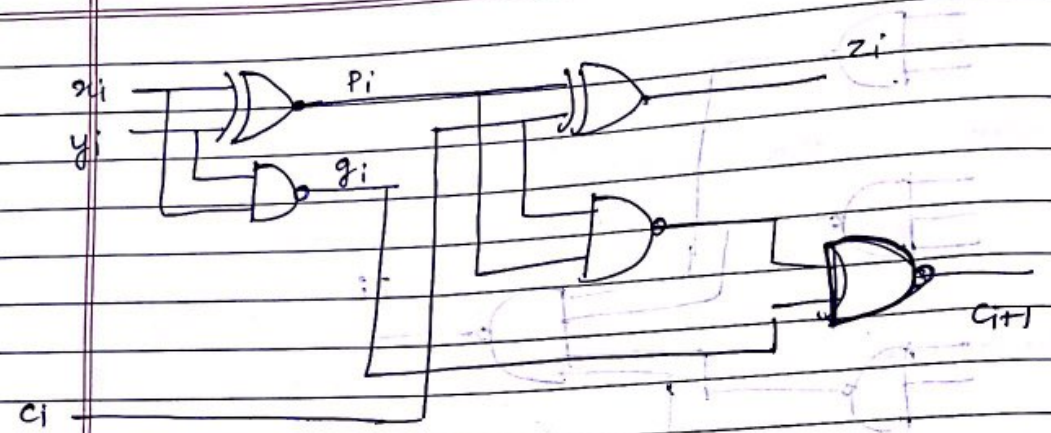
$$c_{i+1} = g_i + P_i c_i$$



$C_{i+1} = x_i y_i + x_i c_i + y_i c_i$



(Architecture #1)



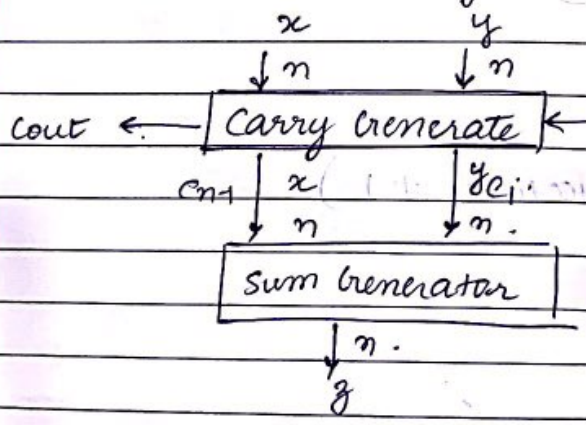
n: n bit adder

Delay $t_p = t_{XOR} + 2(n-1)t_{nand} + \max(t_{nand}, t_{XOR})$

Carry Look Ahead Adder:

Breaks the carry propogation path:

→ Decrease the values of all carries of the FA.
→ Then simultaneously compare to all reset bits.



$C_{i+1} = g_i + P_i C_i$

$g_i = x_i y_i$

$P_i = x_i \oplus y_i$

$c_1 = g_0 + P_0 C_0$

$c_2 = g_1 + P_1 C_1$

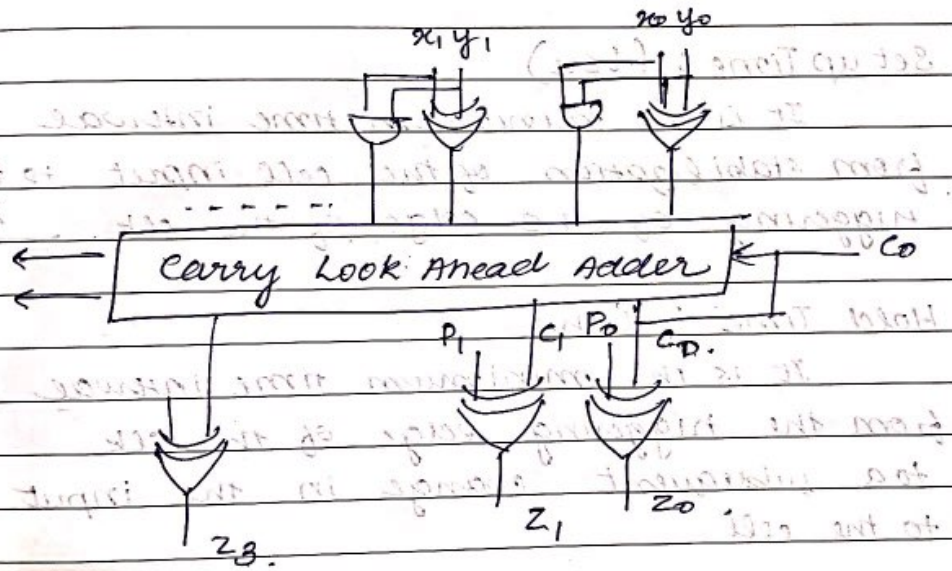
$= g_1 + g_0 P_1 + P_1 P_0 C_0$.

$c_3 = g_2 + P_2 C_2$

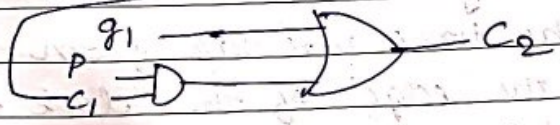
$= g_2 + g_1 P_2 + g_0 P_1 P_2 + P_0 P_1 P_2 C_0$

$Z_i = x_i \oplus y_i \oplus C_i$

$= P_i \oplus C_i$



$c_1 = g_0 + P_0 C_0$



4 CLA

CLG-4

$C_4 = G_0 + P_0 C_0$.

$C_8 = G_1 + P_1 G_0 + P_1 P_0 C_0$

$C_{12} = G_2 + G_1 P_2 + G_0 P_1 P_2 + P_0 P_1 P_2 C_0$

$C_{16} = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_0$

Set up Time: (t_{su})

It is the minimum time interval from stabilization of the cell input to the triggering of the edge of the clk.

Hold Time: (t_h)

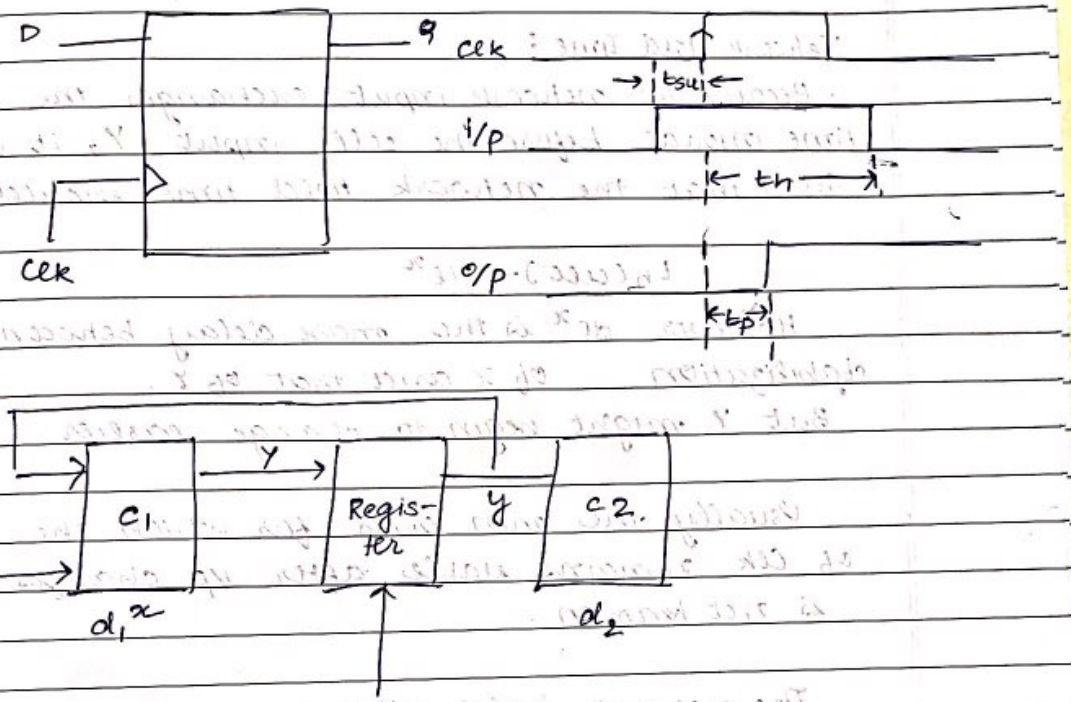
It is the minimum time interval from the triggering edge of the clk to a subsequent change in the input to the cell.

Pulse width: (t_w)

It is the minimum width of the synchronizing clk pulse.

Propagation Delay: (t_p)

It is the time interval from triggering of the edge of the clock to the stabilization of the new state (output.

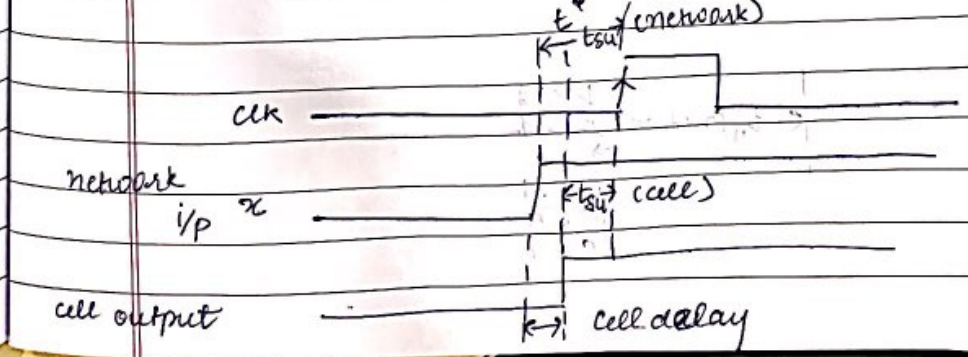


d_1^x = delay of the combinational network C_1.

$d_1 y$ = wrt input x and to the input from the state register y

1. Network setup time : It is the time i/p to the cell to be stable at t^+

$$t_{su(net)}^{x} = d_e^x + t_{su(cell)}$$



Network Hold Time:

Because the network input exchanges the Time interval before the cell input Y, it would seem that the network hold time should be

$$t_h(cell) - de^x$$

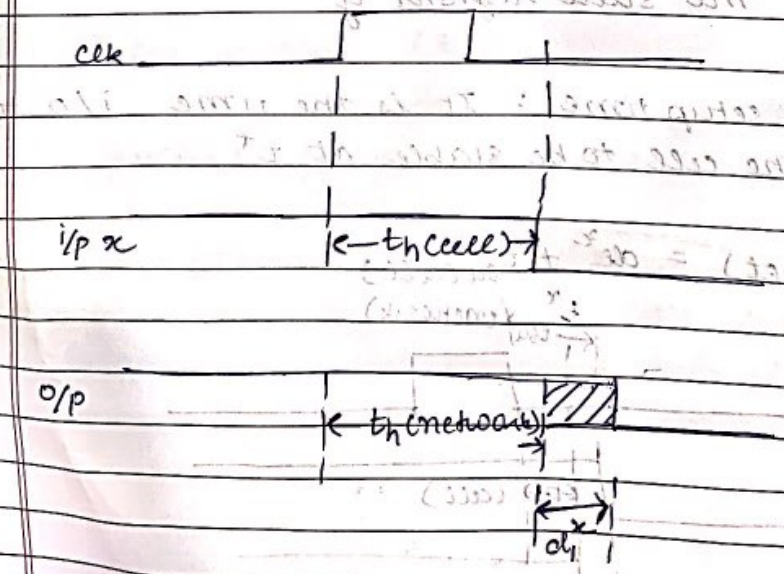
However de^x is the max delay between the stabilization of x and that of Y.

But Y might begin to change earlier.

Usually the min terms for which the O/p of clk remains stable after i/p changes is not known.

The network hold time is ~~consequ~~ conservatively equal to the hold time of the cell.

$$t_h(network) = t_h(cell)$$

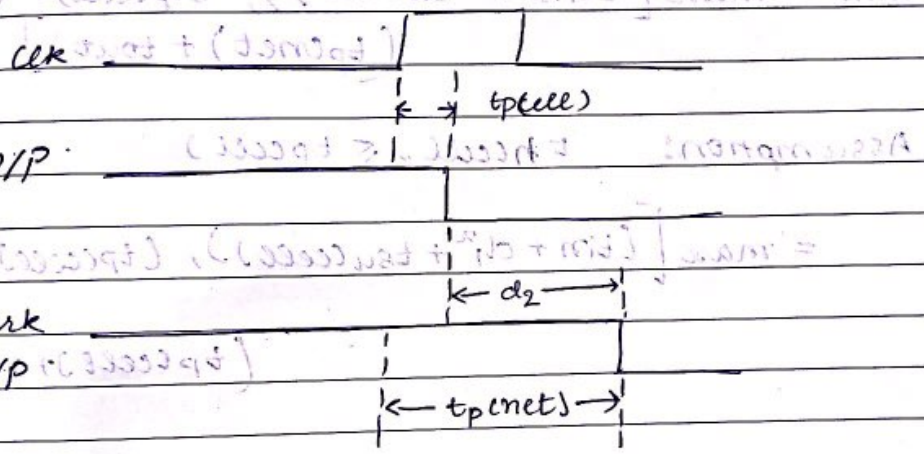


Network Propogation Delay:

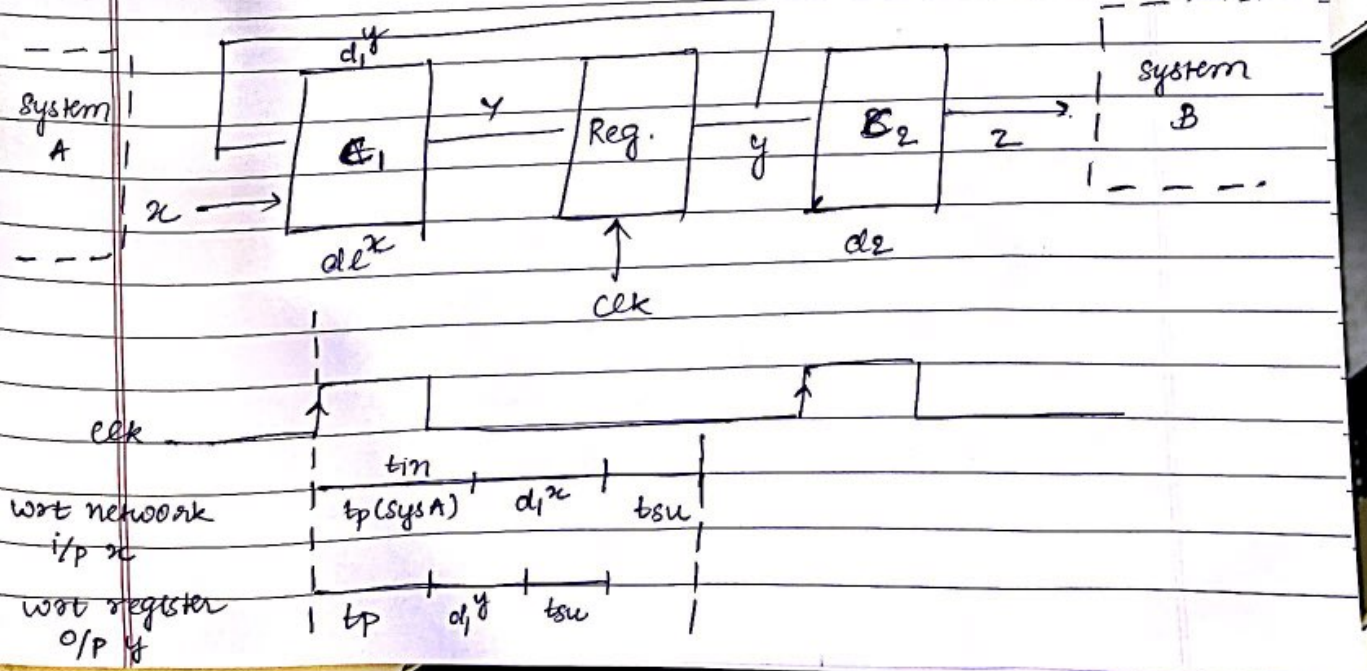
C_2 has a delay $= d_2$

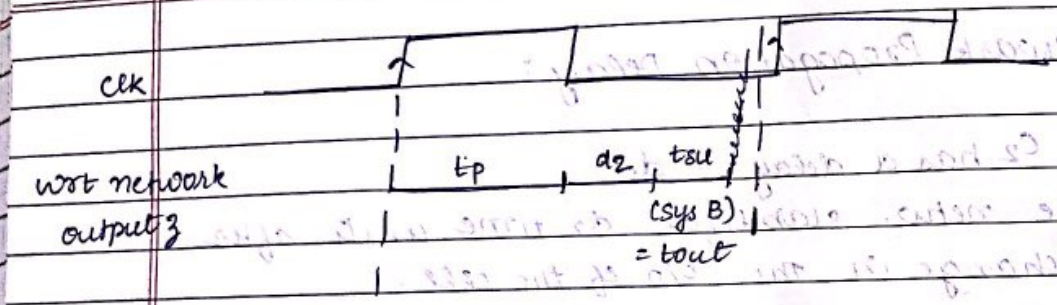
The netw. stabilizes d_2 time units after a change in the o/p of the cell.

$$t_p(\text{net}) = t_p(\text{cell}) + d_2$$



Maximum Clock Frequency:





Worst case path : Maximum clk freq (1st)

$$T_{min} = \max\left[\left(t_{in} + t_{su}^{x}(cnet)\right),\ \left(t_{p}(ccell) + t_{su}^{y}(cnet)\right),\ \left(t_{p}(cnet) + t_{out}\right)\right]$$

Assumption: $t_h(ccell) \leq t_p(cell)$

$$= \max\left[\left(t_{in} + d_1^{x} + t_{su}(ccell)\right),\ \left(t_p(ccell) + d_1^{y} + t_{su}(ccell)\right),\ \left(t_p(ccell) + d_2 + t_{out}\right)\right]$$