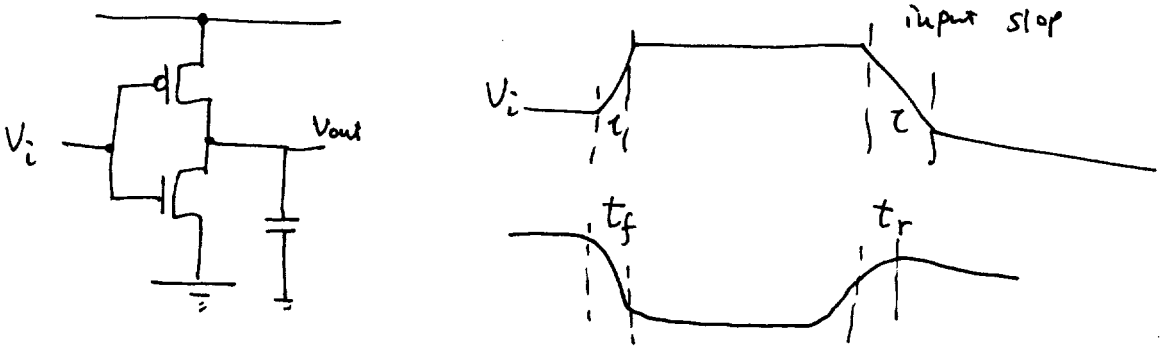
RT and Algorithmic-level optimization for Low power

- A very new area of research
- Target of the optimization is dynamic (switching) component of power.
- **RT-level power optimization**
  - * architecture-driven voltage down scaling
  - * Reduction of total switching capacitance of the design
- **Architecture-driven voltage down scaling**

$$P = \frac{1}{2} E_{sw} \cdot C_L V_{dd}^2 f$$

∴ Try to reduce Vdd.

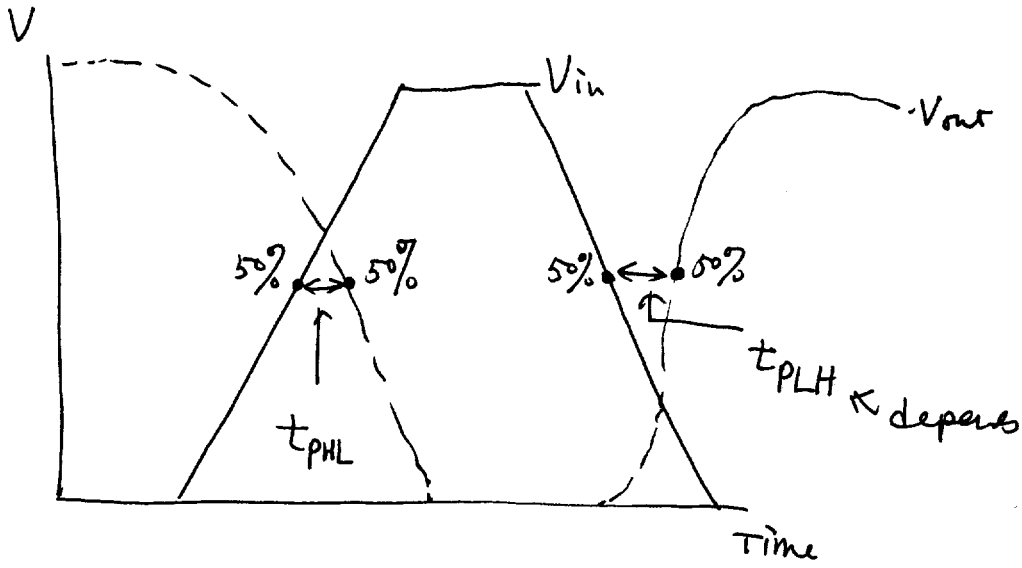
But, this affects circuit speed.



Three diffent delays:

t_f. t_r. t_{pHL}

Circuit speed



$$t_{PLH} = \frac{C_L}{\beta_p (V_{DD} - V_{TP})} \left[\frac{2V_{TP}}{V_{DD} - V_{TP}} + \ln\left(\frac{3V_{DD} - 4V_{TP}}{V_{DD}} \right) \right] + \left[1 + \frac{2V_{TP}}{V_{DD}} \right] \frac{\tau}{6}$$

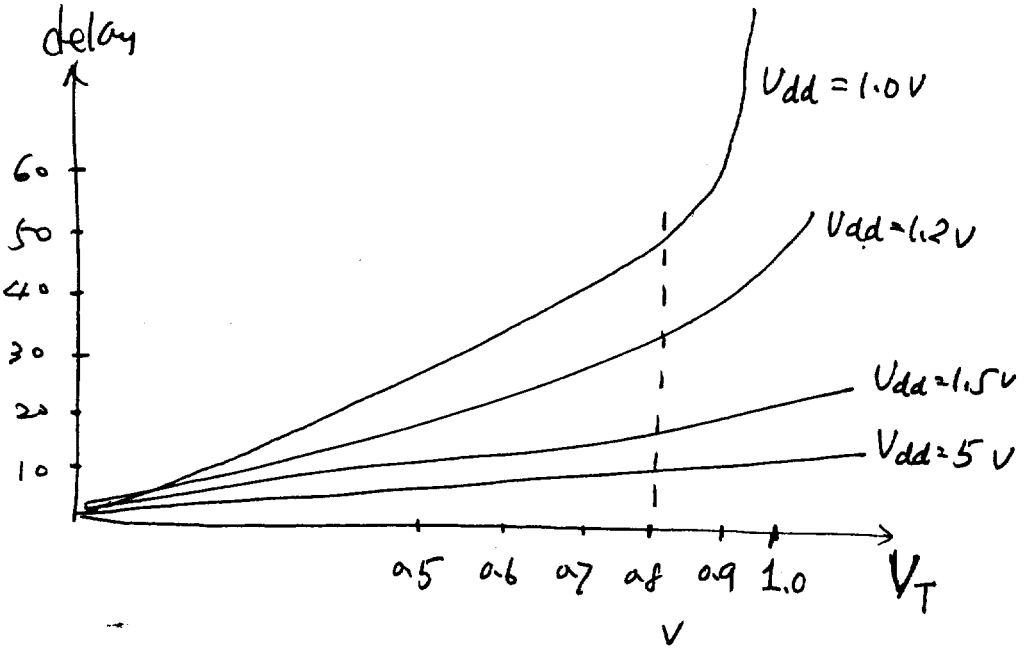
where $\beta_p = \frac{W_p}{L_p} \frac{\varepsilon_{ox} \mu}{t_{ox}}$

t_{PLH} can be approximated by $\dfrac{C_L V_{dd}}{k (W/L) (V_{dd} - V_T)^2}$

p or n depends on t_{PLH} or t_{PHL}.

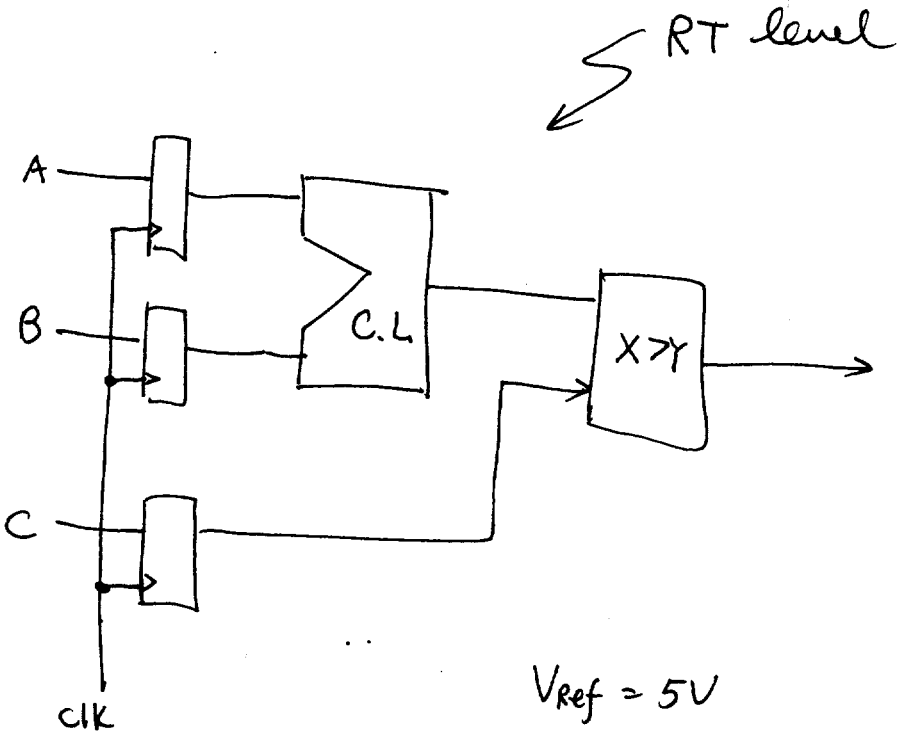
$\therefore$ $V_{dd} \downarrow \quad T \uparrow$

when $V_{dd} \to V_{T0} \Rightarrow T \uparrow$ dramatically



- The increase in delay due to Vdd reduction must be compensated through architecture modification
- Use parallel/pipeline architecture to maintain the same throughput.

Example:



$V_{Ref} = 5V$

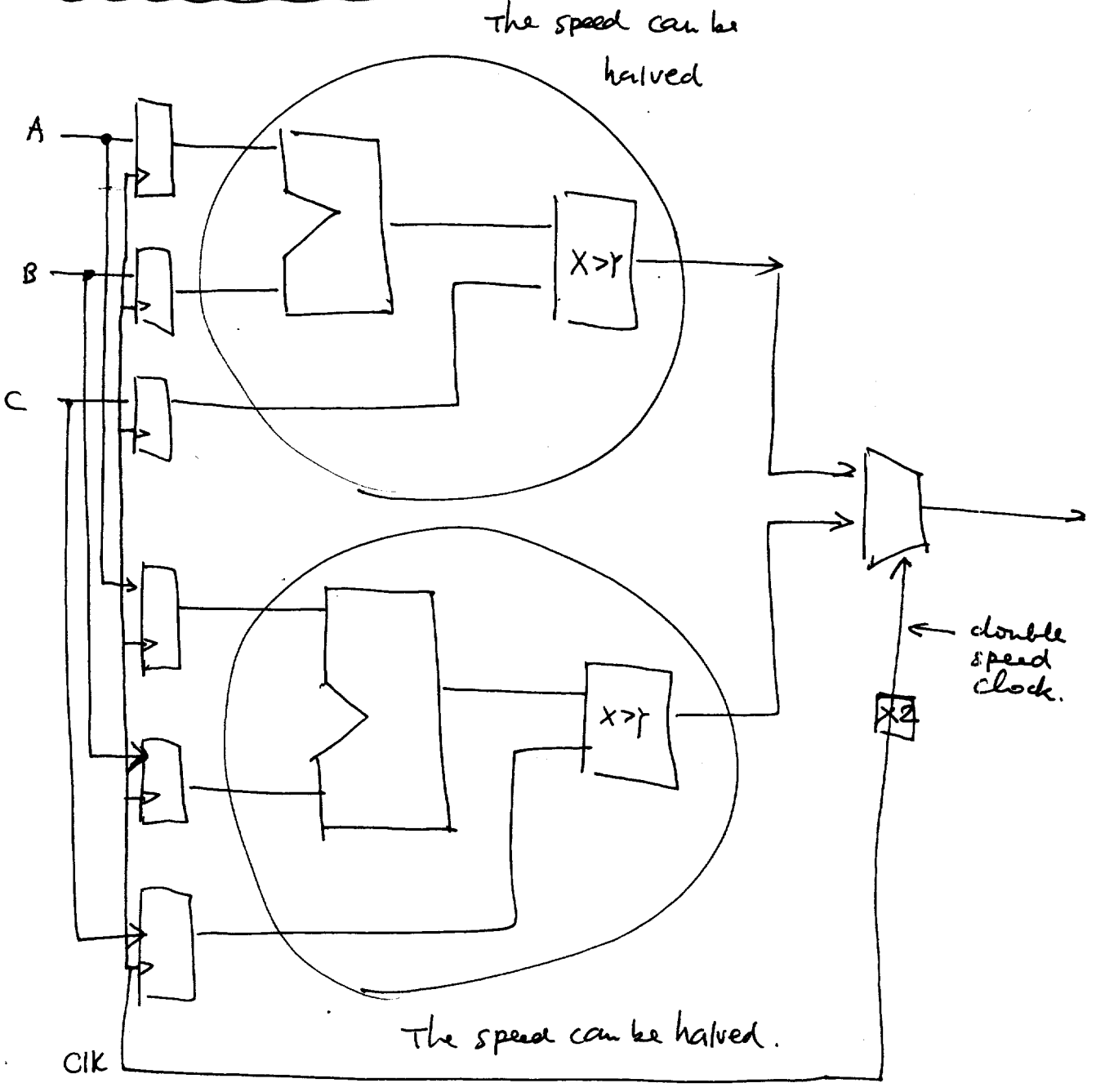
Worst delay = 25 ns

$\therefore T_{ref} = 25$ ns.

$P_{Ref} = \frac{1}{2} C_{Ref} V_{Ref}^2 f_{Ref}$ (switching activity is not ∴ A, B, C are changed, ∴ ignored). ∴ random #s

where C_{Ref} is total effective capacitance of the ckt.

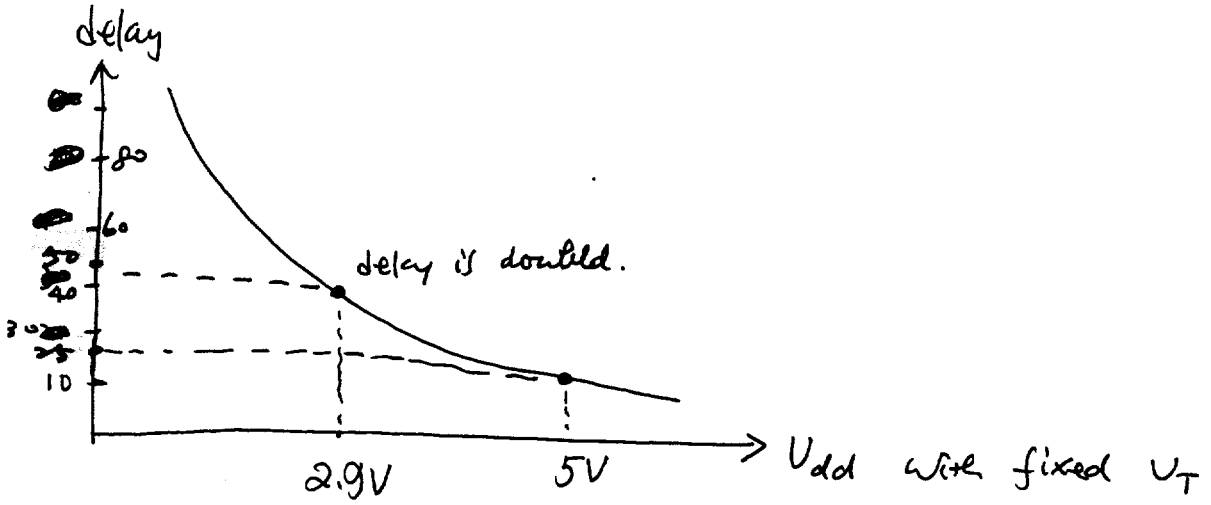
Use parallel architecture



∴ CLK can be reduced to $T_{ref} > 50$ ns

with the same throughput.

- How much can the V_{DD} be reduced?



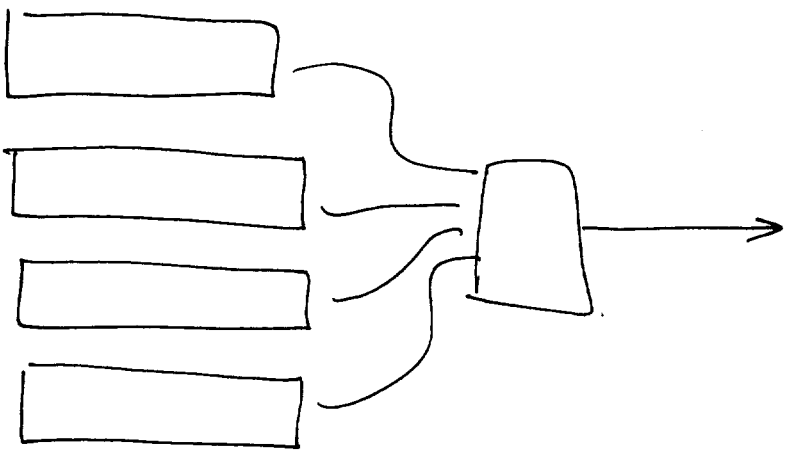
$$\Rightarrow V_{par} = 2.9V = 0.5 f V_{Ref}$$

$$C_{par} \approx 2.15\, C_{Ref}$$

$$\Rightarrow P_{par} = \frac{1}{2} C_{par} V_{par}^2 f_{par} = \frac{1}{2} (2.15\, C_{Ref})(0.5 f_{Ref})^2 \cdot \frac{f_{ref}}{2}$$

$$\approx 0.36\, P_{ref}$$

- How much parallelism can be used?



can be 4 times slower!?

Can V_{dd} be highly reduced?

Problem: When Vdd approaches V_T,

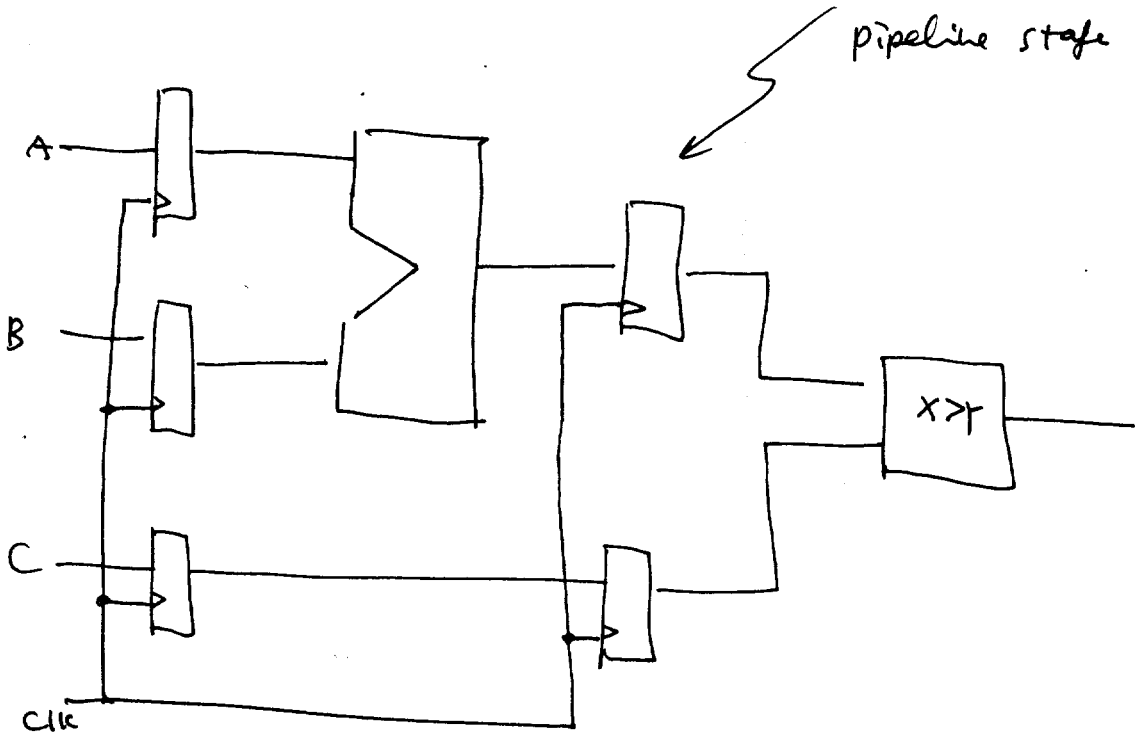
→ delay ↑ rapidly.

→ much more duplication of the ckt is required

→ power ↑ more than power saved!!

→ parallelism does not pay off. when $V_{DD} \to V_T$.

Pipeline implementation



- The critical path has been reduced.

- Time = max [T_{adder}, $T_{comparator}$]

  Worst delay of adder & comparator, respectively.

Assume $T_{adder} = T_{comparator} = 12.5\ ns$

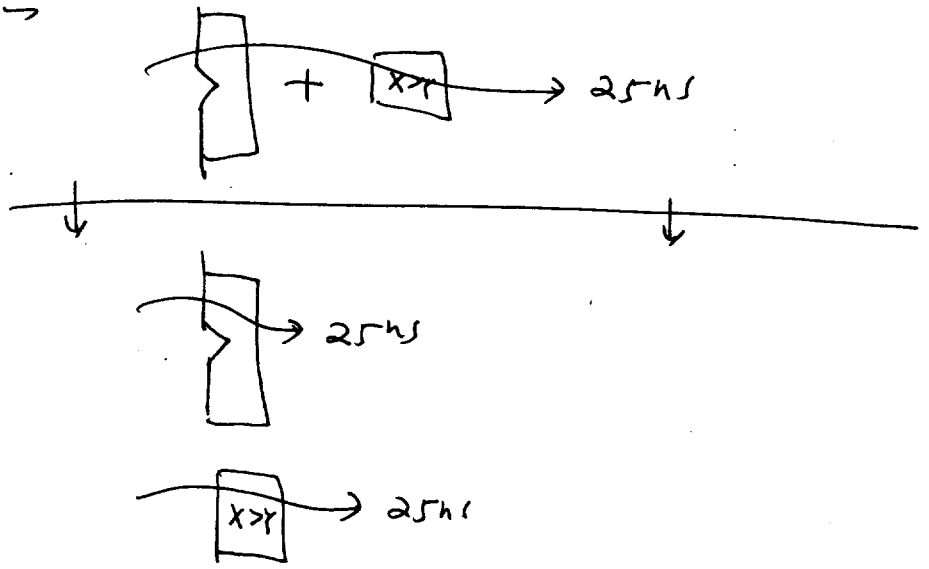
The clock ~~speed~~ period can be 12.5 ns.

But, we just like to keep the same speed

$\therefore$ Clock speed (period) is still 25 ns

$\therefore$ The ckt speed of each one can be slowed by 2 times. (25 ns)

$\therefore$



$\therefore$ Voltage supply of each ckt can be reduced

Again, $V_{pipe} = 2.9V = 0.5 f$ Ref by checking the curve

$C_{pipe} = 1.15\ C_{ref}$ (increased slightly only)

$f_{pipe} = f_{ref}$ (still work with the same speed)

$$P_{pipe} = \frac{1}{2} C_{pipe} V_{pipe}^2 f_{pipe}$$

$$= \frac{1}{2} (1.15 C_{ref}) (0.5^2 V_{ref})^2 f_{ref}$$

$$\approx 0.39 P_{Ref}$$

∴ power saved is about the same as in parallel architecture

But, area penalty is much smaller.

Further, pipelining causes reduction of ~~setup~~ ckt depth

⇒ reduce the power dissipation by glitches.

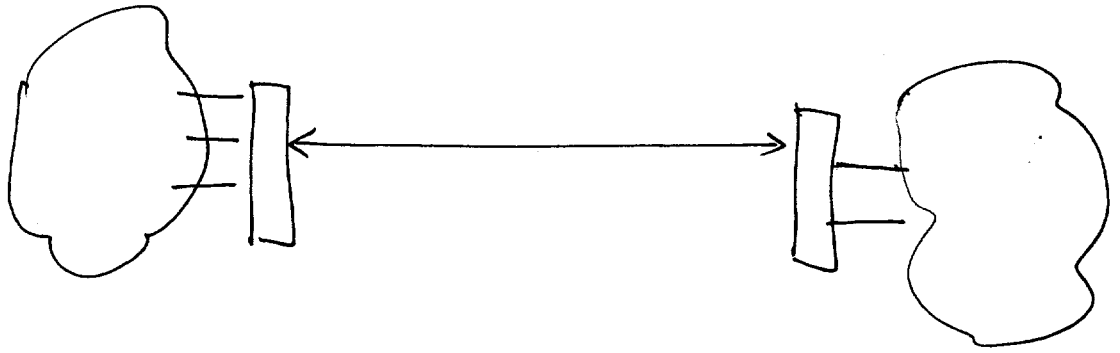
∴ Pipelining is prefered if possible.

- Merging parallel & pipelining ⟹ Reduce even more significant power

$$\underline{\underline{P_{PP} \approx 0.2 P_{Ref}}}$$

· Effective Capacitance Reduction

* Data representation



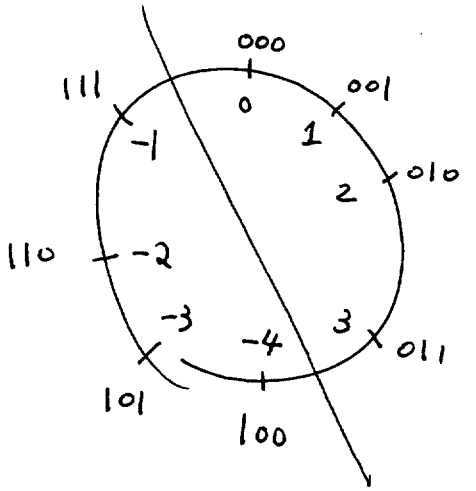
· Try to find a good data representation s.t the effective Capacitance can be reduced.

⇑

Average switching activity × total capacitive load

· Two's Complement representation

$\mathbf{2}_2 - 3 = ?$ → $2 + (-3) = -1$



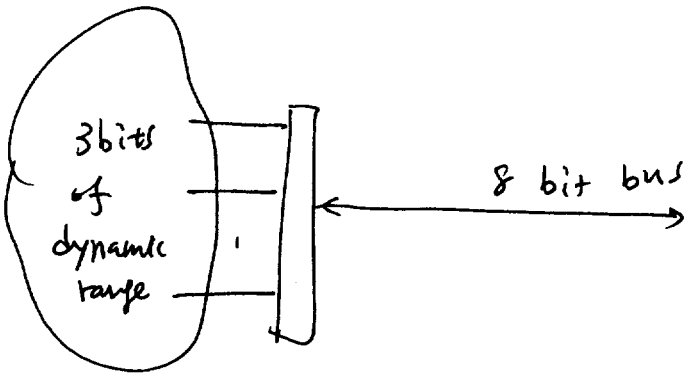
$$\begin{array}{r} 010 \\ 101 \\ \hline 111 \leftarrow -1 \end{array}$$

· Very useful in DSP and up chip

°° make arithmetic operations simple.

· Sign-bit extension.

Example:



111 = -1 must be extended to 11111111

(If no sign extension, ① sign of # will be changed. ② 2's complement sign extension properties cannot be held.)

- This happens when dynamic range << actual bit width.
- If the numbers being transferred or manipulated are switching around zero frequently + do not use the entire bit width
  ⇒ may incur lots of transitions.

$+1 = 00000001_2$ $\quad -1 = 11111111_2$

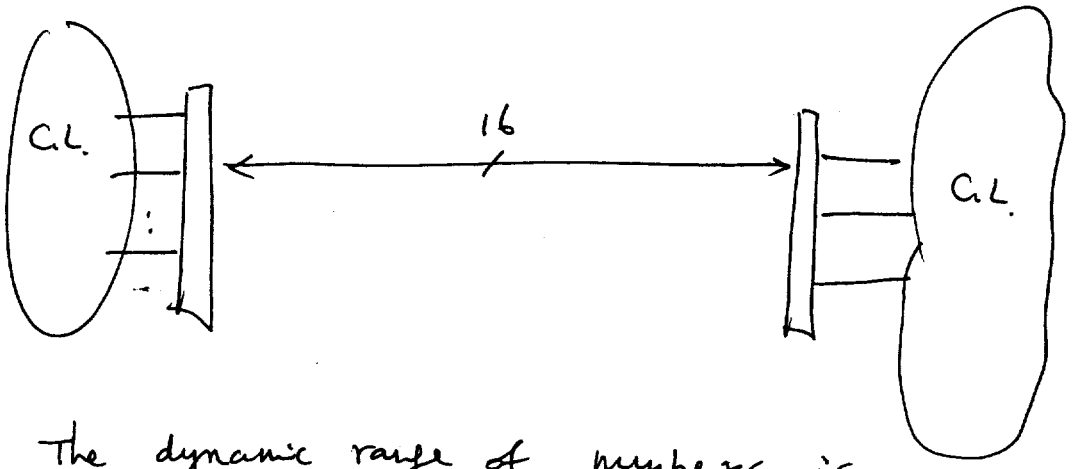
- However, if the #s are represented by sign and magnitude notation ⇒

$+1 = 00000001_2$ $\quad -1 = 10000001_2$

Very small # of switchings.

Conclusion: The degree of correlation between data being processed influences the power dissipation.

- Case study.



The dynamic range of numbers is mostly

$[-2^{11}+1,\ 2^{11}-1]$.

$\therefore$ only 12 bits out of the 16 bits are used.

Problems: • Try to determine the transition probability, P, for each of the 16 bus lines for

① 2's complement representation

② Sign and magnitude representation.

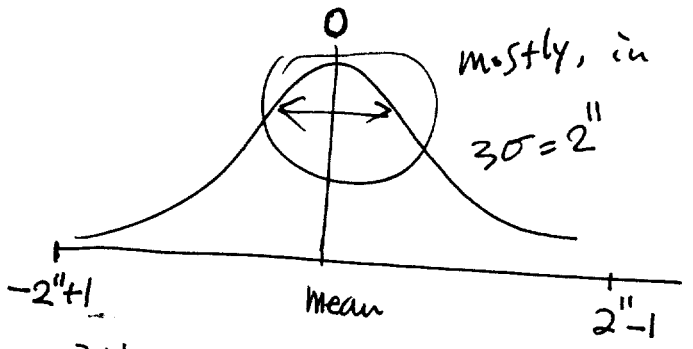
- Data correlations

① No correlation

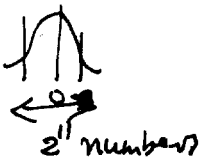
② positive correlation : p (2 consecutive #'s have same sign) ↑

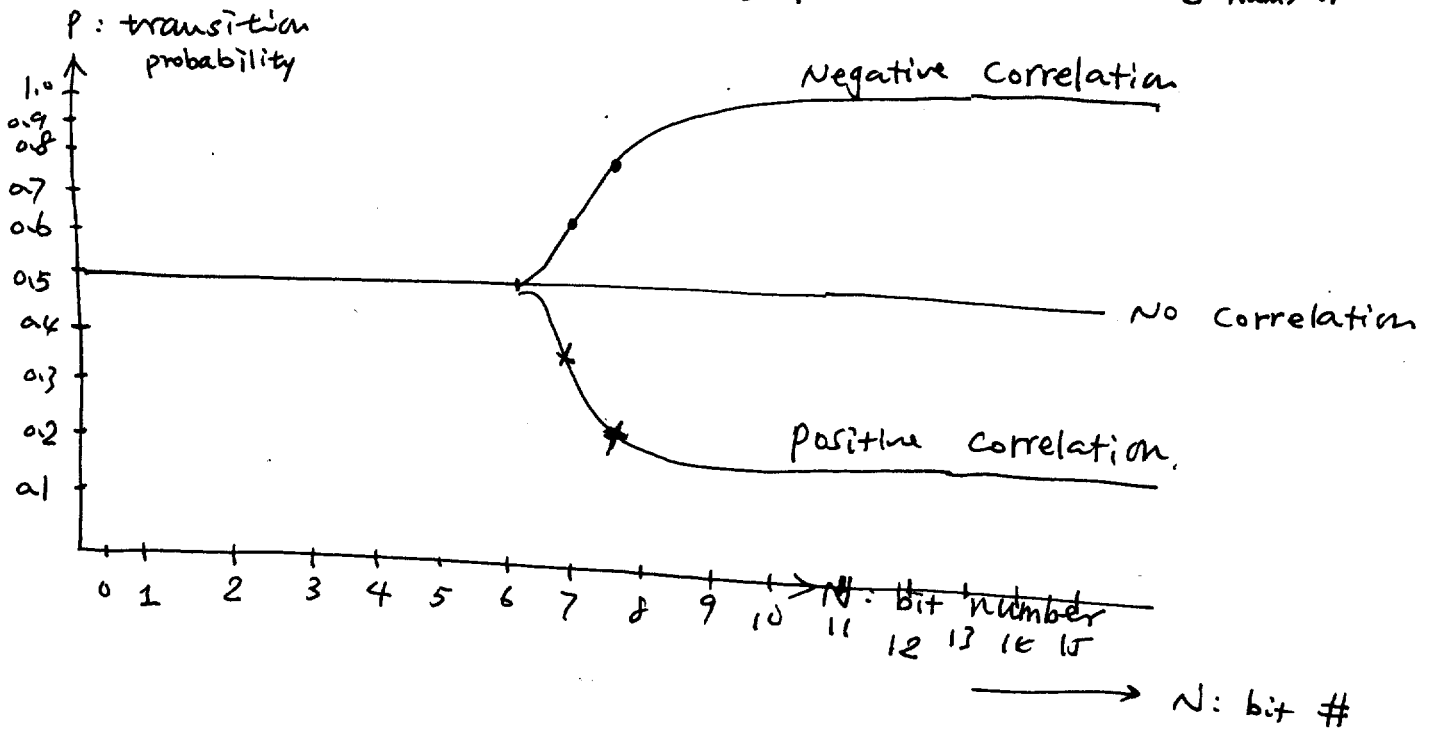
③ Negative Correlation: P(2 consecutive #'s have same sign) ↓

• Data distribution: Gaussian distribution

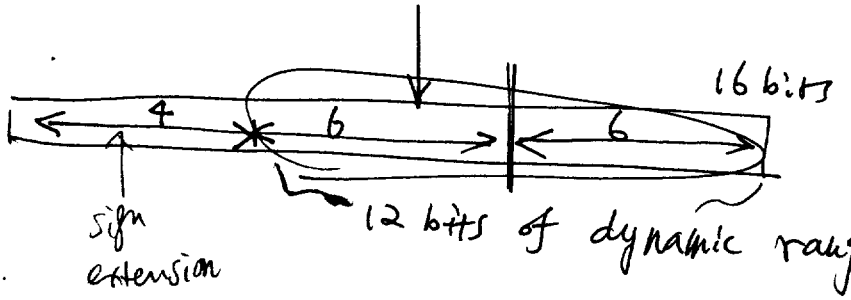


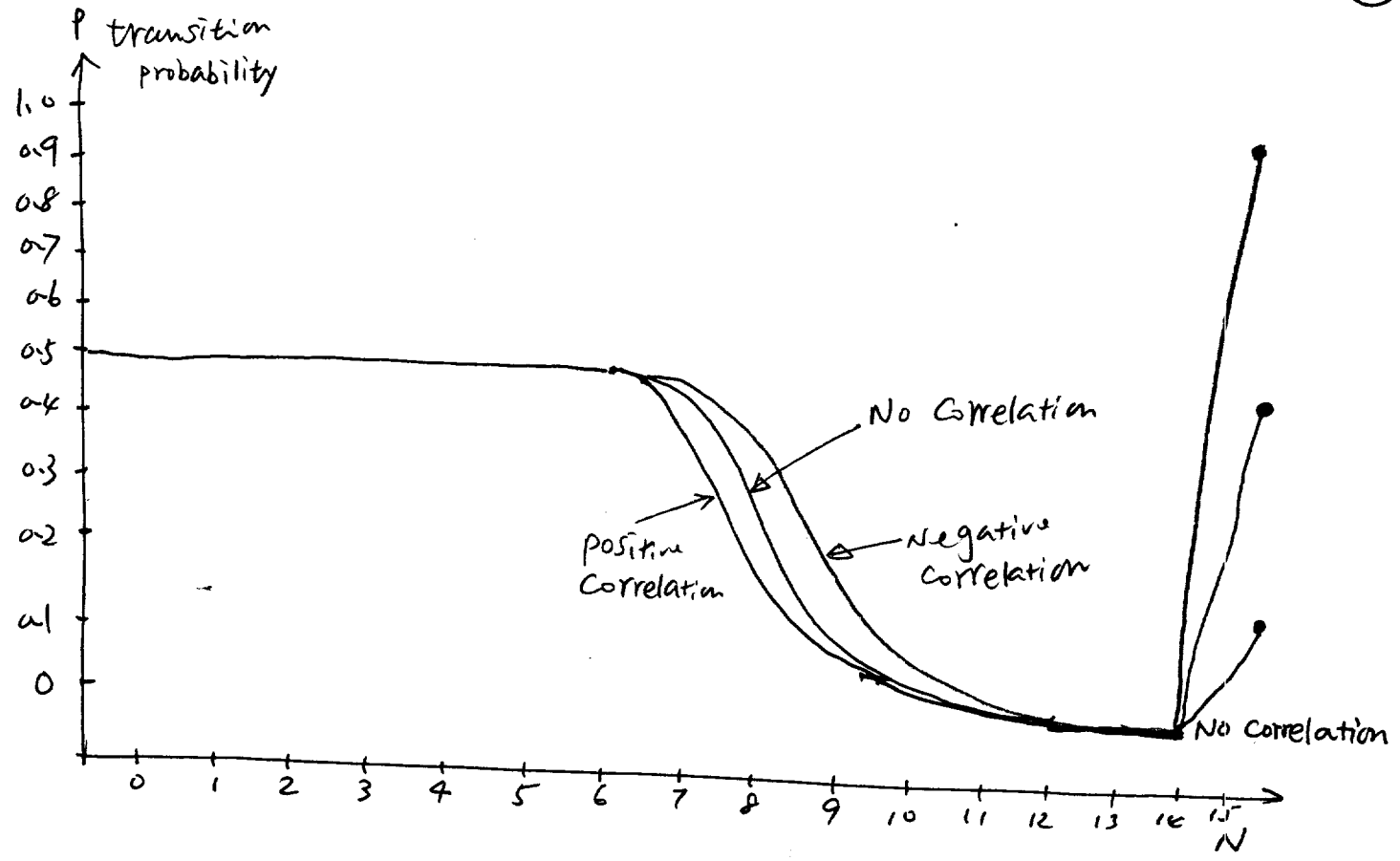
• 93.32% that data will be distributed

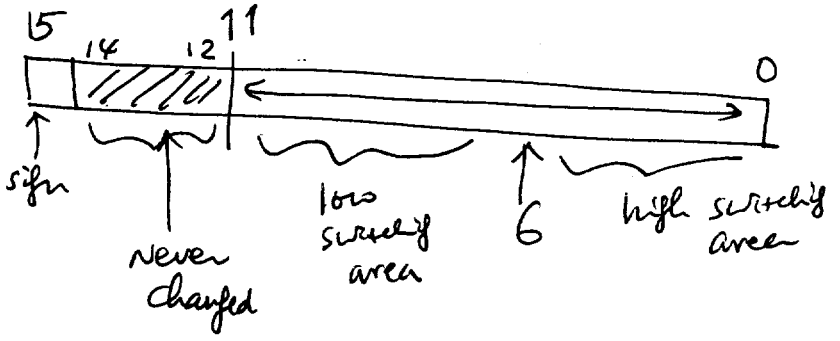




2's complement representation.







- The biggest win of sign and magnitude system is
  - ① Dynamic range of the represented numbers ~~are~~ is small
  - ② highly negatively correlated between the consecutive numbers.

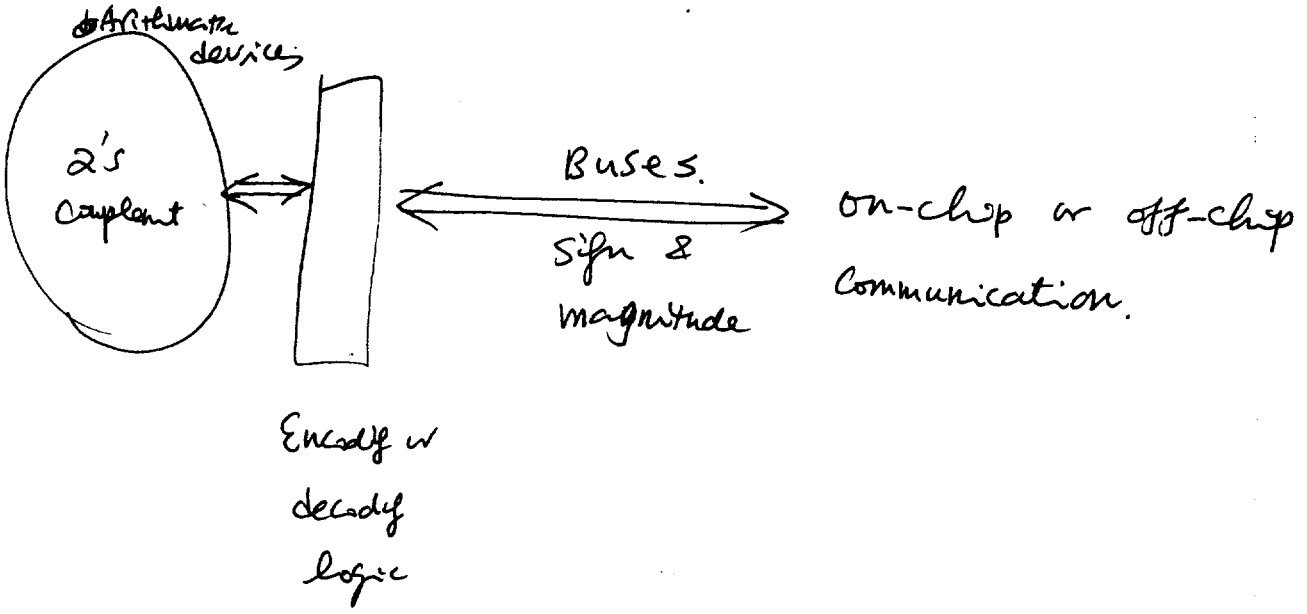
Conclusion: sign & magnitude representation ⇒ good for data transfer devices, e.g., buses

2's complement representation ⇒ good for arithmatic ckts.

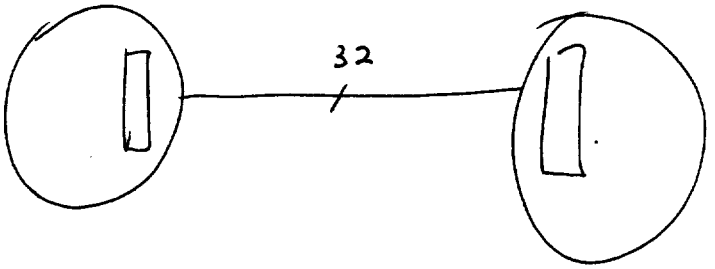
∴ simple algorithm ⇒ simple ckts ⇒ smaller ckt

⇒ lower power dissipation.

Best Solution:



Bus Encoding



- Try to use signed magnitude representation for communication
- Can be further improved by bus encoding.
- Motivation:

  power dissipation at I/O pins accounts for 50% of overall switching power

  ∵ pin has very high capacitance.

  ∴ Should try to reduce the switching activities of I/O pins. by encoding.

- Encoding methods:

  Active high encoding:

  high-level voltage : 1

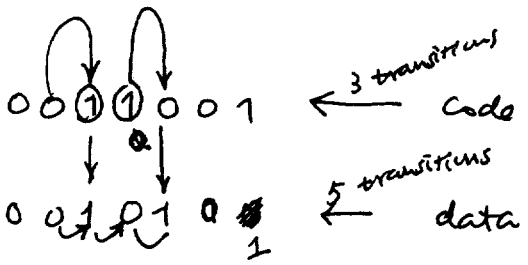
  low-level voltage : 0.

  transition-based encoding:

  Voltage change: 1

  no voltage change: 0.

Example:



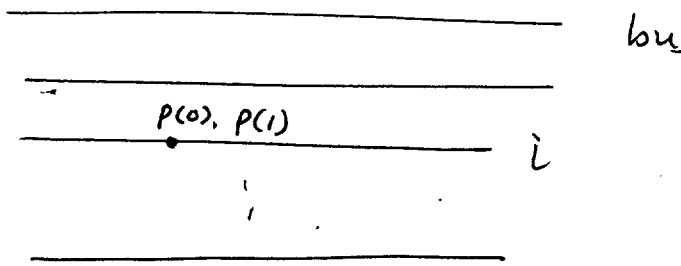
Code ↓ Data ↓

voltage change → 1

no voltage change → 0

Transition-based encoding

- Good only when there are many 0's transmitted.



Assume: $p(0) > p(1)$

No temporal correlation between consecutive values.

Switching activities:

Active-high encoding:

average # of transitions / clock cycle

$$p(0)p(1) + p(1)p(0) = 2p(1)(1-p(1))$$

if $p(1) = \frac{1}{4}$

$p(0) = \frac{3}{4}$

⇒ Switching activity is $3/8$.

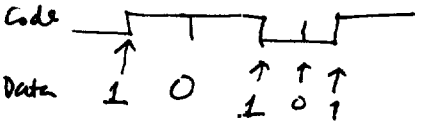
average

Transition-based encoding:

Transition required only when there is a logic one.

⇒ average switching activity is $p(1)$.

$1/4$ in this case

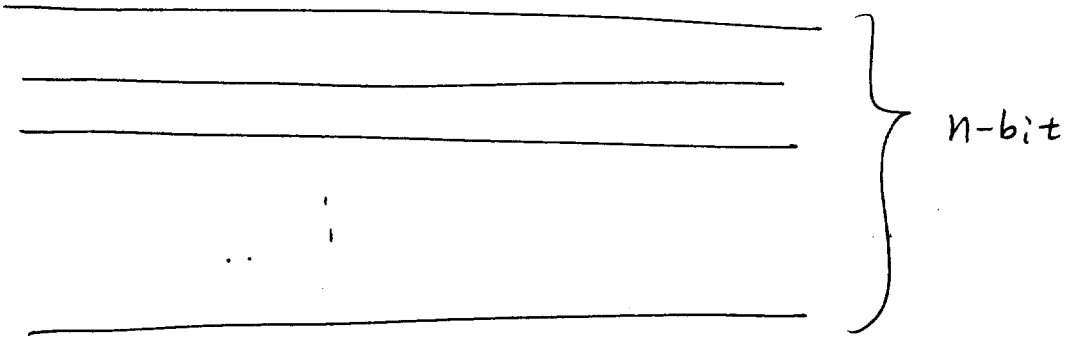


⇒ 33% less than the active-high encoding.

- Transition-based encoding is good when inputs are strongly not equiprobable.

  * If $p(0) > p(1)$ for a line, then the average switching activity of the line is smaller (than active high code).

  * Try to use limited-weight code to make sure that $p(0) > p(1)$ for each bit line.



K-limited-weight code: There are k 1's in the n bits.

↑ can represent $\binom{n}{k}$ data

Example: $k = 1$.

0 0 0 0 1 ← code word1
0 0 0 1 0 ← code word2
0 0 1 0 0
0 1 0 0 0 ⋮
1 0 0 0 0 ← code word5

5 code words

$\therefore P(0) = \frac{n-1}{n}$ $\quad P(1) = \frac{1}{n}$

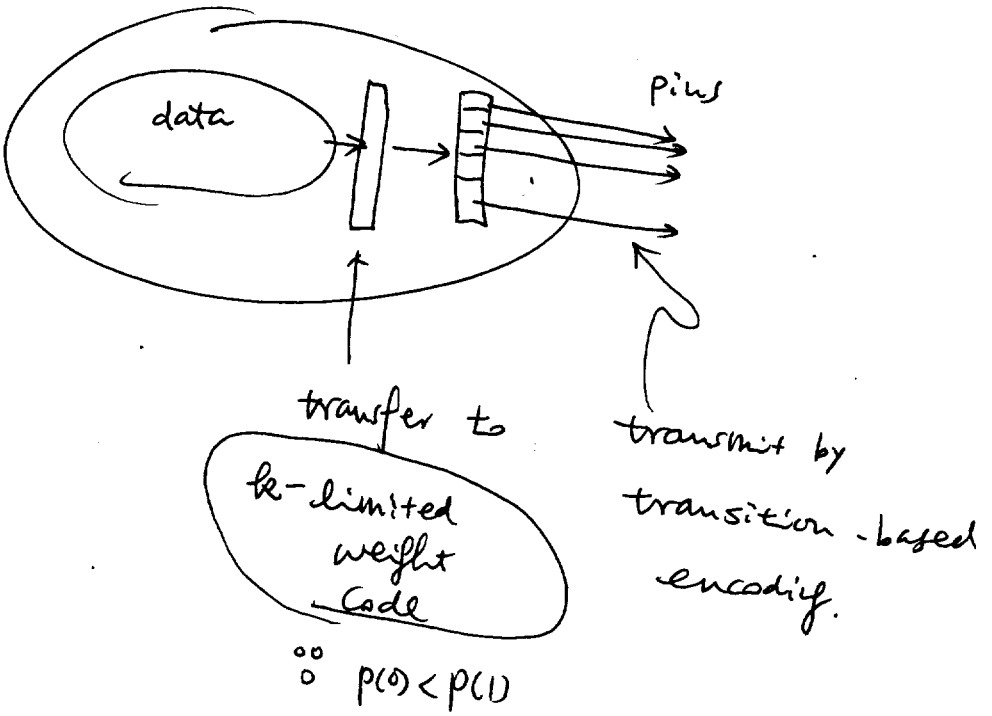
$k = n/2$

$P(0) = P(1) = \frac{1}{2}$

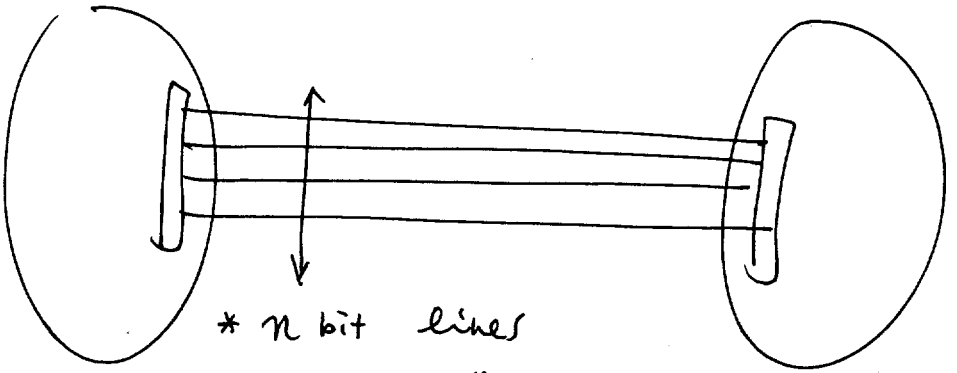
* for $1 \le k < n/2$, we are sure that $P(0) > P(1)$

$\therefore$ the transition-based encoding benefits.

Conclusion:



• Bus inverting encoding



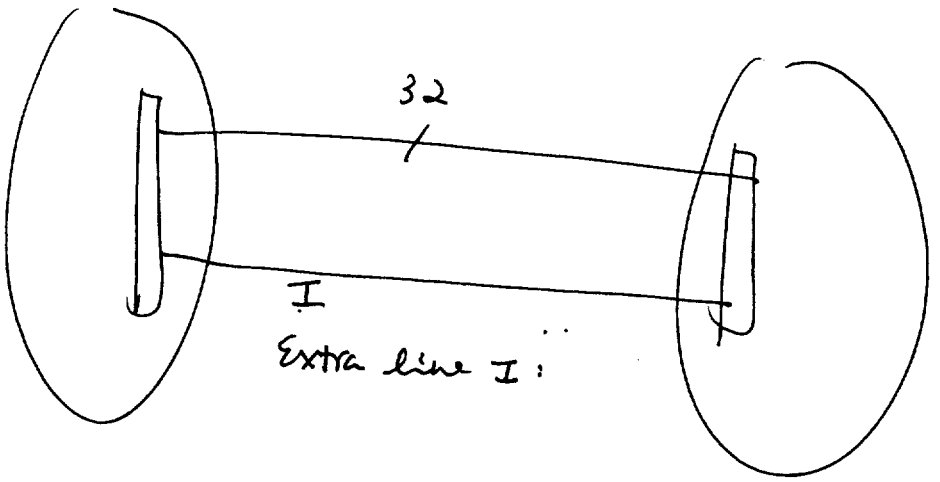
* n bit lines
* represent 2^n data

* $P(0) = P(1) = \frac{1}{2}$ for each line

$$\frac{\text{Avege \# of transitions}}{\text{clock cycle}} = n/2.$$

$$\frac{\text{Worst \# of transitions}}{\text{clock cycle}} = n.$$

* Try to reduce both avege (worst) # of transitions per cycle.



- Compare each pair of consecutive patterns.

① If Hamming distance between current & next patterns $\leq n/2$, $I=0$, send as what it is.

② If Hamming distance between ——————

$> n/2$, $I=1$, transmit the data.

pattern is converted

∴ This guarantee that the max # of transition per clock cycle is $n/2$.

* By statistical analysis, the avege # of transitions per clock cycle is lowered by 25% of the orginal.

- Hardware overhead for code generation must be invested.
- All above discussions are for data bus.

- How about address bus ?

  - Addresses generated by a running up are often consecutive.

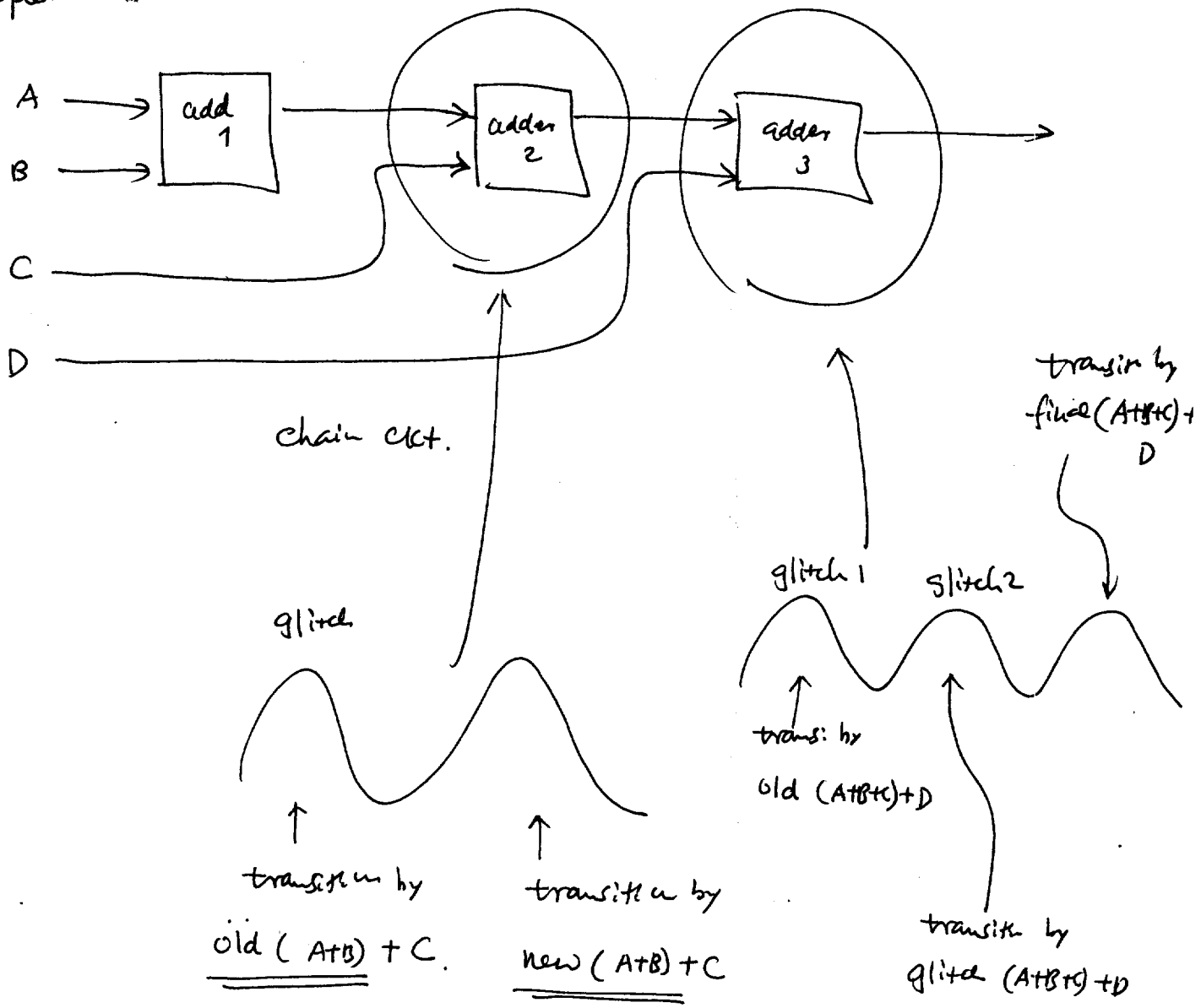
∴ Try to represent address by Gray Code.

(guarantee single-bit change for current & next address)

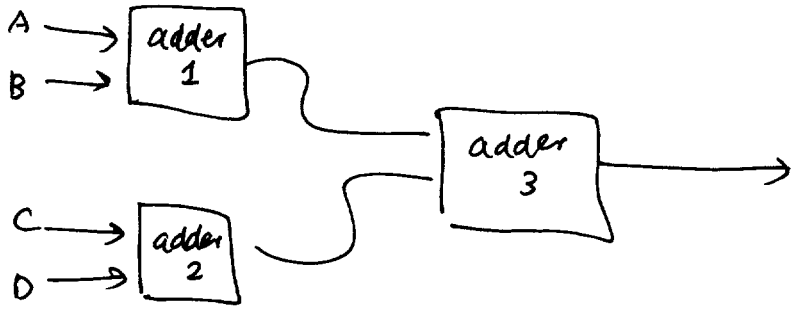
- Computer simulation demonstrate about 37% power reduction for several benchmark programs.

- Signal Synchronization
  - Try to reduce the # of glitches in RT-level by balancing signal paths or reducing logic depth.

Exeple:

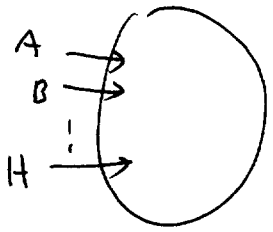


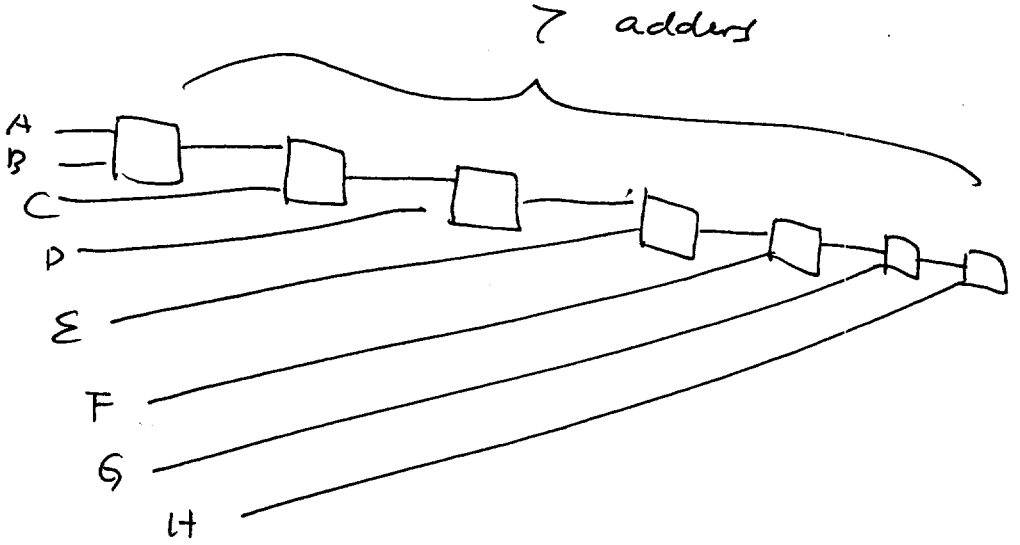
- Low-power design

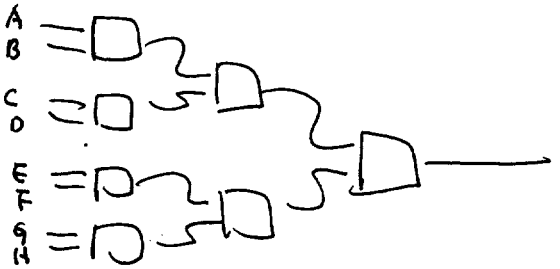


* paths are balanced, so the # of glitches is minimized.

- 8 inputs







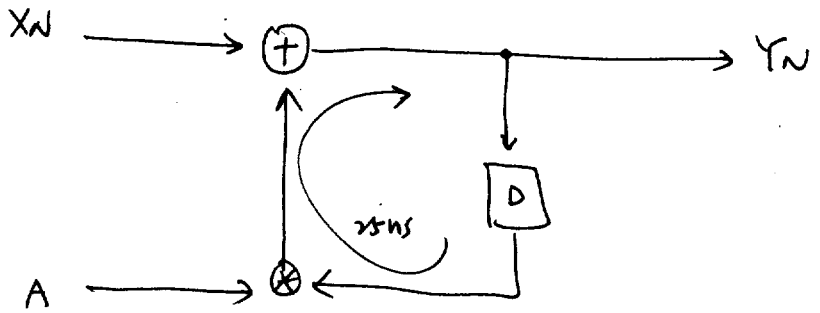
Decreasy the logic depth increases the # of registers required by the design ?? → add extra power consumption!!

- Resource sharing: readily assignment.

- **Algorithmic-Level optimization**

  - Try to modify the computational structure of the algorithm while preserving I/O behavior
  - Objective: optimize the power dissipation with a specific throughput.

- Speed-up transformations



$$Y_N = A \cdot Y_{N-1} + X_N$$

infinite impulse response

first-order IIR filter

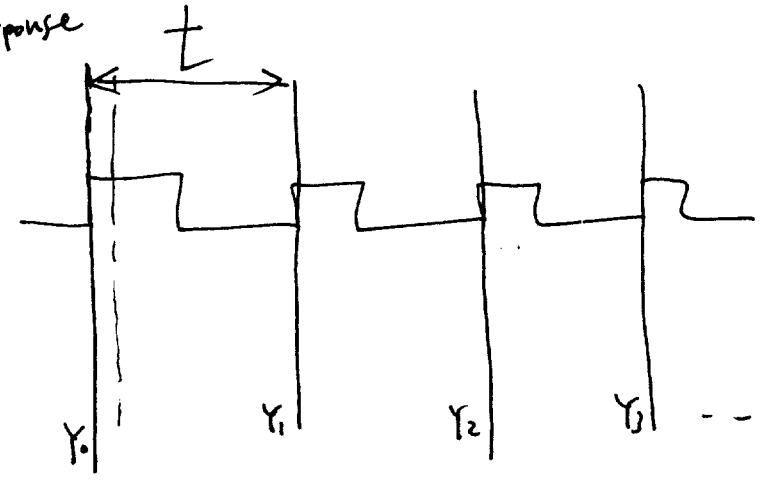
$C_{eff} = 1$

$V_{dd} = 5V.$

Throughput = 1

Critical path = 2

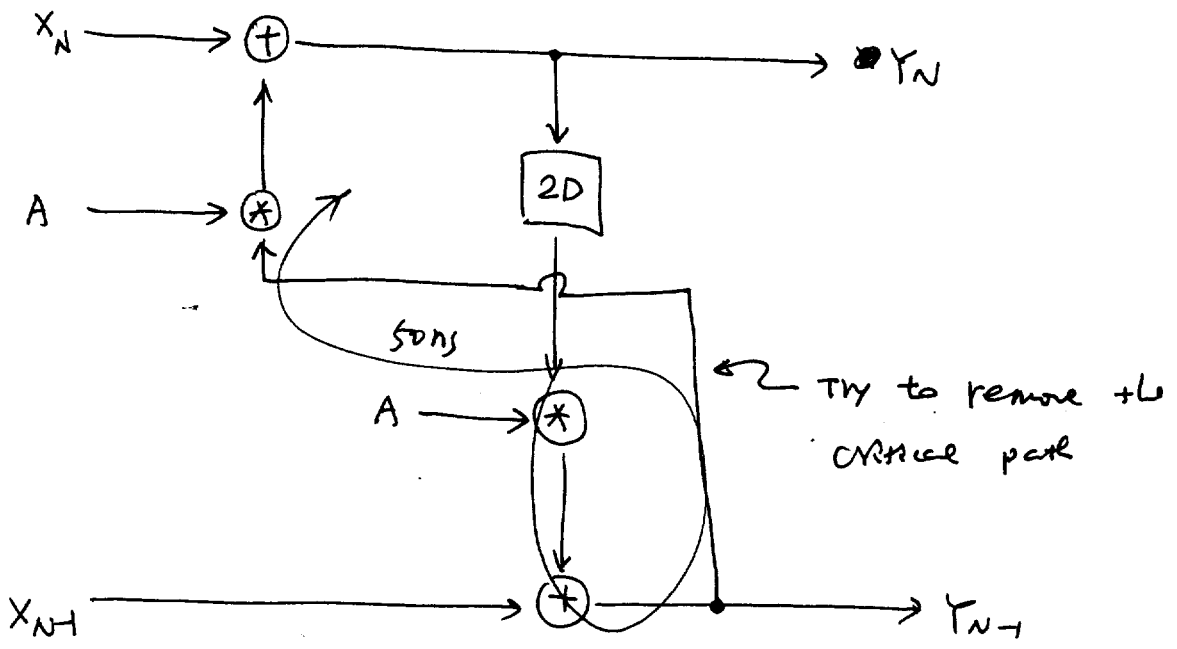
$CV^2 \rightarrow$ power = 25



e.g. 25ns

- Can't be further optimized by pipelining, re-timing...

- Try to unfold the loop



$$Y_{N-1} = Y_{N-2} \cdot A + X_{N-1}$$

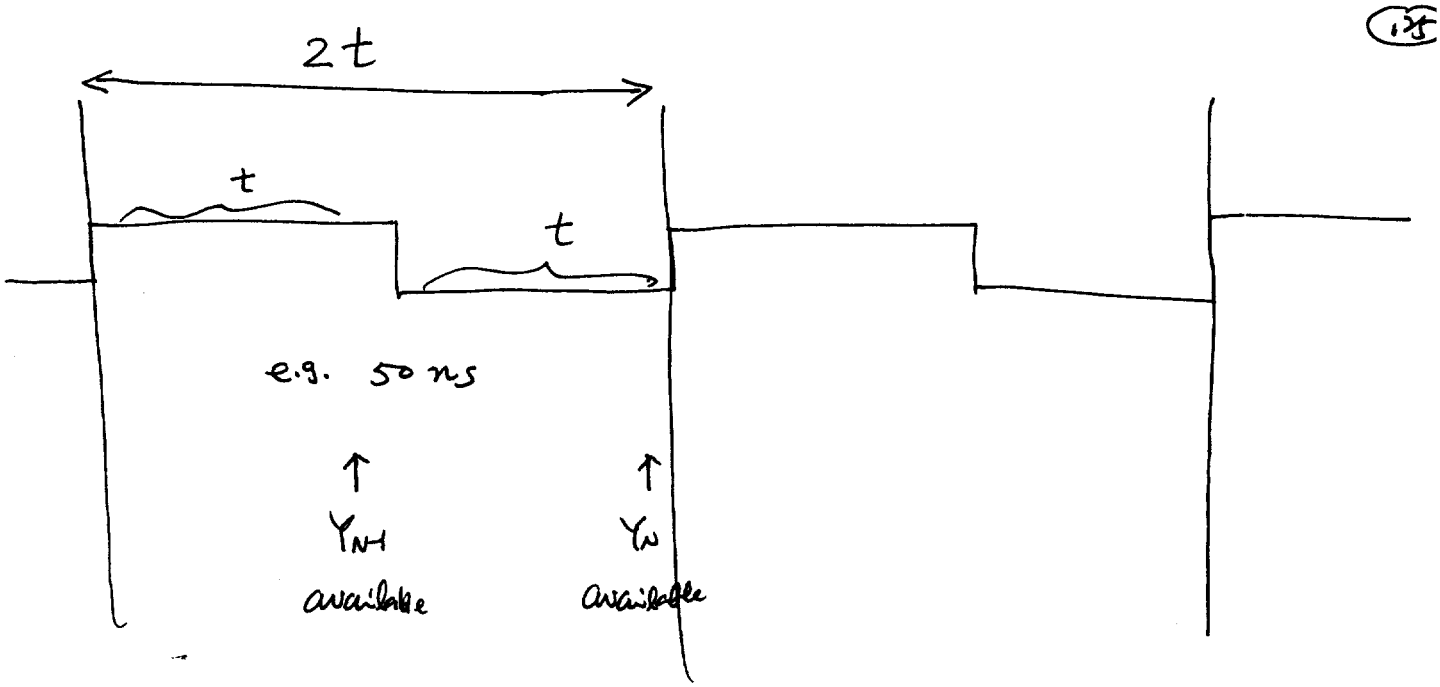
$$Y_N = Y_{N-1} \cdot A + X_N$$

$$= \left(Y_{N-2} \cdot A + X_{N-1}\right) A + X_N$$

for next update

$$= Y_{N-2} A^2 + X_{N-1} A + X_N$$

∴ function is the same.



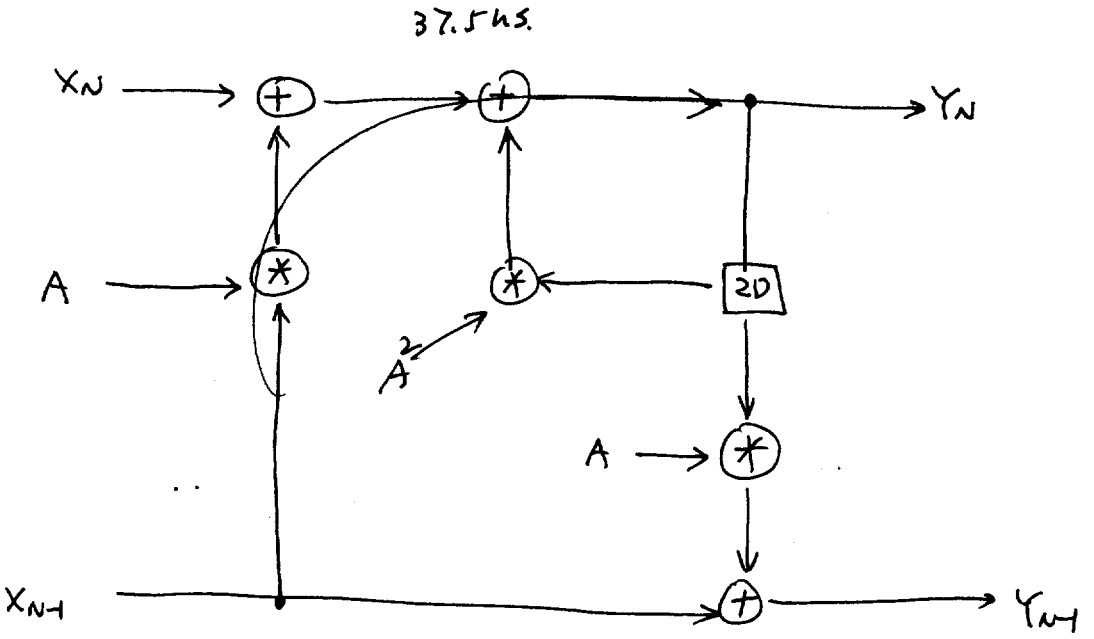
$C_{eff} = 1$ (C ↑ but $\mathcal{E}(sw) \cdot f$ ↓)

Vdd = 5V

throughput = 1

critical path = ●●4, But effective critical path = 2.

power = 25



$$Y_{N-1} = Y_{N-2} \cdot A + X_{N-1}$$

$$Y_N = (X_{N-1} \cdot A + X_N) + A^2 \cdot Y_{N-2}$$

$$= A^2 Y_{N-2} + A X_{N-1} + X_N$$

* Critical path has been reduced from 4 to 3.

→ Supply voltage can be reduced to keep the same throughput

$C_{eff} = 1.5$ (added more multipliers & adders)
Also, more devices are switching simultaneously.

$V_{dd} = 3.7V$

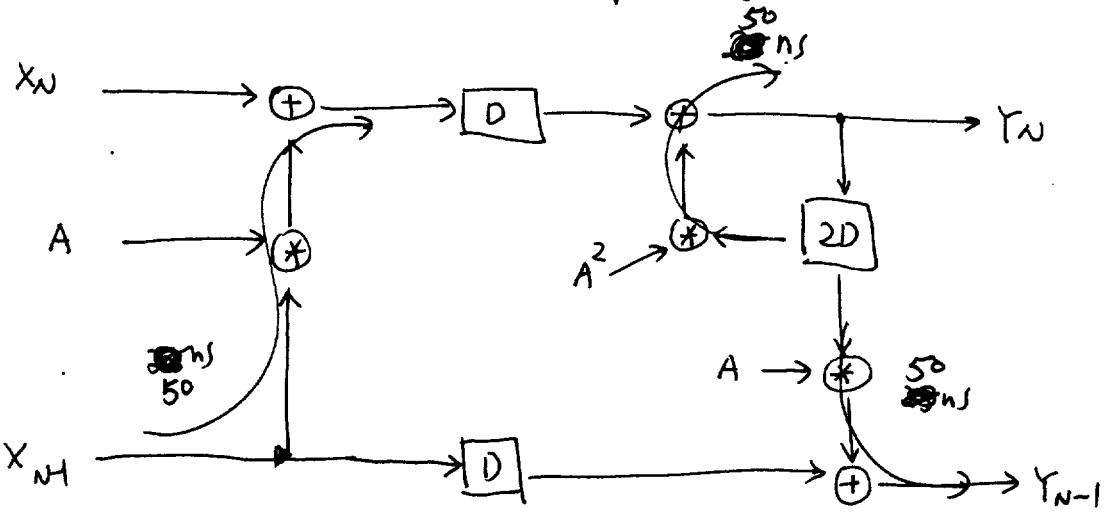
Throughput = 1

Critical path = 3

Power = 20

∴ 20% power saving over the original design.

* Further improvement by pipelining



$$\begin{array}{r} 4^2 \\ 34 \\ \hline 16^8 \\ 126 \\ \hline 1428 \end{array}$$

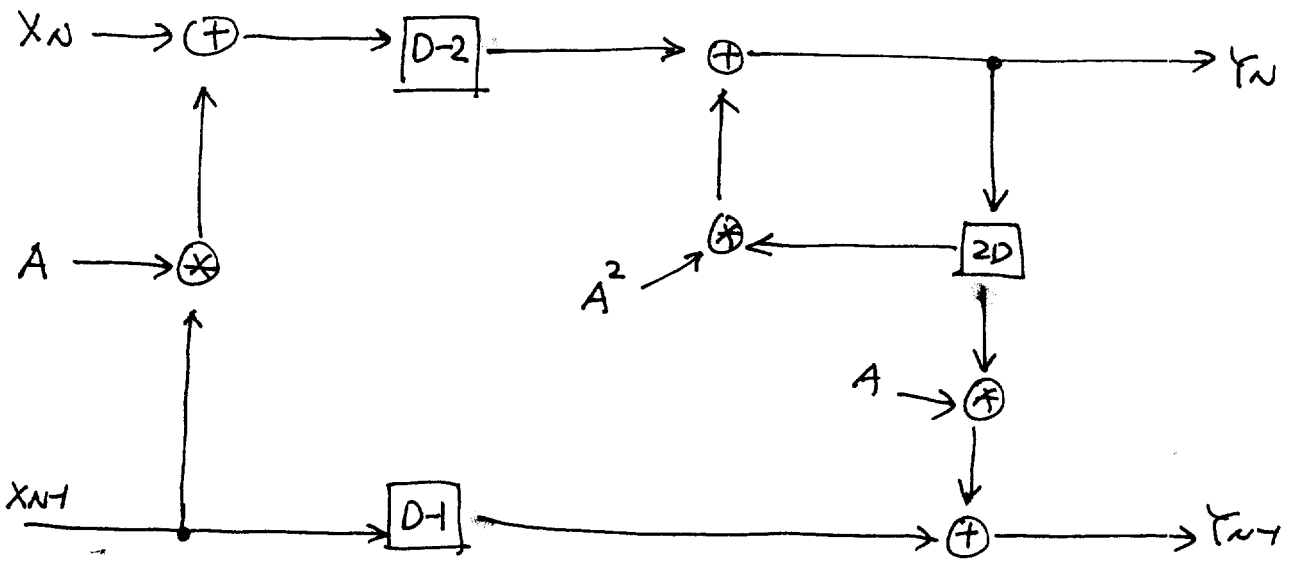
$$Y_0 = X_0 + A \cdot Y_{-1}$$

$$Y_1 = X_1 + A \cdot Y_0 = X_1 + A\,(X_0 + A \cdot Y_{-1})$$

$$Y_2 = X_2 + A \cdot Y_1 = X_1 + A \cdot X_0 + A^2 \cdot Y_{-1}$$

$$Y_3 = X_3 + AY_2 = X_3 + A\,(X_2 + AY_1)$$

$$= X_3 + AX_2 + A^2 Y_1$$



X_{N-1}	X_N	D-1	D-2	2D	Y_{N-1}	Y_N
X_0	X_1	×	×	×	×	×

(by reset)

X_0	$A \cdot X_0 + X_1$	Y_{-1}	Y_0	Y_1

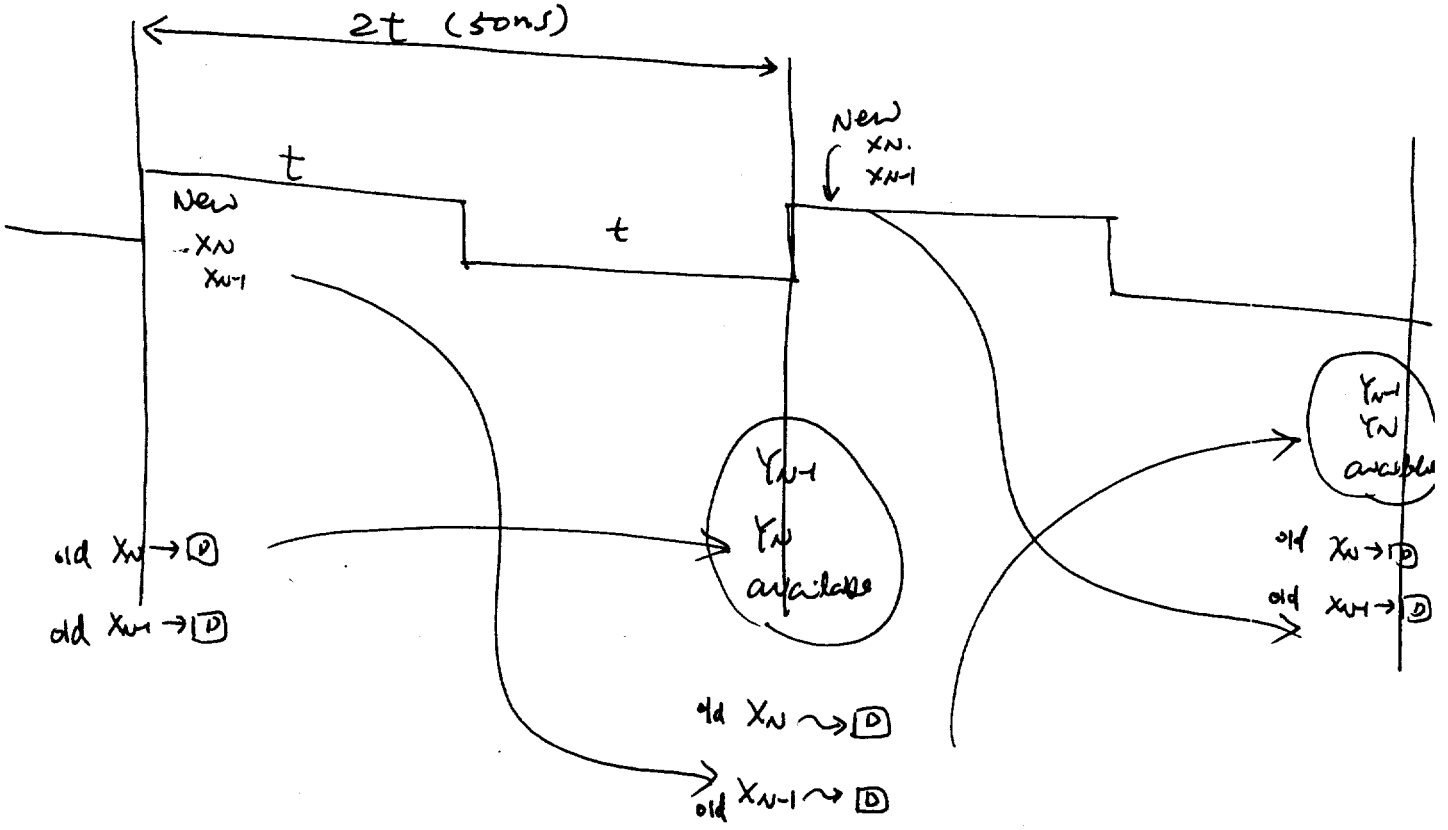
X_2 X_3

X_2	$AX_2 + X_3$	Y_1	$Y_2 = AY_1 + X_2$

$$Y_3 = X_3 + AX_2 + A^2 Y_1$$

· To Get the same throughput, we still use 2t Clock cycle (~~50~~ 50 ns)

∵ Y_N & Y_{N-1} can be generated simultaneously.



∴

$C_{eff} = 1.5$

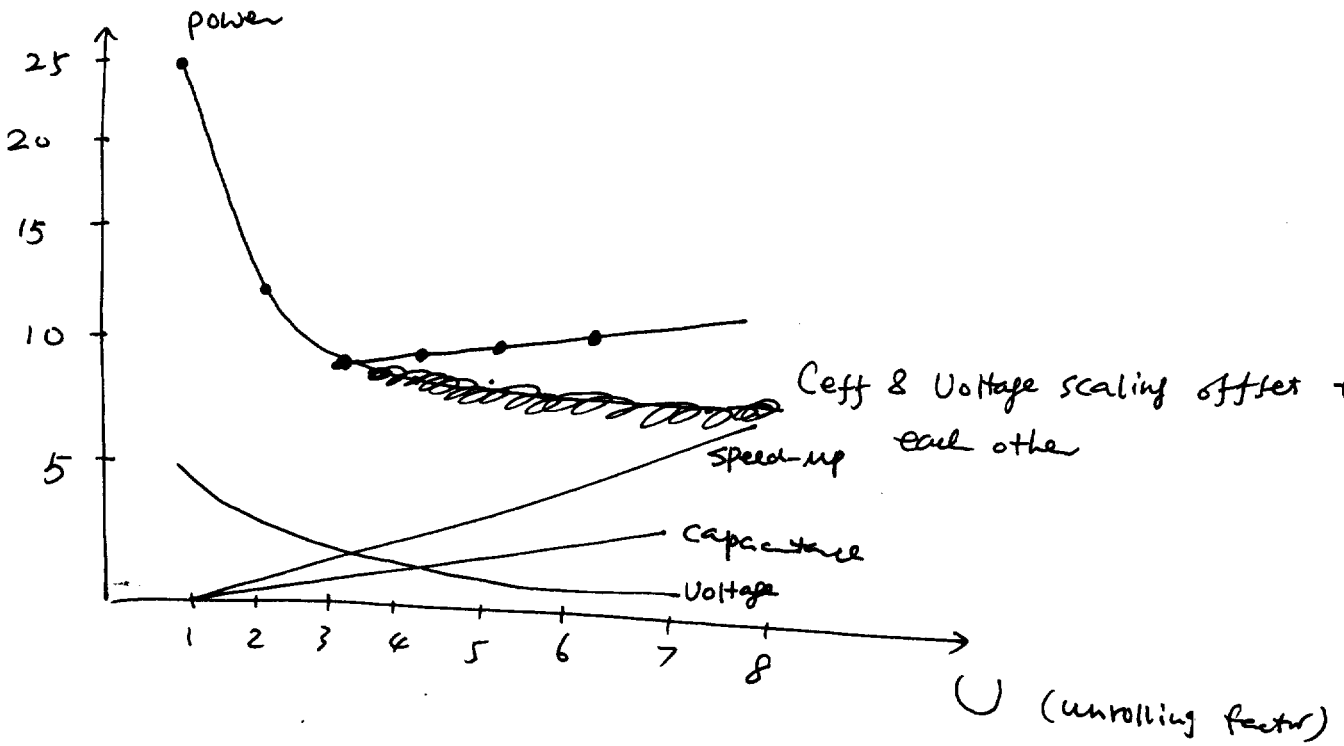
$V_{dd} = 2.9V$ ← delay is changed from 50 ns → 29 ns

Throughput = 1

Critical path = 2

CV^2 power = 12.5

50% power reduction



power reduction for unrolling factor 2.3

power increase from factor = 4