Campus 无线装订本
A5 40页
WCN-CNB3430
6 937748 309949
MADE IN CHINA

国誉商业(上海)有限公司
http://www.kokuyo.cn/st/
上海市奉贤区人杰路128号
TEL：400-820-0798　FAX：021-3255-8508
产地：上海市　QB/T1438-2007合格
A5: 210×148mm

● 采用日本进口纸张，
纸质细滑，牢固不掉页。

Campus | A5 | 7mm×24行 | 40页

System on CHip Techniques

Campus®

A5 点线本 | 7mm×24行 | 40页

KOKUYO

No.

Date

Adder { Half Adder: No Cin
Full Adder: With Cin

Binary Adder

Cin	A	B	S	C
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$S = Cin' \cdot A' \cdot B + Cin' \cdot A \cdot B' + Cin \cdot A' \cdot B' + Cin \cdot A \cdot B$

$= Cin'(A'B + AB') + Cin(A'B + AB)$

$= Cin'(A'B + AB')' + Cin(A'B + AB)'$ $= A \oplus B \oplus Cin$

$C = \cancel{Cin \cdot AB} + AB + CinA + CinB = Cin(A+B) + AB$

[If do not use logic minimisation]

$C = Cin'AB + CinA'B + CinAB' + CinAB$

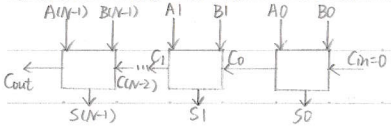
$= Cin(A \oplus B) + AB$

$S = A \oplus B \oplus Cin$

2X XOR + 2X OR + 2X AND

2X XOR + 2X AND + 1X OR

Multi-bit Adder (cascade N number of full adders)



Linear increase of delay with number of bits

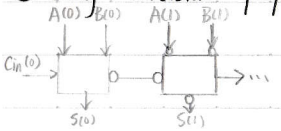
$Td(worst) = (N-1)Tcarry + Tsum$

Carry chain optimisation

① Express S in terms of the signal C: $S = ABCin + (A+B+Cin) \cdot C'$

Thus for each adder the C signal is coming earlier than the sum signal.

② Using inversion property



Reduce one CMOS invert per 2 adders / Significant decrease time delay in carry chain

将内部(adder)输出端的CMOS反向器去掉，第1个adders输出的C位反向，利用反演属性，将输入端 P→N，N→P，使第二个adder输出原C位值(第2个adder不去掉CMOS反向器)。

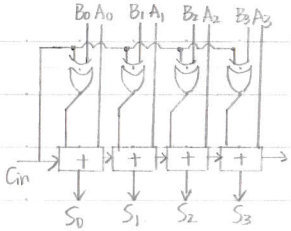
Subtractor

2's complement: $5_{10} = 1001_2$ $-5_{10} = 00001001_2 \rightarrow 11110110_2 \rightarrow 11110111_2$

A, B — XOR — z

A	B	z
0	0	0
0	1	1
1	0	1
1	1	0

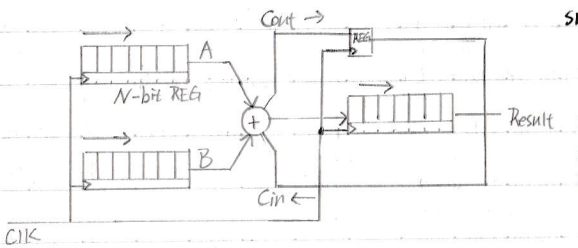
↑ carry bit
{ 0: adder
1: subtractor



No.

Date

Bit Serial Adder



carry (t) = A(t)B(t) + C(t-1)[A(t)+B(t)]
sum (t) = carry (t)'[A(t)+B(t)+C(t)] + A(t)B(t)

① Equal sum and carry delay
② N number of cycles required
③ Good for technology in the range 2-5 μm
④ Nibble (4bit) and byte (8bit) adders are frequently
⑤ Carry-save-addition

Carry-lookahead Adder (CLA) try to predict C_k earlier than $T_c * k$

Look at the 0th stage carry and predict all the carry corresponding to different stages

[Carry propagation]

A	B	Cin	C	
0	0	X	0	kill
0	1	C	C	propagate
1	0	C	C	propagate
1	1	X	1	generate

$\begin{cases} P = A+B \\ g = AB \end{cases}$

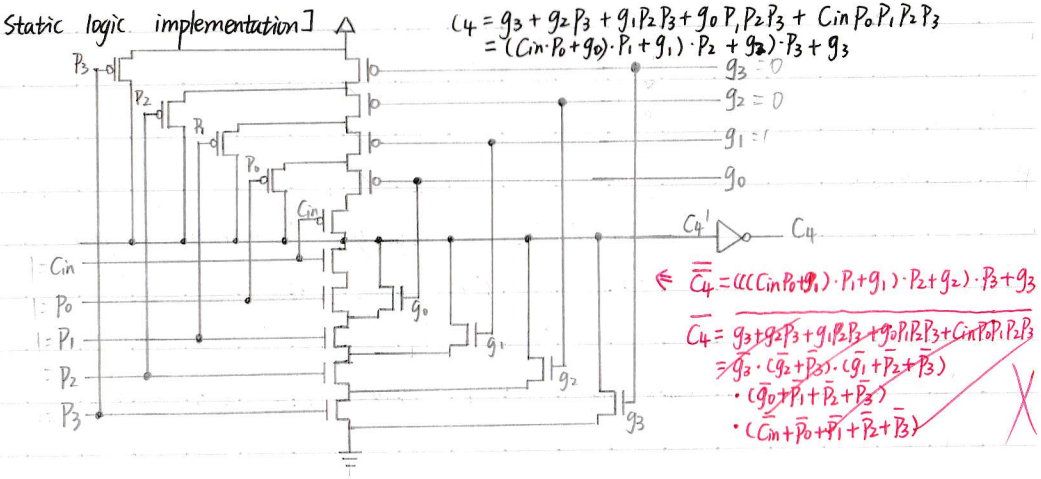
$C_4 = 1$?

$C_4 = OR \begin{cases} g_3 \\ g_2 \cdot P_3 \\ g_1 \cdot P_2 \cdot P_3 \\ g_0 \cdot P_1 \cdot P_2 \cdot P_3 \\ C_{in} \cdot P_0 \cdot P_1 \cdot P_2 \cdot P_3 \end{cases}$

generate at bit position 3
… 2, propagated 3
… 1, … 2,3
… 0, … 1,2,3
… input, … 0,1,2,3

[Static logic implementation]

$C_4 = g_3 + g_2P_3 + g_1P_2P_3 + g_0P_1P_2P_3 + C_{in}P_0P_1P_2P_3$
$= (C_{in}\cdot P_0 + g_0)\cdot P_1 + g_1)\cdot P_2 + g_2)\cdot P_3 + g_3$



$\overline{C_4} = (((C_{in}P_0+g_0)\cdot P_1+g_1)\cdot P_2+g_2)\cdot P_3+g_3$

$\overline{C_4} = \overline{g_3+g_2P_3+g_1P_2P_3+g_0P_1P_2P_3+C_{in}P_0P_1P_2P_3}$
$= \bar{g}_3\cdot(\bar{g}_2+\bar{P}_3)\cdot(\bar{g}_1+\bar{P}_2+\bar{P}_3)$
$\cdot(\bar{g}_0+\bar{P}_1+\bar{P}_2+\bar{P}_3)$
$\cdot(\bar{C}_{in}+\bar{P}_0+\bar{P}_1+\bar{P}_2+\bar{P}_3)$ ✗

{ P-MOS passes '1' very efficiently
{ n-MOS passes '0' very efficiently

No.

Date

f(A,B)

$C_4 = g_3 + g_2P_3 + g_1P_2P_3 + g_0P_1P_2P_3 + C_{in}P_0P_1P_2P_3$
$= (C_{in}P_0 + g_0)\cdot P_1 + g_1)\cdot P_2 + g_2)\cdot P_3 + g_3$

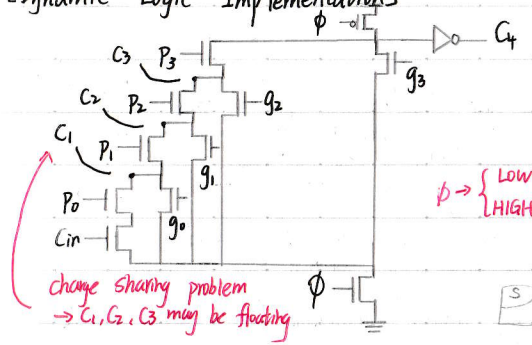
$f(\bar{A},\bar{B})$

$C_4 = g_3 + g_2\cdot P_3 + g_2\cdot P_2P_3 + g_1P_1P_2P_3 + g_0P_0P_1P_2P_3 + C_{in}P_0P_1P_2P_3$
$= P_3(g_3 + P_2(g_2 + P_1(g_1 + P_0(g_0 + C_{in}))$

$\bar{C_4} = \bar{P}_3 + \bar{g}_3\cdot(\bar{P}_2 + \bar{g}_2\cdot(\bar{P}_1 + \bar{g}_1\cdot(\bar{P}_0 + \bar{g}_0\cdot\bar{C}_{in}))$

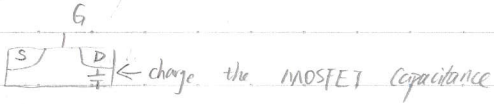
$\begin{cases} C_4 = (C_{in}\cdot P_0 + g_0)\cdot P_1 + g_1)\cdot P_2 + g_2)\cdot P_3 + g_3 \\ \bar{C_4} = (C_{in}\cdot\bar{g}_0 + \bar{P}_0)\cdot\bar{g}_1 + \bar{P}_1)\cdot\bar{g}_2 + \bar{P}_2)\cdot\bar{g}_3 + \bar{P}_3 \end{cases}$

[Dynamic Logic Implementation]



$C_4 = g_3 + P_3(g_2 + P_2(g_1 + P_1(g_0 + P_0\cdot C_{in})))$

$\phi \rightarrow \begin{cases} \text{LOW: Pre-charge} \\ \text{HIGH: Get } C_4 \end{cases}$



[CLA - Summary]

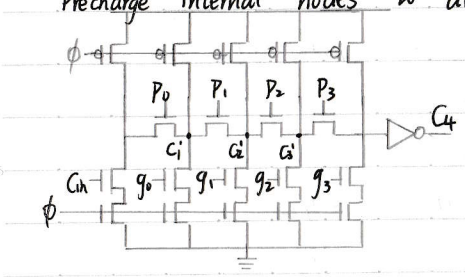
CLA compared to ripple-carry adder: { Faster / Larger Area

Limitation: cannot go beyond 4 bits of look-ahead
↳ large P, g fan-out slows down carry generation

Way out: Manchester carry chains
↳ Tries to reuse logic by pre-charging each carry position

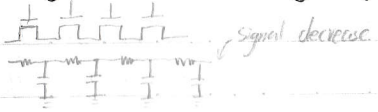
Manchester Carry Chain

Precharge internal nodes to avoid charge-sharing problem

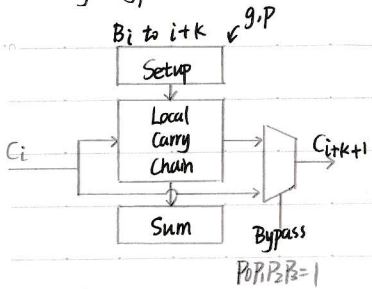


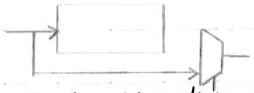
Compared to CLA:
① Smaller area
② Cin close to the output

Carry chain can be any length



Carry-bypass adder



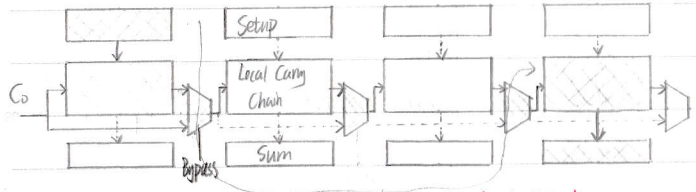


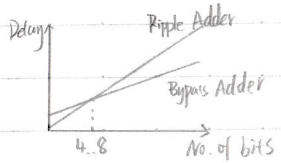
[Pass mode]
Local carry vector computes intermediate carries (possibly incorrect)
At the same time, mux selection set to pass
When input arrives, intermediate carries might be recomputed
Meanwhile, input carry is sent to Cout

[If not pass mode]
Local carry vector computes intermediate carries (bits 10, 11 correct)
At the same time, mux selection set to local
Meanwhile, output carry is sent to Cout correctly
When input carry arrives, intermediate carries C8 and C9 will be recomputed correctly

[Timing]

每个Cout同时产生





$Delay = t_{setup} + max\{t_{select}, 4\times t_{FA}\} + 3\times t_{mux\_pass} + 4\times t_{FA} + t_{sum}$

Faster than ripple adder but still linear

[EXAM]

$A_{xor} = 1\mu m^2$
$A_{and} = 0.5\mu m^2$
$A_{FA} = 2\mu m^2$
$A_{2:1\,muti} = 0.2\mu m^2$

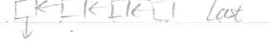
$A_{setup} + A_{LCC} + A_{muti} = 14.2\mu m^2$
$A_{setup} = 4(A_{and} + A_{xor}) = 4\times 1.5 = 6\mu m^2$
$A_{LCC} = 4\times 2 = 8\mu m^2$
$A_{2:1\,muti} = 0.2\mu m^2$

$A_{all} = 3\times 14.2\,\mu m + 14$

$t_{and_2} = 1$
$t_{xor} = 2$
$t_c/bit = 4$
$t_{sum} = 2$
$t_{2:1\,mux} = 1$

$t_{setup} = 2$
$t_{FA} = 4\times 4 = 16$
$t_{select} = P_0P_1P_2P_3 = 2$
$t_{mux} = 3\times 1 = 3$

$2+16+3+(6+2)$
$3t_c + t_{sum}$



Carry-select adder

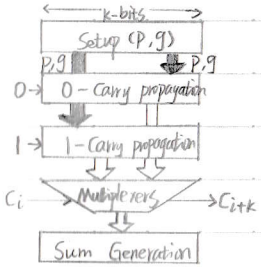
$t_{and} = 1$
$t_{xor} = 2$
$t_{FA}/bit = 4$
$t_{2:1\,mux} = 1$

$t_{setup} + 4t_{FA} + 4t_{mux} = 2+16+4 = 22$

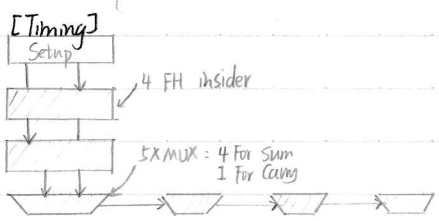
$A_{FA} = 2\mu m^2$
$A_{2:1\,mux} = 1\mu m^2$

4个3会 Sum, 1个3会 Carry

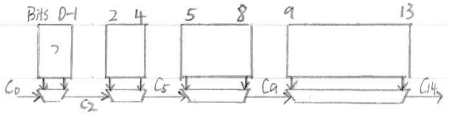
$8A_{FA} + 5A_{mux} = 21$



[Timing]



Square root carry select adder



N-bit adder
P stages (delay directly depends on P)

$N = M + (M+1) + (M+2) + \cdots + (M+P-1)$
$= MP + \frac{P(P-1)}{2}$
$= \frac{P^2}{2} + P(M - \frac{1}{2})$

$P \approx \sqrt{2N}$

Multiplier

13x5 :
```
  1101   ← Multiplicand 被乘數
x0101    ← Multiplier   乘數
------
  1101
 0000
1101
0000
-------
0|000001  ← Product
```

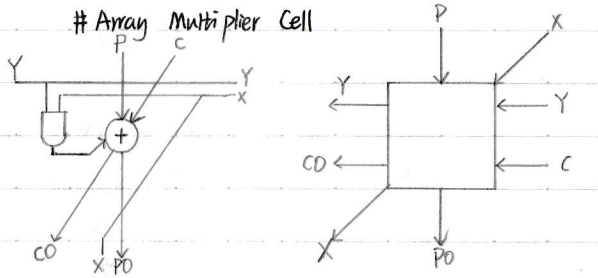
If length of the multiplicand is N-bit and length of the multiplier is R-bit, then length of the product P = N+R

Unsigned Digital Multiplication

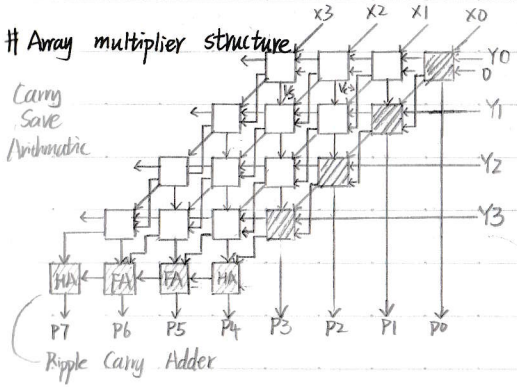
Parallel Array Multiplier 平行陣列

```
                          x3   x2   x1   x0
                          Y3   Y2   Y1   Y0
                        ------------------
                        x3Y0 x2Y0 x1Y0 x0Y0
                   x3Y1 x2Y1 x1Y1 x0Y1
              x3Y2 x2Y2 x1Y2 x0Y2
         x3Y3 x2Y3 x1Y3 Y0Y3
        ------------------------
P7   P6   P5   P4   P3   P2   P1   P0
```

Array Multiplier Cell



Array multiplier structure



In each stage, LSB drops out to form part of partial product. The accumulating product is shifted right one place by wiring between each stage.

Total $6 \cdot T_{FA}$ T_{FA} for each line

Signed Binary Multiplication

```
  1010   -6  ← Signed Multiplicand
x 0101   5
---------
 1111 1010
0000 000
 1110 10
 00000
---------
11100010  ← 2's complement of 30
```

Sign extension →

```
  0110   6   Signed
x 1011  -5  ← Multiplier
---------
 00000110
 0000110
 000000
 110110
---------
11100010
```

No.

Date

Booth's algorithm

For i^{th} bit of the multiplier

r_i	r_{i-1}	$r_{i-1}-r_i$	Action
0	0	0	Add zero (do nothing)
1	0	-1	Subtract multiplicand
0	1	1	Add multiplicand
1	1	0	Add zero (do nothing)

```
  0 1 1 0  ← 6
X 1 0 1 1  ← -5
-----------
1 1 1 1 1 0 1 0  ← 0-1=-1  2's complement of 6
0 0 0 0 0 0 0 0  ← 1-1=0   Add zero
0 0 0 1 1 0      ← 1-0=1   Add 6
1 1 0 1 0        ← 0-1=-1  2's complement of 6
X 1 1 1 0 0 0 1 0  ← 2's complement of 30
```

Not necessary to know when we come to the MSB and switch to a subtraction

Modified Booth Algorithm

A faster multiplier can be realised by looking at two bit

For i^{th} and $(i+1)^{th}$ bit of multiplier, the algorithm is:

	r_{i+1}, r_i	r_{i-1}	$\{r_{i+1}, r_i\} + r_{i-1}$	Action
(0)	0 0	0	0	Do nothing
(1)	0 1	0	1	Add 1^* Multiplicand
(-2)	1 0	0	-2	Subtract 2^* Multiplicand
(-1)	1 1	0	-1	Subtract 1^* Multiplicand
(0)	0 0	1	1	Add 1^* Multiplicand
(1)	0 1	1	2	Add 2^* Multiplicand
(-2)	1 0	1	-1	Subtract 1^* Multiplicand
(-1)	1 1	1	0	Do nothing

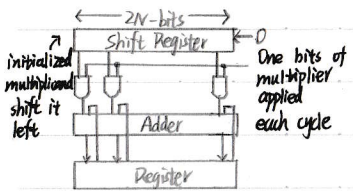
```
  0 1 1 0
X ( 1 0 0 1 0 )
---------------
1 1 1 1 1 0 1 0  ← 2's complement of 6
  ✗ ✗
1 1 1 0 1 0      ← Subtract 6   (进两位)
---------------
1 1 1 0 0 0 1 0

1 1 0 → -1  Subtract 1*
1 0 1 → -1  Subtract 1*
```

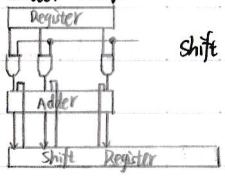
Grouping multiplier bits into pairs ⇒ Reduces number of partial products to half
Use of higher radix reduces number of intermediate addition
- Radix-8 should implement *3, *-3, *4, *-4
- Complicated, trade-off has to be made

Depends on architecture

Sequential Multiplier



Better Design



Shift result register to right
- N-bit AND gate
- N-bit Adder

- 2N-bit adder
- 2N-bit register

No.

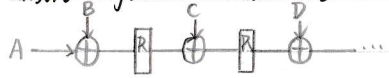
Date

Shift and Rotate Operation

Difference between
- Shifter: shifts any bits
- Shift register: shifts only one bit

Pipelined multiplier

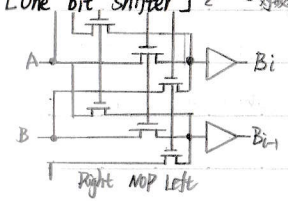
Insert registers between rows



LOSE Speed ⇒ Get lower power consumption

T0: A0B0C0D0 A0+B0 WAIT WAIT
T1: A1B1C1D1 A1+B1 A0B0+C0 WAIT
T2: A2B2C2D2 A2+B2 A1+B1+C1 A0B0+C0+D0

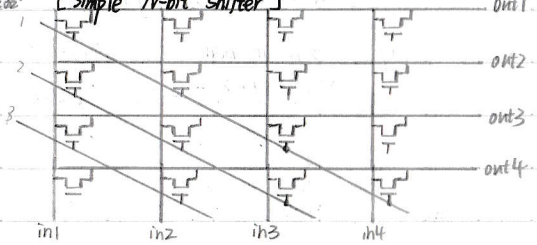
[One bit shifter]



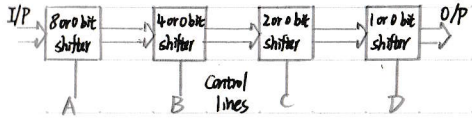
Logarithmic shifter [由 one bit 组成]

[Simple N-bit shifter]

Barrel shifter



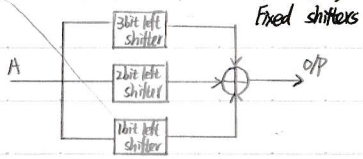
Programmable shifter 16-bit



Multiplication using shifter

$A \times 14 = A\,(1110) = A(2^3+2^2+2^1)$

Fixed shifters



Summary

Trade-offs between area and delay
- Barrel shifter: fastest O(1), N^2 transistors
- Logarithmic shifter: O(log N), N log N transistors
- One bit shifter: O(N), N transistors

Barrel shifter: wire dominated circuit

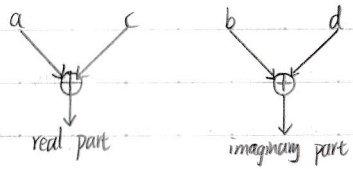
No.

Date

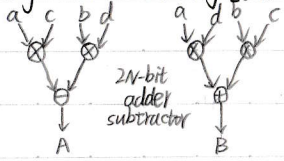
Complex numbers and addition

$x = a + b \cdot j$ { a: real part, b: imaginary part

$(a+j\cdot b) + (c+j\cdot d) = (a+c) + j(c+d)$



$xy = (ac - bd) + j(ad + bc) = A + j\cdot B$



1 N-bit real multiplier { N(N-2) full adders, N half adders, N^2 AND gates

Total: 4[N(N-2) full adders + (N/2) full adders + N^2 AND] + 2 (2N-bit) full adders

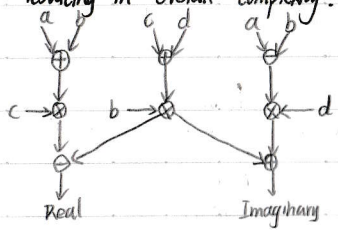
$= 2N(2N-1)$ full adders + N^2 AND gates

Reduce complexity

$\Rightarrow xy = (ac-bd) + j(ad+bc) = [(a+b)-b(d+c)] + j[d(a-b)+b(c+d)]$

resulting in overall complexity: 3 N-bit adders, 3 (N+1)-bit multipliers, 2(2(N+1))bit adders

$\approx (\frac{1}{2})[6N^2+17N+5]$ full adders + $3(N+1)^2$ AND gates.



$4[N(N-2) + \frac{N}{2}] + 4N^2$

Campus

No.

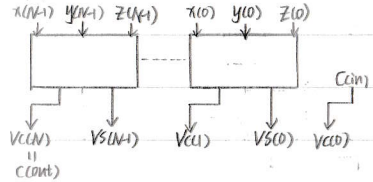
Date

Redundant Arithmetic

Carry-save addition (CSA)

Perform an addition of three vectors (or inputs) using an array of 1-bit adders without propagating the carries.

The sum r of 3 numbers x, y, z is represented as two vectors - carry vector (vc) and pseudo-sum vector (vs) $x+y+z = vc + vs = v$



x	0	1	1	1	0	1	0	0
y	0	0	1	1	1	0	1	1
z	1	0	1	0	1	0	1	0
Vs	1	1	1	0	0	1	0	1
Vc	0	1	1	1	0	1	0	0

The final result of CSA is in carry-save format. Just add Vs and Vc

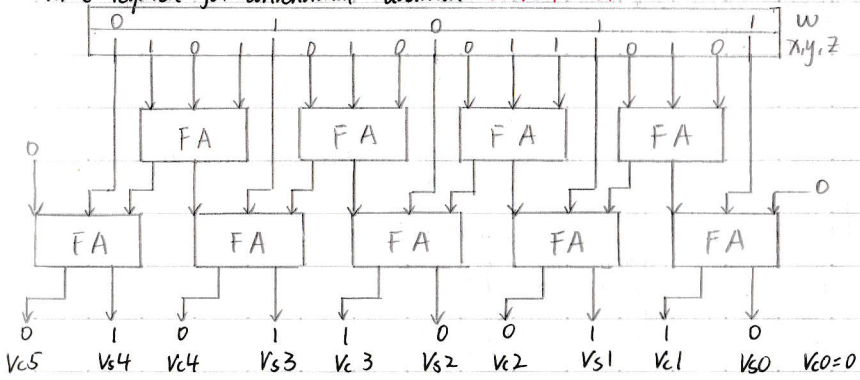
Multiple operand addition

x = 1010, y = 0111, z = 1000 $\Rightarrow$ Vs = 00101, Vc = 10100 $\Rightarrow$ 00101 + 10100 = 11001

Considering one full adder delay = T

Time required for carry-save addition: T + 5T = 6T (Need a 5-bit adder for converting)

Time required for conventional addition: 4T + 5T = 9T



4:2 adder

Conventional addition delay = 4T + 5T + 5T = 14T

Carry-save delay (include conversion) = 2T + 5T = 7T

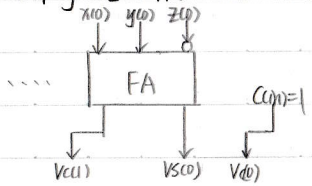
Actual carry-save delay is independent of the wordlength until you do conversion

KOKUYO

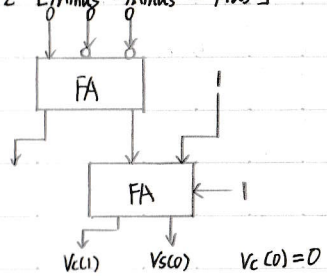
~~# Minus + Minus = Minus + Puls (MMP)~~

Subtraction

x + y − z [Plus −Plus −Minus operation]



x − y − z [Minus −Minus − Plus]



[Determine sign of CSA number]

In coventional 2's complement sign can be determined by checking the MSB
Not applicable for CSA number
Sign can be determined by adding Vs and Vc : speed advantage gone

Use generate - propagate formulation.
Take Vs(i) and Vc(i) as a bit pair and compute propagate p(i) starting from MSB position until you encounter 11 or 00 condition.

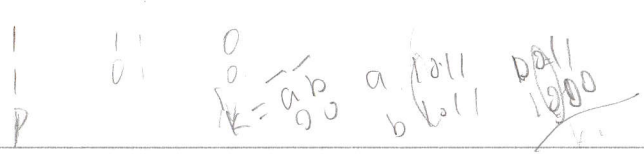
① If $p(N-1) \cdot p(N-2) \cdots p(k+1) = 1$ and $g(k) = 1$ ⇒ Positive Number
② If $p(N-1) \cdot p(N-2) \cdots p(k+1) = 1$ and $g(k) = 0$ and $p(k) = 0$ ⇒ Kill condition : Negative number
③ If $g(N-1) = 1$ ⇒ Negative Number

[The Logic Function]

0 as output will signify a +ve number and 1 will signify a -ve number
Considering detecting a -ve number, i.e., output logic function sign should be 1
Adding two 2's complement number will be -ve when

① $g3 = 1$: both numbers are -ve
② $p3k2 = 1$: bit 3 propagate, bit 2 kill (00 condition $k(i) = a(i)' \cdot b(i)'$
③ $p3p2k1 = 1$: bit 3 and bit 2 propagate, bit 1 kill
④ $p3p2p1k0 = 1$: bit 3, 2 and 1 propagate, bit 0 kill
⑤ $p3p2p1p0 = 1$: all bits propagate

Therefore $sgn = p3p2p1p0 + p3p2p1k0 + p3p2k1 + p3k2 + g3$
If sgn = 1, the number is



[EXAMPLE]

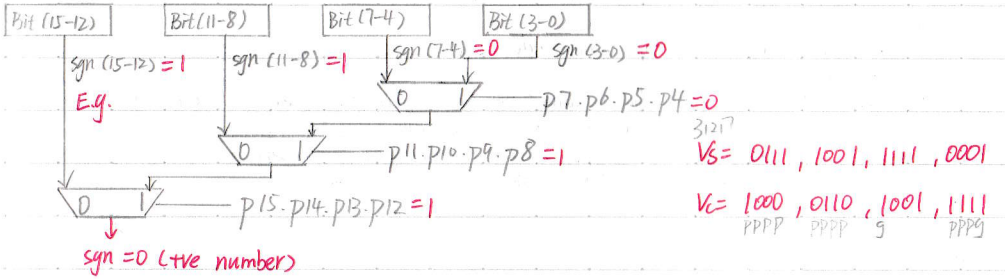
g3 = 1 : addition of two -ve numbers is -ve number
p3k2 = 1 : 1011 + 0011 = -5 + 3 = -ve
p3p2k1 = 1 : 1001 + 0101 = -7 + 5 = -ve
p3p2p1k0 = 1 : 1010 + 0100 = -6 + 4 = -ve
p3p2p1p0 = 1 : 1010 + 0101 = -6 + 5 = -ve

Therefore, in the worst case we need to ~~block~~ check all the bit pairs - a propagation chain exists from the LSB to MSB, slow operation particular for longer wordlength
↳ Way out : partition the wordlength and apply the criteria in parallel and finally combine
↳ In this case in the sgn expression use g(N-1) only for the final block
For the ~~final~~ intermediate blocks replace generate function by kill function

Consider block of 4 the sgn expression for 16-bit wordlength is

$sgn(15-12) = p15 \cdot p14 \cdot p13 \cdot p12 + p15 \cdot p14 \cdot p13 \cdot k12 + p15 \cdot p14 \cdot k13 + p15 \cdot k14 + g15$
$sgn(11-8) = p11 \cdot p10 \cdot p9 \cdot p8 + p11 \cdot p10 \cdot p9 \cdot k8 + p11 \cdot p10 \cdot k9 + p11 \cdot k10 + k11$
$sgn(7-4) = p7 \cdot p6 \cdot p5 \cdot p4 + p7 \cdot p6 \cdot p5 \cdot k4 + p7 \cdot p6 \cdot k5 + p7 \cdot k6 + k7$
$sgn(3-0) = p3 \cdot p2 \cdot p1 \cdot p0 + p3 \cdot p2 \cdot p1 \cdot k0 + p3 \cdot p2 \cdot k1 + p3 \cdot k2 + k3$

[Sign extractor circuit - 16 bit CSA number]



Summary

CSA can give us the advantage of carry-free or limited carry propagation - very fast addition
Addition operation is independent of wordlength
Higher radix redundant arithmetic (in particular radix-4) has been successfully used in high-speed datapath design
Finding sign of a CSA number is difficult - the sign is given by the most significant non-zero digit
More than one bit (more wires) are needed for representing a digit
~~Fast~~ Conversion to conventional 2's complement requires traditional adders
Not good for non-arithmetic purpose

No.

Date

⇒ Low Power Design

Short Circuit Power Dissipation

The input of a real life logic gate (e.g., an inverter) is actually driven by input waveforms that has finite rise and fall time
- Both the NMOS and PMOS transistors may conduct simultaneously for a very short time during switching, forming a direct current path between the supply and ground.
  - The NMOS starts conducting when rising input voltage exceeds the threshold voltage of the NMOS transistor
  - The PMOS remains "on" until the input reaches the voltage level ($V_{DD}-|V_{tp}|$), creating a time window both when both transistors are on.

The current drawn from the supply when the PMOS and NMOS transistor both conduct does not contribute to the capacitance charging
- Hence the name short ~~current~~ circuit current.

Maybe reduced by making the output voltage transition time longer and/or by making input voltage transition time smaller : should be balanced carefully

Leakage Power dissipation

NMOS and PMOS transistors generally have nonzero reverse leakage and subthreshold currents.
- So even when the gate is idle, it still draws a very small amount of current.
  - Mainly controlled by the process parameters
  - For a chip having millions of transisters, this leakage power dissipation may contribute significantly.

Two main components : biggest concern in deep submicron (ID微缩趋势)
- Reverse gate leakage current
  - Tunnelling of electrons through gate oxide (OgA))
- Subthreshold current
  - A weak current that flows due to carrier diffusion between the source and drain of a transistor in weak inversion region (VGS is less than the threshold voltage but not exactly zero)
  - If VGS is smaller than but very close to the threshold voltage then the subthreshold power dissipation may became comparable in magnitude to the switching power dissipation

Process level:
- channel level
  - Scaling the vertical dimension of the MOSFET like oxide thickness, junction depth
  - Substrate doping : to control the depletion width
- Well engineering
  - change of doping profile in the channel : retrograde doping (1-D characteristics of well profile), halo doping (localized 2-D dopant distribution near S/D extension)

Circuit level : Controlling voltages at different device terminals.
- Standy leakage control using transistor stack
- Multiple threshold design
- Sleep transistor insertion

No.

Date

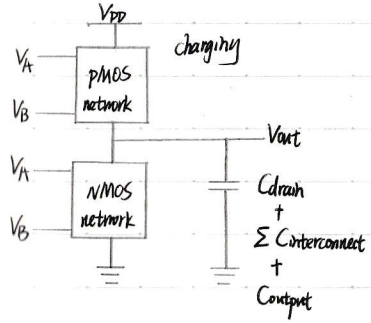
Low Power Design

Why

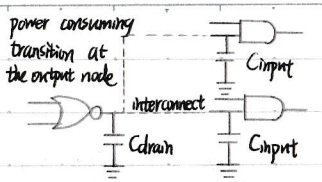
- Increasing prominence of portable and mobile systems
  - Requirement of low power dissipation to maximise battery life
- High-performance microprocessors and DSPs require complex chips with high clock frequency
  - Power dissipation and hence temperature increase linearly with frequency and complexity
- Reliability
  - Close correlation exists between the peak power dissipation of digital circuits and reliability
- General-purpose microprocessors and DSPs are not a good solution
  - Each instruction execution typically requires a number of clock cycles and thus, the power dissipation is at ~~less~~ least 1 or 2 orders of magnitude higher than the hardware approach

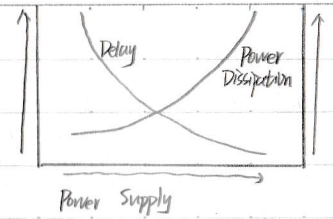
Technology level
- Selection of right device geometry, interconnect and process
- Circuit level
  - proper choice of circuit design styles, reduction of voltage swing, appropriate clocking strategy
- Architecture level
  - Smart power management strategy for various blocks, utilization of pipelining (流水线) and parallelism (同时发生) and reduction of global interconnects
- Algorithm level
  - Proper selection of the data processing algorithms, specifically to minimise the number of switching events for a given task
- Dynamic Power
  - Power dissipation due to switching of different nodes of a CMOS circuit
  - Predominant power dissipation for CMOS circuits up to 0.13μm
- Short-circuit power dissipation
  - Power ~~loss~~ dissipation due to non-ideal switching characteristics of CMOS logic gates
- Leakage power
  - Power dissipation due to non-zero reverse leakage and subthreshold currents in the NMOS and PMOS transistors
  - Becoming nearly same as the dynamic power dissipation for deep sub-micron technology

Dynamic (Switching) Power Dissipation



Dynamic Power
- Dynamic power is dissipated when energy is drawn from power supply to charge up the output node capacitance
- During the charge-up phase, the output voltage typically makes a full rail transition from 0 to V_{DD}, and the energy drawn is relatively independent of the function performed by the circuit
- In a complex structure the load capacitance that has to be switched typically consists of
  - The output capacitance of the driving gate itself
  - Total interconnect capacitance
  - The input ~~characteristic~~ capacitances of the driven gates.





$T \propto V_{DD}/(V-V_t)^{\alpha}$

α is a technology dependent parameter and varies between 1 & 1.7

The most effective way of power reduction is the reduction of supply voltage, BUT delay will increase

Dynamic Power Dissipation Expression

Average dynamic power can be calculated by considering a step input where the rise and fall times are much less than the pulse repetition period.

Let frequency $f = 1/t_p$, t_p being the time period

$$P_d = \frac{1}{t_p}\int_0^{t_p/2} i_n(t)\, V_{out}\, dt + \frac{1}{t_p}\int_{t_p/2}^{t_p} i_p(t)\,(V_{DD}-V_{out})\, dt$$

$$i_n = C_L\frac{dV_{out}}{dt} \quad \text{and} \quad i_p = C_L\frac{d(V_{DD}-V_{out})}{dt}$$

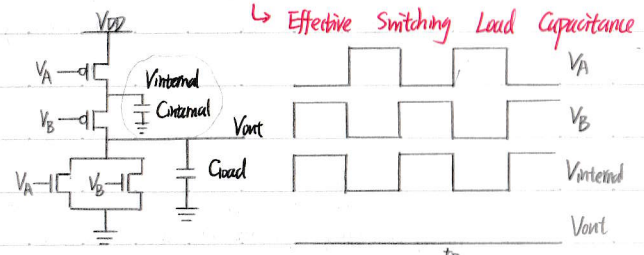
Substituting these values we get

$$P_d = \frac{C_L}{t_p}\int_0^{V_{DD}} V_{out}\, dV_{out} + \frac{C_L}{t_p}\int_{V_{DD}}^{0}(V_{DD}-V_{out})\, d(V_{DD}-V_{out}) = \frac{C_L V_{DD}^2}{t_p} = C_L\cdot V_{DD}^2\cdot f$$

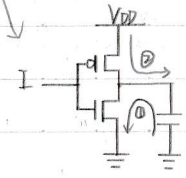
So the average dynamic power is proportional to the load capacitance and frequency and quadratically proportional to supply voltage

$$P_d = \sum_i G_i \alpha_i V_{DD}^2 f$$

↳ Effective Switching Load Capacitance



Internal nodes are making power consuming transistors transitions but output node is not showing the effect effect



I=1: ① $P_d = \frac{1}{t_p}\int_0^{t_p/2} V_{out}\cdot i_n\cdot dt$

I=0: ② $P_d = \frac{1}{t_p}\int_{t_p/2}^{t_p}(V_{DD}-V_{out})\cdot i_p\cdot dt$

$$\begin{cases} i_n = C_L\cdot\frac{dV_{out}}{dt} \\ i_p = C_L\frac{d(V_{DD}-V_{out})}{dt} \end{cases}$$

$$P_d = \frac{1}{t_p}\int_0^{t_p/2} V_{out}\cdot C_L\frac{dV_{out}}{dt}\cdot dt + \frac{C_L}{t_p}\int_{t_p/2}^{t_p}(V_{DD}-V_{out})\cdot\frac{d(V_{DD}-V_{out})}{dt}\,dt$$

$$= \frac{C_L}{t_p}\left[\frac{1}{2}V_{out}^2\right]_0^{V_{DD}} + \frac{C_L}{t_p}\int_{V_{DD}}^{0}\left[\frac{1}{2}(V_{DD}-V_{out})^2\right]_{V_{DD}}^{0}$$

$$= \left(\frac{1}{2}V_{DD}^2 + \frac{1}{2}V_{DD}^2 + 0\right)\cdot\frac{C_L}{t_p} = V_{DD}^2\cdot\frac{C_L}{t_p} = C_L\cdot V_{DD}^2\cdot f$$

$\frac{1}{4\times10^3} = \frac{10^3}{4} = 250$

Generalised average dynamic power dissipation expression

$$P_{avg} = \left(\sum_{i=1}^{\#\text{of nodes}} \alpha_{Ti}\cdot C_i\cdot V_i\right)\cdot V_{DD}\cdot f_{CLK}$$

α_{Ti} is the switching probability of the i^{th} ~~case~~ node capacitance C_i

(The assumption that all the nodes switches at every clock cycle is not correct and thus, the number of power consuming transistors for the nodes is a probabilistic function and depends on the input state.

V_i is the voltage swing of the i^{th} internal node

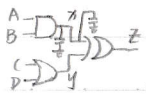
In general, $V_i = V_{DD}$

Low power, by Reduction of Switching Activity

- Switching activity (α_i) gives the probabilistic measurement of number of transitions of a particular node
  - Number of times we need to charge the capacitance at that node, i.e. power consumption
- How to calculate switching activity (activity factor)?
  1. Consider that at the primary input logic 0 and 1 have equal chance of occurance and they are uncorrelated
  2. Find out the probability of occurance of a logic 1 at the output node ($P(1)$)
  3. Find out the probability of occurance of a logic 0 at the output node ($P(0)$)
  4. The power consuming transition (switching activity) can be derived as $P_{0\to1} = P(1)\cdot P(0)$

IX

$P_{avg} = \sum_i \alpha_i C_{Li} V_{DD}^2 f$ $\quad \#\alpha(i) = P_{0\to1}(i)$



Cout = 1pf

Cin = 1pf

$P_{avg} = [\alpha(x)C_x + \alpha(y)C_y + \alpha(z)C_{(z)}]V_{DD}^2 f$

$P_1(x) = \frac{1}{4}$ $P_0(x) = \frac{3}{4}$ $\Rightarrow$ $P_{0\to1}(x) = \frac{3}{4}\frac{1}{4} = \frac{3}{16}$ $\boxed{\alpha(x)}$

$P_1(y) = \frac{3}{4}$ $P_0(y) = \frac{1}{4}$ $\Rightarrow$ $P_{0\to1}(y) = \frac{3}{4}\frac{1}{4} = \frac{3}{16}$ $\boxed{\alpha(y)}$

$Z = \bar{x}y + x\bar{y} = P_0(x)\cdot P_1(y) + P_1(x)\cdot P_0(y)$

$P_1(z) = \frac{5}{8}$ $P_0(z) = \frac{3}{8} = \frac{3}{4}\cdot\frac{3}{4} + \frac{1}{4}\frac{1}{4} = \frac{10}{16} = \frac{5}{8}$

$P_{0\to1}(z) = P_1(z)\,P_0(z) = \frac{15}{64}$ $\boxed{\alpha(z)}$

$C_x = C_{int} + C_{out} = 2\mu F$

$C_y = C_{int} + C_{out} = 2\mu F$

$C_z = C_{out} = 1\mu F$

$t_{x\text{-}delay} = 1ms$

$t_y = 2m$

$t_z = 2m$

$\Rightarrow t = 4ms \Rightarrow f = 250Hz$

ns.

$V_{DD} = 3.3V$

$P_{avg} = \left[\frac{3}{16}\cdot 2\times10^{-6} + \frac{3}{16}\times2\times10^{-6} + \frac{15}{64}\cdot1\times10^{-6}\right]\cdot 3.3V^2\cdot 250Hz$

#AND

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

$\Rightarrow$ $P('1') = \frac{1}{4}$ (probability) $\quad P('0') = \frac{3}{4}$

$P_{0\to1}(X) = P(1)\cdot P(0) = \frac{3}{16}$

#XOR

A	B	y
0	0	0
0	1	1
1	0	1
1	1	0

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

$P_{0\to1}(X) = P(1)\,P(0) = \frac{1}{4}$

AND

$X = A\cdot B$

$P_X(1) = P_A(1)\cdot P_B(1) = \frac{1}{2}\cdot\frac{1}{2} = \frac{1}{4}$

$P_X(0) = 1 - P_X(1) = \frac{3}{4}$

$P_{0\to1}(X) = P(0)P(0) = 3/16$

OR

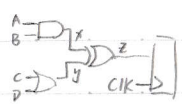
$y = A+B$

$\bar{y} = \bar{A}\cdot\bar{B}$

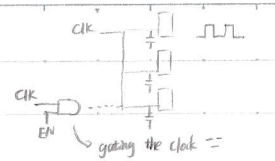
$P_B(0) = P_A(0)$ $\quad P_B(1) = \frac{1}{2}\cdot\frac{1}{2} = \frac{1}{4}$

$P_B(1) = 1 - P_B(0) = \frac{3}{4}$

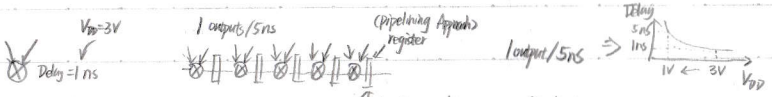
$P_{0\to1}(y) = P_B(0)\cdot P_B(1) = \frac{3}{16}$



$t_{setup} = 0.4\,ns$
$t_{hold} = 0.2\,ns$
$t = 4 + 0.4\,ns$
$f = \frac{1}{4.4}$

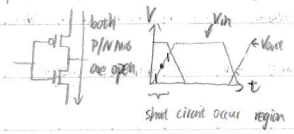


Huge current is required to change all the flip-flops when EN goes to 1 from 0

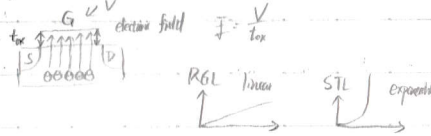


Using parallel structure to satisfy the specification constraint and then reduce V_{DD} to get the suitable delay time
↳ become faster ↳ increase the propagation delay

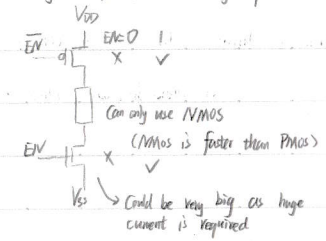
Short power dissipation (Non-ideal switching)



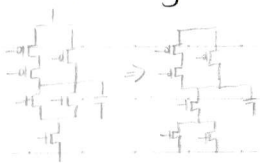
Leakage Power Dissipation



Best way to reduce the leakage power



Circuit Restructuring to Reduce Power (Stuck flipping)



Logically equivalent circuits could have different power consumptions

The one of the right is better because of low capacitance on o/p node

Power Reduction by Clock Gating

Power consumption breakdown in MCU and ASIC

Clock goes to all the registers in the chip. Even if some part of the chip is not working, still it is dissipating power because of clock activity at the input of the register : Total wastage

Reduce signal toggling when when unnessary

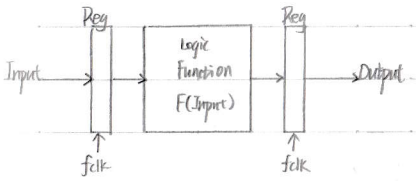
※ Cons :
Increases validation complexity
Timing budget may not allow it

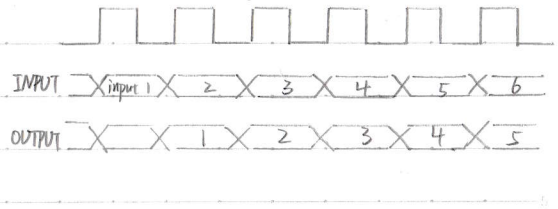
Low power Design through Voltage Scaling

Most effective way to reduce power dissipation
- Results in a quadratic improvement in power performance
- Given a certain technology, circuit designers may use on-chip DC-DC convertors and/or separate power pins to achieve this

Problem
- Reduction of power supply increases the propagation delay
  This is due to the threshold voltage cannot be scaled to the same extent as the supply voltage.
- Linear scaling of the threshold voltage with the supply voltage is supposed to fix the problem, but...
  Threshold voltage scaling is restricted by the concerns on noise margins (smaller threshold voltage : smaller noise margin)
  Threshold voltage < 0.2 : subthreshold conduction may become a very significant component of overall power dissipation

Low Power Design Using Pipelining Approach





Total capacitance (C_{total}) consists of
- The capacitances switched in the input register array
- The capacitances switched to implement the logic function
- The capacitances switched in the output register array

The expression for power dissipation

$$P_{reference} = C_{total} \cdot V_{DD}^2 \cdot f_{clk}$$

Create an N-register pipeline for the circuit
- Logic function F(Input) has to be partitioned into N successive stage each having approximately equal delay
- A total of (N-1) register arrays have to be introduced in addition to the original input and output registers
- All registers are clocked at f_{clk} (through it is possible to increase clock speed to N.f_{clk}) and thus, delay for each logic block can be increased by a factor of N without sacrificing ~~through output~~ throughput

Physically, we are running the logic segments below their actual speed capacity
- This gives us a chance to lower the supply voltage while maintaining the clock rate of f_{clk}
  - C_{reg} represents the capacitance switched by each pipeline reg

$$P_{pipeline} = [C_{total} + (N-1) C_{reg}] \cdot V_{DD,new}^2 \cdot f_{clk}$$

Total power reduction factor : $\frac{P_{pipeline}}{P_{reference}} = \left[1 + \frac{C_{reg}}{C_{total}}(N-1)\right] \cdot \frac{V_{DD,new}^2}{V_{DD}^2}$

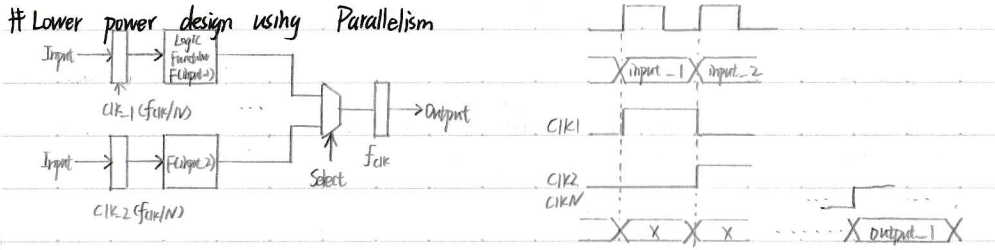
[Example]

Consider replacing a single-stage logic block (V_{DD}=5 V, f_{clk} =20MHz) with a four stage pipeline, running same f_{clk}
- The propagation delay of each pipeline stage can be increased by a factor of 4 without sacrificing data throughput
- Assume the magnitude of threshold voltage ※ for each transistor is 0.8V
- The designed desired speed reduction (4 times) can be achieved by reducing the supply voltage from 5V to ≒2V
- Assume C_{reg}/C_{total} = 0.1

Then the overall power reduction factor = $\frac{1}{5}$ $(1+0.1\cdot 3)\frac{4}{25} = \frac{5.2}{25}$

Thus a dynamic power saving of about 80% is achieved while maintaining the same throughput

Lower power design using Parallelism



Assume that the consecutive input data arrive at the same rate as in the single-stage case
The input data are routed to all registers in N processing blocks
- Gated clock signals with clock period of NT_{CLK} are used to load each register every N clock cycles.
  - The individual clock signals are skewed by T_{CLK}, such that each one of the N consecutive data is loaded into a different input register
  - Each register is thus clocked at a lower frequency f_{CLK}/N, the time allowed to compute the function for each input data is increased by a factor of N

The supply voltage can be reduced until the path delay equals the new clock period of NT_{CLK}

$$P_{parallel} = \left[1 + \frac{C_{reg}}{C_{total}}\right] \cdot C_{total} \cdot V_{DD,new}^2 f_{CLK}$$

$$\frac{P_{parallel}}{P_{reference}} = \frac{V_{DD,new}^2}{V_{DD}^2} \cdot \left[1 + \frac{C_{reg}}{C_{total}}\right]$$

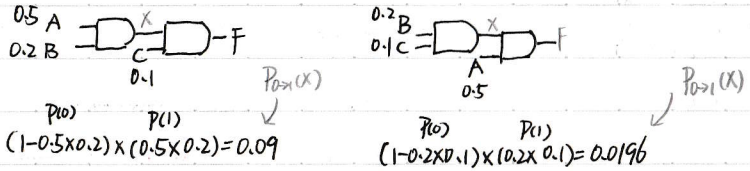
A judicious balancing between the pipelining and parallelism depending on specific circuit circumstances is needed to make the architecture efficient in terms of power dissipation

Penalty of Pipelining and Parallelism

Pipeline : ① Increasing area
② Increase of latency from 1 cycle to N cycles

Parallelism : ① Increase area and latency
② Silicon area will grow faster than the number of processor because of signal routing and the overhead circuitry

Input Reordering to reduce Switching Activity



$P_{0\to1}(X)$ $P_{0\to1}(X)$

Principles of Timing in Synchronous Systems

Explain the difference between Synchronous and Asynchronous systems

Introduction

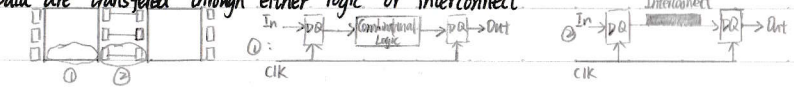
Circuit design style can be classified into two major categories, which differ in how they indicate when the signals are in a stable 0 or 1 state (and thus are ready to be sampled reliably):

- Synchronous methodologies globally distribute a timing signal (a clock) to all parts of the circuit. Transitions (rising and/or falling depending on the design) on this clock line indicate moments at which the data signals are stable
- Asynchronous methodologies are fundamentally different; there is no common and discrete time. Instead the circuits use handshaking between their components in order to perform the necessary synchronization, communication, and sequencing of operations

Asynchronous systems don't have Clock Generator

Synchronous Systems (Logic vs. Interconnect)

In synchronous system all memory elements in the system are simultaneously updated using a globally distributed periodic synchronization signal (i.e., a global clock signal)
Data are transfered through either logic or interconnect



memory element used

This functionality is ensured by strict timing constraints applied to the design. These constraints depend on the type of

Explain the timing constraints for synchronous systems (setup time, hold time)

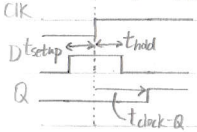
Flip-flop Based Synchronous Design

Q becomes D after clock edge

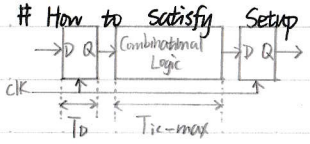
Timing Constraint: The data applied to the D flip-flop should be stable for at least T_{setup} and should stay stable for at least T_{hold}

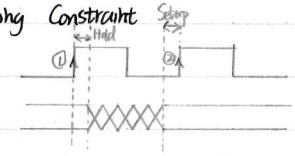
Setup Time (T_{setup}): is the minimum amount of time the data should be stable before the application of the clock signal

Hold Time (T_{hold}): is the minimum amount of time the data should be stable after the application of the clock signal



How to satisfy Setup Timing Constraint





Setup Time Constraint (for ideal clock):

$$T_{CLK-min} \geq T_D + T_{IC-max} + T_{setup}$$

$T_{CLK-min}$: the minimum clock period
T_D : the delay of the flip-flop
T_{IC-max} : the critical path delay

Holding Constraint

$$T_{Hold} \leq T_D + T_{IC-min}$$

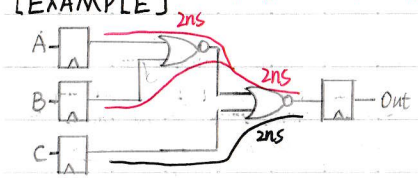
T_{IC-min}: the minimun IC delay (contamination delay)

No.

Date

※Calculate the maximum frequency of a digital synchronous circuit

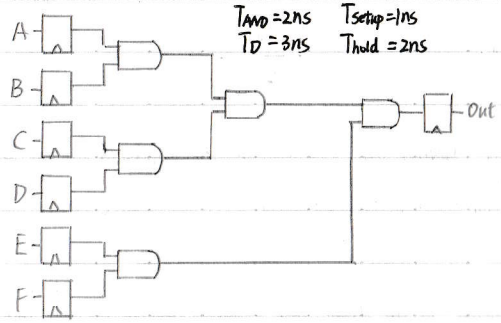
[EXAMPLE]



$T_D = 3ns$
$T_{NAR} = 2ns$
$T_{setup} = 1ns$
$T_{hold} = 2ns$

① $T_{CLK} \geq T_D + T_{IC-max} + T_{setup}$
$T_{CLK} \geq 3 + 4ns + 1ns$
$T_{CLK} \geq 8ns$
$f_{max} = 1/T_{CLK} = 125MHz$

② $T_{hold} \leq T_D + T_{IC-min}$
$2ns \leq 3ns + 2ns$
$2ns \leq 5ns$



$T_{AND} = 2ns$ $T_{setup} = 1ns$
$T_D = 3ns$ $T_{hold} = 2ns$

① $T_{IC-max} = 2ns + 2ns + 2ns = 6ns$

② $T_{IC-min} = 2ns + 2ns = 4ns$

③ $T_{CLK} \leq T_D + T_{IC-max} + T_{setup} = 3ns + 6ns + 1ns = 10ns$
$f_{CLK} = 1/T_{CLK} = 100MHz$

④ $T_{hold} \leq T_D + T_{IC-min} = 3ns + 4ns = 7ns$

⑤ $T_{AND} = 0.5ns$ $T_D = 0.5ns$
$T_{IC-min} = 0.5ns + 0.5ns = 1ns$
$T_{hold} = 2ns \leq T_D + T_{IC-min} = 1.5ns$ X

※Perform timing analysis that includes interconnect effects

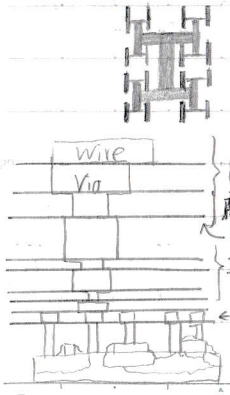
The Role of Interconnect

Bues are implemented as long metal lines on a silicon wafer transmitting data using electromagnetic waves (finite speed limit)

In addition to data transfer Metal lines (interconnect) are also used power distribution networks and for clock distribution network in synchronous design.

Interconnect networks have become increasingly complicated with greater Integration

Computation Vs. communication



Operation	Delay (0.43μm)	(0.15μm)
32 ALU Operation	650ps	250ps
32 Register Read	325ps	125ps
Read 32b from 8K RAM	780	300
Transfer 32b across chip (10mm)	1400	2300
Transfer 32b across chip (20mm)	2800	4600

2:1 global on-chip comm to operation delay

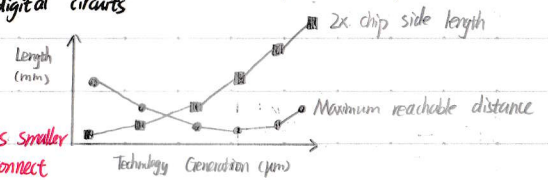
No.

Date

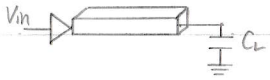
The impact of interconnect in Modern Technologies

Limit Performance: Signals cannot travel across the entire 横具 die within a global clock cycle.
For example, 6-10 clock cycles are needed to transfer data on a bus at 50 nm technology
Unpredictability in signal propagation time has serious consequences for performance and correct functioning of synchronous digital circuits
Consume up to 50% of chip power
Affect system reliability (crosstalk errors)

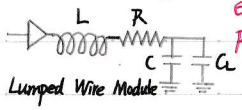
⇒ The figure shows that
Chip size increases while the technologies gets smaller
This leads to increased delays of global interconnect



Paracitic Modelling



Wire Modelling

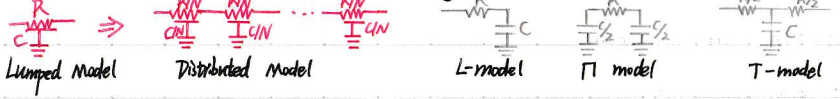


Lumped Wire Module

Accurate and efficient models are essential to estimate the impact of interconnect on design budget (delay, power, reliability) at early stage of the design.
Most design tools use RC models.

Distributed Models Vs Lumped Models

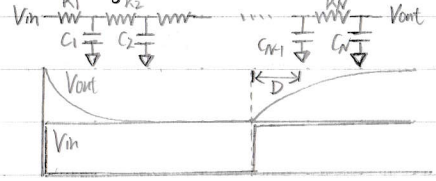
Thin on-chip are often modeled by distributed RC lines rather than lumped model
Three standard approximations are L-model, T-model, and Π model, so named because of their space.
The Π model is the most accurate, three segments are sufficient around for 97% accuracy level.



Lumped Model Distributed Model L-model Π model T-model

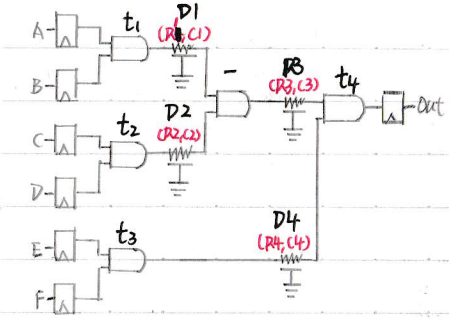
Wire delay modelling

Elmore analyzed the distributed model and came up with figures for delay in 1948 before VLSI
Elmore delay which is taken to be the time between the 50% crossings of the input and output



$D_{wire} = 0.4 * R_{wire} * C_{wire}$

[Exercise]



The circuit in figure 1 is synchronous and uses positive edge triggered flip-flops.
The delay of each logic gate is 0.1ns.
The delay of each flip-flops is 0.3ns
T_{setup} = 0.1ns T_{hold} = 0.1ns

$1F = 10^{-15}F$

$R_1 = R_2 = 500\Omega$, $R_3 = 400\Omega$ $R_4 = 1500\Omega$
$C_1 = C_2 = 100fF$, $C_3 = 80fF$ $C_4 = 250fF$

Calculate the maximum achievable clock Frequency F_{max}

$D1 = D2 = 0.4 * 500 * 100 * 10^{-15} = 0.02ns$

$D3 = 0.4 * 400 * 80 * 10^{-15} = 0.0128ns$

$D4 = 0.4 * 1500 * 250 * 10^{-15} = 0.15ns$!!! $D4 > T_{AND}$

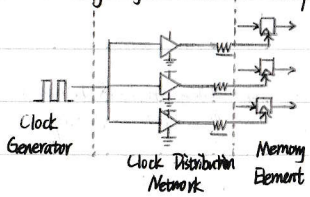
$T_{IC-max} = t_3 + D4 + t_3 = 0.35ns$

$T_{clk-min} \geq T_D + T_{IC-max} + T_{setup} = 0.3 + 0.35 + 0.1 = 0.75ns$

$F_{max} = 1/T_{clk-min} = 1.33GHz$

※Explain the principles of clock generation

Clocking System Main Components



Clock Generation: Usually a system on a chip receives one or more primary clock signals from an external clock chip and, in turn, generates necessary div derivatives for its internal use

Clock Distribution Network: The primary design goal in clock distribution networks is to ensure that a clock signal arrives at every register within the entire synchronous system at precisely the same time (i.e. reduce clock skew and jitter)

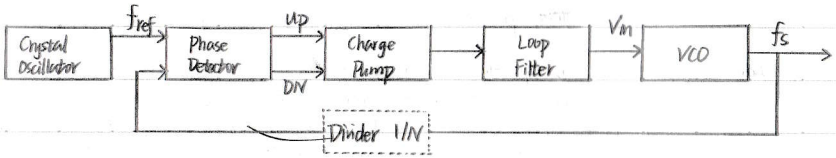
Memory Elements: Store the data values of the output of combinational logic blocks

Clock Generation : PLL Basic Operation

Phase-Locked Loop is a control system that generates a signal that has a fixed relation to the phase of a "reference" signal. A phase-locked loop circuit responds to both the frequency and the phase of the input signals, automatically raising or lowering the frequency of a controlled oscillator until it is matched to the reference in both frequency and phase

Crystal Oscillator → f_{ref} → PLL → f_s

Clock Generation : PLL Basic Operation



Phase-locked loop (PLL) is typically used to generate the high frequency required by SoC. A PLL takes an external low-frequency reference crystal frequency signal and multiplies its frequency by a rational number N

The reference clock is typically generated off-chip from an accurate crystal reference. The reference clock is typically compared to a dived version of the system clock (i.e. the divided clock) using a phase detector that compares the phase difference between the signals and produces an Up or Down signal when the local clock lags or leads the reference signal. It detects which of the two input signals arrives earlier and produces an appropriate output signal.

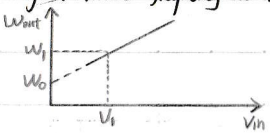
Next, the Up and DN signals are fed into a charge pump, which translates the digital encoded control information into an analogue voltage, An Up signal increases the value of the control voltage and speeds up the VCO, which causes the local signal to catch up with the reference clock. A DN signal, on the other hand, shown down the oscillator and eliminates the phase lead of the local clock.

The Loop filter removes the high frequency components from the VCO control voltage and Smooth's out its response

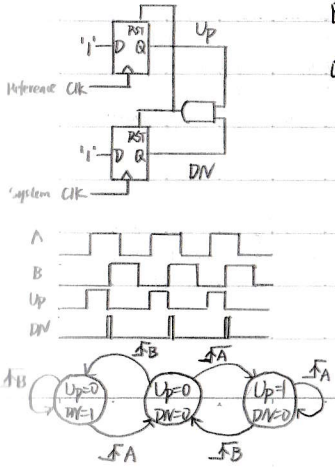
Voltage Controlled Oscillator

An ideal VCO generates a periodic signal whose frequency is a linear function of its input voltage

$W = W_0 + k_{VCO} \cdot V_{in}$
↳ gain of the VCO
↳ fixed frequency offset



Phase Frequency Detector

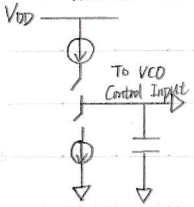


Function: It determines the relative phase and frequency difference between two incoming signals and outputs a signal that is proportional to this difference

Operation: When input A leads B, the Up output is asserted on the rising edge of input A. The Up signal remains in this state until a low to high transition occurs on input B. At that time, the DN output is asserted, causing both flip-flops to reset through the asynchronous reset signal. The duration of the small pulse on the DN output is equal to the delay through and the AND gate and register reset delay. The pulse width of the UP pulse is equal to the phase error between the two signal. The roles are reversed for the case when input B leads A. The circuits also acts as a frequency error detector. For the case when A is at a higher frequency than B, the PFD generates a lot more UP pulses while the DN pulses average close to zero. The opposite is true for the case when B has a frequency larger than A — many more pulses are generated on the DN output than the UP output.

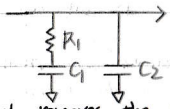
Charge Pump

Function: It converts the UP/DN signal pulses into an Analog voltage that controls the VCO.



Operation: A pulse on the Up signal add a charge packet to the storage capacitor proportional to the size of the UP pulse, and a pulse on the DN signal removes a charge packet proportional to the DN pulse. If the width of the UP pulse is larger than the DN pulse, then there is a net increase in the control voltage. This effectively increase the frequency of the VCO.

Loop Filter



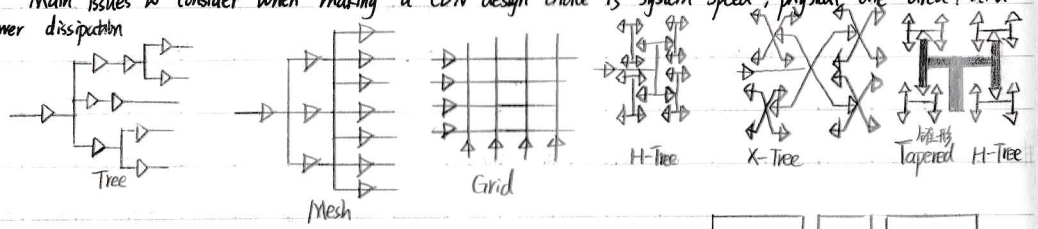
Function: Low-pass filter that removes the high-frequency components from the VCO control voltage and smoothens out its response, which results in a reduction of the jitter (jitter is the temporal variation in the signal period). It is usually made of passive components

Clock Distribution Networks (CDN)

Design Requirement: To distribute a tightly controlled clock signal to each synchronous register on a large non-redundant hierarchically structured integrated circuit

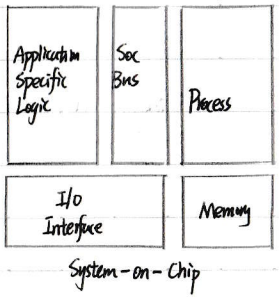
Solutions: include the use of asymmetric structures such as buffered tree or symmetric structure such as H-trees

Main issues to consider when making a CDN design choice is system speed, physical die area, and power dissipation



Symmetric CDN structure such as H-tree and X-tree are difficult to implement in irregular VLSI-based systems due to routing constraints and different fan-out requirements.

In these types of systems, buffered tree topologies integrated with structured custom design methodologies should be used



Clock Tree Synthesis for ASICs

Modern back-end tools include clock tree synthesis
- Creates balanced RC-trees
- Uses special clock buffer standard cells
- Can add clock shielding
- Can exploit useful clock skew

Consider a zero-skew clock routing network that was constructed based on estimated routing parasitic

- The primary design goal in clock distribution networks has been to ensure that a clock signal arrives at every register within the entire synchronous system at precisely the same time (i.e. reduce clock skew and jitter)
- Reduce Power: In many high-speed process processors, a majority of the dynamic power is dissipated in the clock network. One option to reduce dynamic power is to use clock gating (shutting down (disabling the clk) unit)
- Reduce Area
- Clock Slew Rate: To maintain signal integrity minimum skew (转换率) rates are required. However, aggressive reduction of slew may lead to increase power consumption and signal reflections.

Optimise the power consumption for the clocking systems

Power reduction techniques

Dynamic Power Consumption of a clocking scheme can be given as:

$$P_{clk} = \delta \cdot f \cdot C \cdot V_s^2$$

δ: Switching activity of the clock distribution network

f: Clock Frequency

V_s: The swing voltage of the clock signal

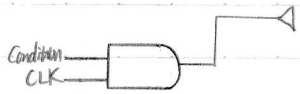
$$C = C_{CDN} + C_{ME}$$

C_{CDN}: the total capacitance of the clock distribution interconnect including all buffers

C_{ME}: the sum of the capacitances of all memory elements used in the clocking system (e.g. flip-flops or latches)

Power reduction techniques : clock gating

Principle : CG reduces switching activity by Not clocking some parts of the system if their outputs are not needed. Most clock-gating is done at the Register Transfer Level.

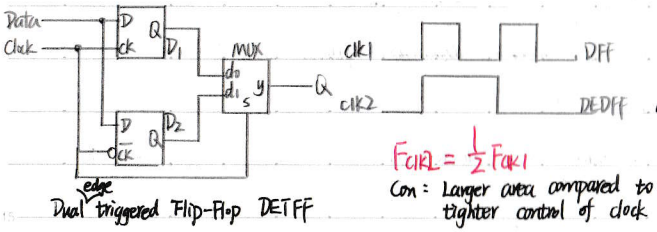


Combinational Clock Gating : reduces power by disabling the clock on registers when the output is not changing

System-level clock-gating stops the clock for an entire block, effectively disabling all functionality On the contrary, combinational and sequential selectively suspend clocking while the block continues to produce an output. Condition for disabling a certain clock is identified by the designer.

Power reduction techniques : Dual Edge Triggered Flip-Flop

Principles: Use the special types of storage elements such that the clock frequency can be reduced by half by and keep the same data rate



Dual triggered Flip-Flop DETFF

load D1, 次 D2

↑ D1 transparent, Q ← D2

↓ D2 transparent, Q ← D1

Adv: Can be used to reduce clock frequency by half and keep the same data rate
Saving Power

$F_{CLK2} = \frac{1}{2} F_{CLK1}$

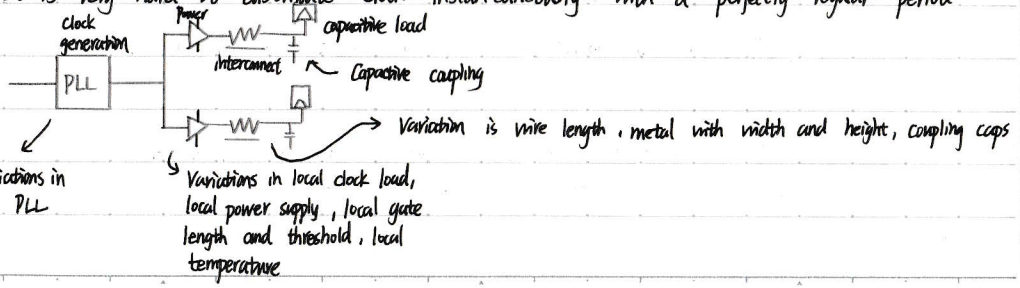
Con: Larger area compared to edge triggered flip-flops and require high tighter control of clock duty cycle.

Perform timing analysis for Synchronous Systems with skew and jitter

~~# FF based Synchronous Design~~

Clock Signal is not ideal

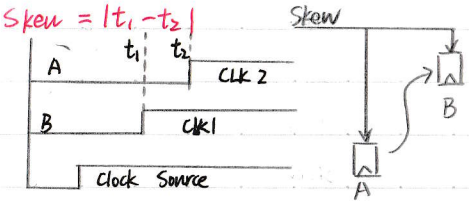
It is very hard to distribute clock instantaneously with a perfectly regular period



Clock Skew : Spatial Clock Variation

Clock skew Static (constant over time) point-to-point variation (of clock arrival time to FFs)

Clock skew between two flip-flops is computed as the difference in arrival times of clock signal



Causes of Clock Skew

Designed Variations : mismatch in buffer load sizes, interconnect lengths

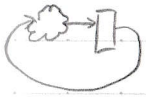
Process Variations : process speed spread across die yielding different L_{eff}, Tox, etc. values

Temperature Gradients : change MOSFET performance across die

IR voltage drop in power supply : changes MOSFET performance across die.

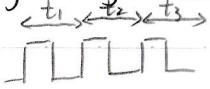
Clock Jitter : Temporal Clock Variation

Clock Jitter is a variation in clock edge timing between clock cycle i.e. Cycle-to-Cycle variation



Jitter $J_1 = t_2 - t_1$

Jitter $J_2 = t_3 - t_2$



Sources of Jitter

Variations in PLL oscillation frequency : This is an analogue circuit and highly sensitive to intrinsic device noise and power supply variations. A major problem is the coupling from the surrounding noisy digital circuitry through the substrate.

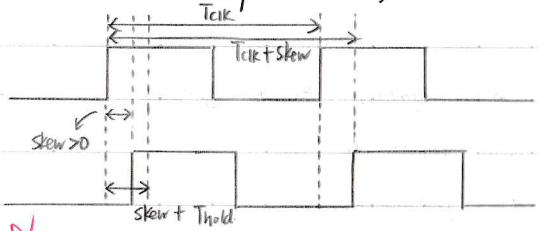
High Frequency Power Supply Variations : This is caused by instantaneous IR drops along the power signal grid due to fluctuation (ばらつき) in switching activity can affect the delay of buffers in the clock distribution network which lead to temporal variation in clock cycle.

Variation In Capacitive Load : Due to coupling between the clock lines and adjacent (隣接) signal wires. The exact coupling to the clock network is not fixed from cycle-to-cycle.

Skew vs Jitter

① Uncertainty in the clock generation circuit — Jitter
② Process variation in devices — Skew
③ Interconnect variation ← design variation — Skew
④ Power supply noise — Jitter
⑤ Data Dependent Load Capacitance — Jitter
⑥ Static Temperature Gradient — Skew

Positive Skew Impact on Performance



$T_{clk-min} + Skew \geq T_D + T_{ic-max} + T_{setup}$

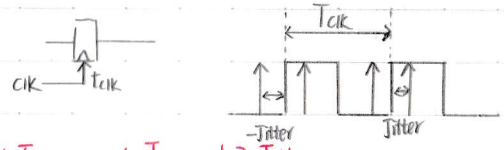
$T_{hold} \leq T_D + T_{ic-min} - Skew$

Skew >0 : Improves performance, but makes Thold harder to meet

If Thold is not met (race conditions), the circuit malfunctions in dependent of the clock speed.

Jitter Impact on Performance

Jitter causes T to vary on a cycle-by cycle basis



$T_{clk} \geq T_D + T_{ic-max} + T_{setup} + 2\,Jitter$

$T_{Hold} \leq T_D + T_{ic-min} - 2\,Jitter$

Jitter directly reduces the performance of a sequential circuit.

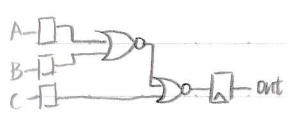
Combined Impact

$T_{clk} \geq T_D + T_{ic-max} + T_{setup} - Skew + 2\,Jitter$

$T_{hold} \leq T_{hold_D} + T_{ic-min} - Skew - 2 \cdot Jitter$

Skew >0 with jitter: Degrades performance, and makes Thold even harder to meet

[EXAMPLE]



T_D=3ns
T_{wire}=2ns
T_{setup}=1ns
T_{hold}=2ns
(1) Skew=1ns
Jitter=1ns

$T_{clk} \geq T_D + T_{ic-max} + T_{setup} + 2J - Skew = 3+4+1+2-1 = 9$

$f_{max} = \frac{1}{T_{clk}} = 111.11\,MHz$

$T_{hold} = 2 \leq T_P + T_{ic-min} - 2J - Skew = 3+2-2-1 = 2$

※ Differentiate between Critical path and False Path

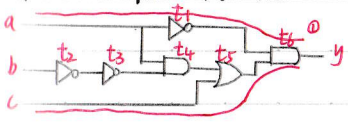
Timing Analysis (There are two main types of timing verification)

Dynamic Timing Analysis: Circuit timing is verified by means of exhausive simulation of the circuit using actual gate delay. The test vectors to the inputs of a circuits, then the changes in all gates and followed through to the outputs in order to find the worst case time. It can take an extraordinary amount of time to run.

Static Timing Analysis (STA): It does not need simulations, the delay of all paths are computed using gate delay information to check for any setup or hold time violations. This method runs quickly due to the use of simplified delay models, and because of its limited consideration of the effects of logical interactions between signals. It is used extensively in all EDA tools.

STA suffer from false path problem which may lead to 保守的 conservative estimation of circuit performance

Critical Path v.s False Path



① $T_{ic} = t_1 + t_6$
② $T_{ic-max} = t_5 + t_6$

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

The clock speed is determined by the slowest feasible path between registers in the design referred to as Critical path

A false path is a path which cannot be exercised due to Boolean gate conditions or it is not relevant to the circuit path. False paths cause pessimistic timing constraints ↑悲観的

Because of these effects, static timing analysis might be overly conservative and predict a delay that is greater than you will experience in practice.

a=0 ⇒ bX
a=1 ⇒ b,cX

No.

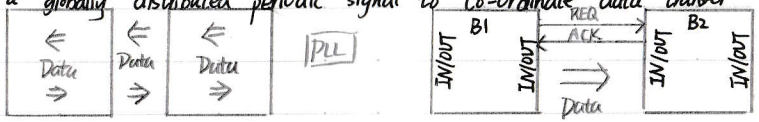
Date

Basic Principle of Asynchronous Circuit Design

* To explain the basic operation principles of asynchronous circuits

Introduction (Synchronous)

Synchronous systems uses a globally distributed periodic signal to co-ordinate data transfer among system's parts.



Asychronous Systems use handshaking protocols to co-ordinate data transfer among system's parts

Key Design Difference

Synchronous logic design:
- Proceeds without taking timing correctness (hazards, signal acking etc.) into account
- Combinational logic and memory latches (registers) are built separately
- Static timing analysis of combinational logic is sufficient to determine the Max Delay (clock period)

Asynchronous logic design
- Must ensure hazard-freedom, signal acking, local timing constraints
- Combinational logic and memory latches (registers) are often mixed in "complex gates"
- Dynamic timing analysis of logic is needed to determine relative delays between paths

To avoid complex issues, circuits may be built as Delay-insensitive and/or Speed-independent

Verification and Testing Difference

Synchronous logic verification and testing:
- Only functional correctness aspect is verified and tested
- Testing can be done with stand ATE and at low speed

Asynchronous logic verification and testing:
- In addition to functional correctness, temporal aspect is crucial: e.g. causality and order
- Testing must cover faults in complex gates (logic + memory) and must processed at normal operation rate

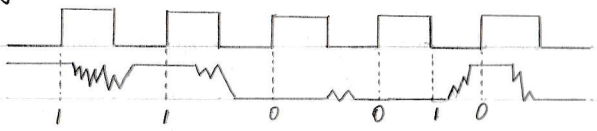
deadlock-freedom

Motivation for Asynchronous

Pro
- Low power consumption
- Less emission of electromagnetic noise (No peak current around clock edges)
- Robustness (強健性) toward variation in supply voltage, temperature and fabrication process
- Modularity for system-on-chip design (Plug and play interconnectivity)
- Average-case performance
- No clock distribution and clock skew problems

Con
- Difficult to design (Hazards, synchronization)
- Complex timing analysis (Difficult to estimate performance)
- Difficult to test (No way to stop the clock)

Synchronous communication

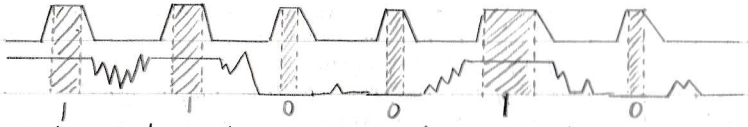


Clock edges determine the time instants where data must be sampled.

Data wires may glitch between clock edges (set-up/hold times must be satisfied)

Data are transmitted at a fixed rate (clock frequency)

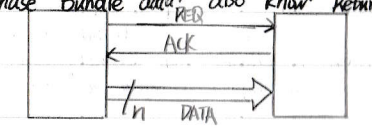
Bundled data communication



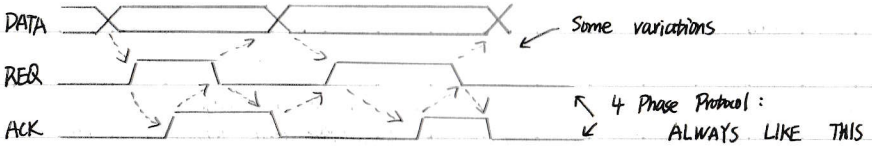
Validity signal • Similar to an aperiodic local clock.
n-bit data communication requires n+1 wires
Data wires may glitch when no valid

Bundled Data Protocol

Four Phase Bundle data: also know: Return to Zero



Push Channel (DATA & REQ SAME DIRECTION)

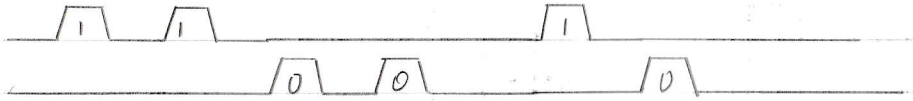


Timing Assumption of Bundled

Each data line is a single wire • "Bundled data also known as single rail"
On sender side, time(DATA) < time(REQ)
This order should be preserved on receiving end: valid(DATA) < REQ [data should be valid precedes REQ↑]
Non-trivial: wire-line skew must be taken care of and hidden

- Placement and routing
- Safety margins at sending-data end
- Buffer insertion

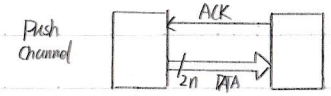
Dual rail communication



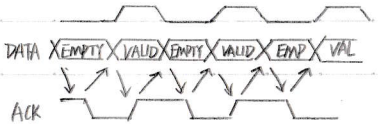
Two wires with L (low) and H (high) per bit
• "LL" = "spacer", "LH" = "0", "HL" = "1".
n-bit data communication requires 2n wires
Each bit is self-timed
Other delay-insensitive codes exist (e.g. k-of-n) and event-based signalling (choice criteria: pin and power efficiency)

4-phase dual rail protocol

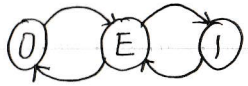
Each data bit encoded into 2 wires



VALUE	d.t	d.f
EMPTY	0	0
VALID"0"	0	1
VALID"1"	1	0
Not used	1	1



← No REQ line, but this is how it would look like if we had one



Delay Insensitive (DI) Each bit can propagate at own speed.

4 phase at higher level (than signals):

1. Sender sends valid word (V)
2. Receiver sets ACK↑
3. Sender sends empty word (E) (removes the data)
4. Receiver sets ACK↓

Each acknowledge is acknowledged / indicated

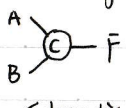
Problems: Glitches, Hazards.

No.

Date

※To describe the structure and the functionality of the Muller C-element

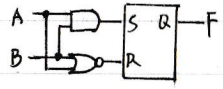
Event Logic - The Muller-C Element

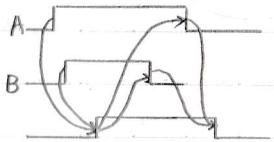


Schematic

A	B	Fn+1
0	0	0
0	1	Fn
1	0	Fn
1	1	1

Truth Table





Acknowledgement / Indication

A gate / circuit acknowledges its input if, for every input change, there is an output change (eg. wire)

Non-indicating example : AND gate { Acknowledge : {01, 10, 00} → 11
Does not acknowledge : 00 → {01, 10}

AND	
00	0
01	0
10	0
11	1

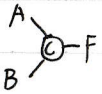
Acknowledgement / Indication

C-element

Acknowledges all ONES : {00, 10, 01} → 11}

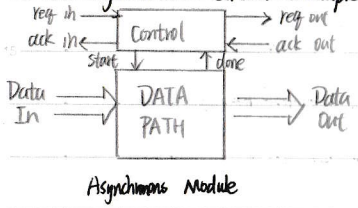
Acknowledges all Zeros : {10, 01, 11 → 00}

A	B	F
0	0	0
0	1	Fn
1	0	Fn
1	1	1



※To design asynchronous circuit for simple combinational functions based on dual rail protocol

An Asynchronous Circuit Example



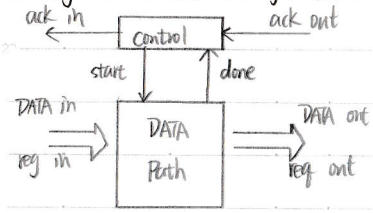
Asynchronous Module

Two essential components of asynchronous circuits:

① Control which generates the handshake signal

② Data path computes data and generates the completion detection signal

Asynchronous Modules for Dual Rail



Request is embedded in the DATA ※

Dual Rail Circuit : Datapath Design

The DATApath performs the computation and generates the completion detection signal (i.e. done) the datapath also generates the request signals. There are many techniques to design asynchronous dual rail logic. We will study one method called Delay insensitive Min-term Synthesis

Dual Rail Logic Design

Delay Insensitive Min-term Synthesis (DIMS)

↳ This technique resembles the traditional sum of products approach, but there are also few important differences :

① The min-terms are formed using C-elements (instead of AND gates)

② Reduction of the Boolean Equations by combining min-terms into simpler terms is (in general) not allowed

Together, these ~~assumes~~ requirements assure that our combinational circuits do not produce any valid output signals until all input signals are valid, and that none of the output signals change back to the empty value until all inputs are empty

DIMS Design Example

How to build a DIMS circuit for a logic function

① Write a binary truth table for the function

A	B	OUT
0	0	0
0	1	0
1	0	0
1	1	1

② Replace each binary value with its dual rail equivalence

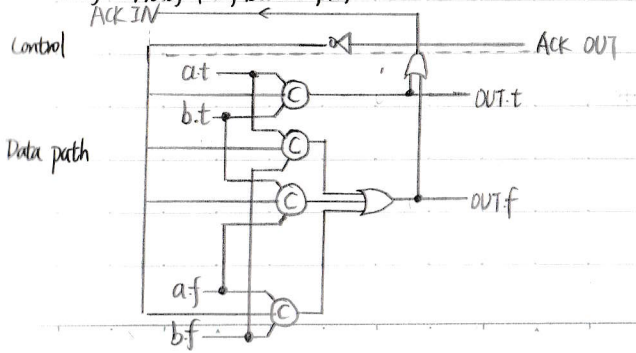
At	Af	Bt	Bf	OUTt	OUTf
0	1	0	1	0	1
0	1	1	0	0	1
1	0	0	1	0	1
1	0	1	0	1	0

③ Build your functions based on the min-terms using C-elements (instead of AND-gates) and OR gates

④ Add to this gate completion detection logic

OUTt = At . Bt

OUTf = At Bf + Af Bt + Af Bf



Dual Rail Logic Design : DIMS Design Example : Adder

① Truth Table

Cin	A	B	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

② Dual Rail Equivalence

Cin_t	Cin_f	A_t	A_f	B_t	B_f	S_t	S_f	Cout_t	Cout_f
0	1	0	1	0	1	0	1	0	1
0	1	0	1	1	0	1	0	0	1
0	1	1	0	0	1	1	0	0	1
0	1	1	0	1	0	0	1	1	0
1	0	0	1	0	1	1	0	0	1
1	0	0	1	1	0	0	1	1	0
1	0	1	0	0	1	0	1	1	0
1	0	1	0	1	0	1	0	1	0

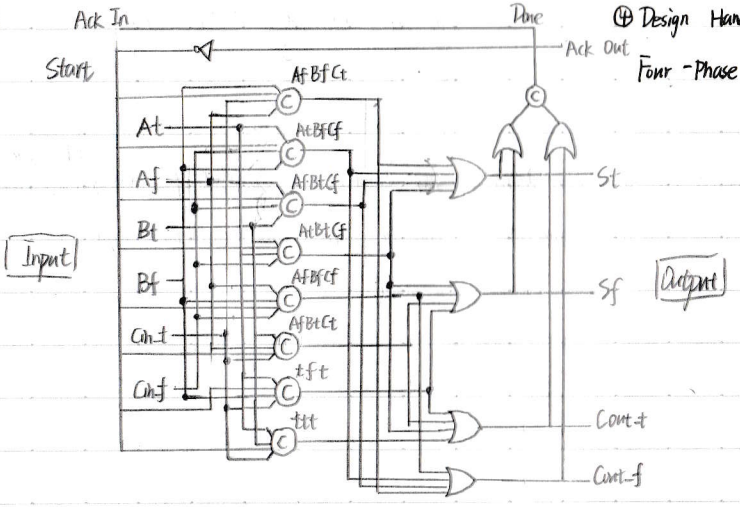
③ Build functions

$St = Af.Bt.Cin\_f + At.Bf.Cin\_f + Af.Bf.Cin\_t + At.Bt.Cin\_t$

$Sf = Af.Bf.Cin\_f + At.Bt.Cin\_f + Af.Bt.Cin\_t + At.Bf.Cin\_t$

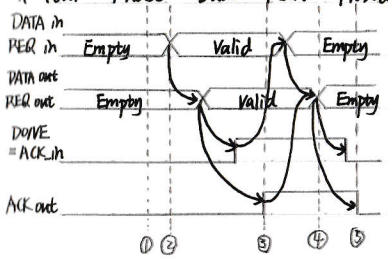
$Cout\_t = At.Bt.Cin\_f + At.Bf.Cin\_t + Af.Bt.Cin\_t + At.Bt.Cin\_t$

$Cout\_f = Af.Bf.Cin\_f + At.Bf.Cin\_f + Af.Bt.Cin\_f + Af.Bf.Cin\_t$



④ Design Handshake circuits to implement Four-Phase Dual Rail protocol

Four Phase Dual Rail Protocol



① At the Beginning

Both inputs and outputs are in the empty state

{ Input = Empty, Output = Empty } Ack_out = 0, Start = 1, Done = 0, Ack_in = 0

② Inputs become valid

③ When all input data transition to a valid state, outputs data become valid and the Done/Ack in signal goes high

{ Input = Valid, Output = Valid } Ack_out = 0, Start = 1, Done = 1, Ack_in = 1

④ When the next state of the pipeline receives the valid data, it asserts Ack out, Ack out goes high and start goes low

The circuit will wait until all the inputs transition to the empty state

After all the input transition to the empty state and Ack_out is asserted, the outputs will transition to the empty state

⑤ After all outputs transition to an empty state Ack in will be de-asserted

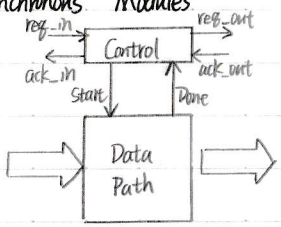
The next stage of the pipeline will de-assert Ack out when it receives the empty data, This takes us back to the initial state.

No.

Date

Logic Design for Asynchronous Control Circuit and Interfaces

Asynchronous Modules



Two essential components of asynchronous circuits:

1. Control which generates the handshake signal
2. Data path computes data and generates the completion detection signal.

Delay Modules

① Bounded delays (BD): ~~Yet~~ realistic for gates and wires

- Technology mapping is easy, verification is difficult

② Speed independent (SI): Unbounded (pessimistic) delays for gates and "negligible" (optimistic) delays for wires

- Technology mapping is more difficult, verification is easy

③ Delay insensitive (DI): Unbounded (pessimistic) delays for gates and wires.

- DI class (built out of basic gates) is almost empty

Asynchronous Control Circuits

There are numerous methods, tools and ~~difficult~~ different assumptions (e.g. delay models)

In this course:

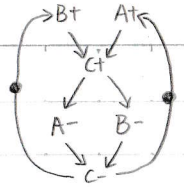
Assumption: Speed independent control circuits

Method: Signal transition graph (A special type of Petri net)

Tool: Workcraft (mostly developed at Newcastle University)

No.

Date



Firing ↓ A+ C+

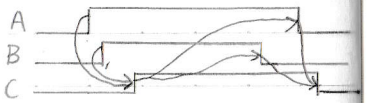
※ Transfer timing specifications into a signal transition graph

Signal Transition Graph

Vertices represent signal transitions

Arcs represent caused relations between transitions

Marking is the assignment of tokens to places and represents the state of the circuit.

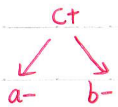


Token preservation

Tokens do not disappear
Tokens do not appear (from nowhere)
One token does not overtake another
A signal transition with n inputs can generate m outputs
- wait for n tokens on inputs
- generate m tokens on outputs

Signal Transition Graph : Concurrency & Choice

Concurrency



a- concurrent with b-

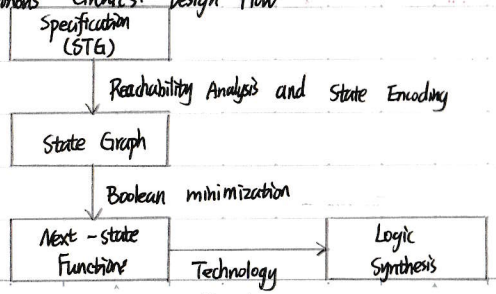
Either a- fires before b-
or b- fires before a-
or a- and b- fires simultaneously

Choice

Either a- fires
or b- fire
but not both
choice made by environment

※ Digital logic circuits for asynchronous controllers

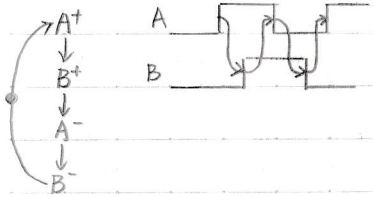
Asynchronous Circuits' Design Flow

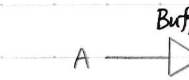


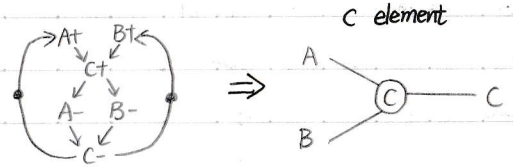
No.

Date

[Specification (STG)]





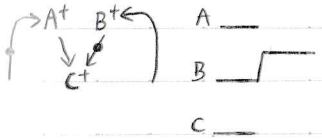


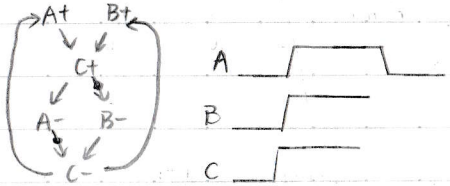
[State Graph]

Reachability Analysis

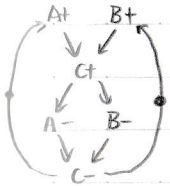
↳ The aim of this analysis is to find all reachable states for a given STG by simulating token flow

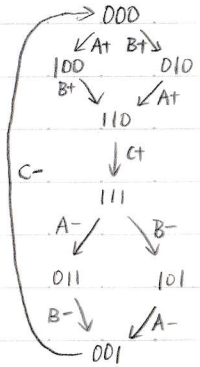
State Graph : Transition





State Graph : State Coding



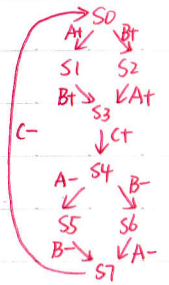


※ State Graph

State Graph : Reachability Analysis

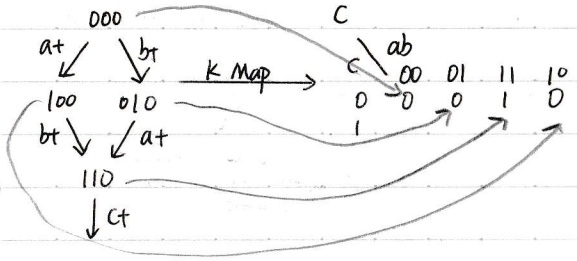
For more complex examples reachability analysis is performed before binary state coding

State coding



Reachability Graph

[Next State Functions]

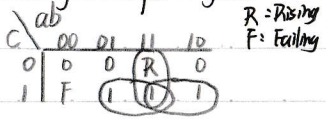


State 000:
C is assigned 0 because C stays low

State 110:
C is assigned 1 because C goes high in this state (C+ is enabled)

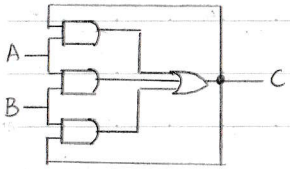
[Logic Synthesis]

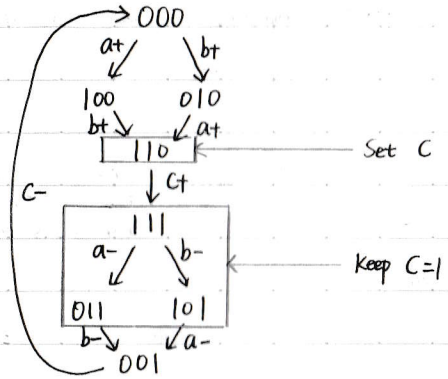
Using Complex gates



R = Rising
F = Falling

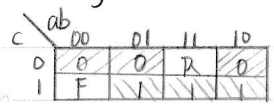
$C = ab + ac + bc$

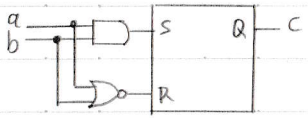


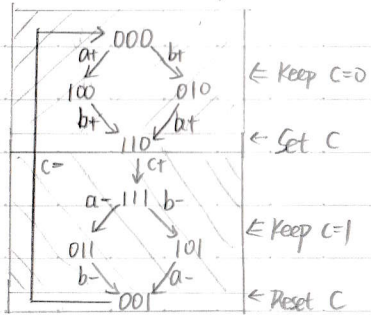


HAZARDOUS Material

Using SR latches







Campus

Logic Synthesis : using SR Latch

The set function (cover) for a variable C:
1. Must contain all states in the SET regions
2. May contain states from the KEEP C=1 regions
3. May contain states not reachable by the circuits

The reset function (cover) for a variable C:
1. MUST contain all states in the RESET regions
2. May contain states from the KEEP C=0 regions
3. May contain states not reachable by the circuits

① Needs (mutually exclusive) SET, RESET regions
② May use KEEP 0, KEEP 1 regions
③ May use ~~unreach~~ unreachable states
④ Area/performance/power may be more or less than gate circuits
⑤ May use C elements instead of SR latches
⑥ SR latches enable implementation with standard libraries

Explain synthesis conditions for speed independent circuit's specifications

Synthesis Conditions

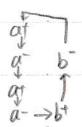
In a speed independent circuit, the delay of the gate does not affect the functionality of the circuit.

To guarantee a hazard-free speed independent implementation for an STG, the following properties must be satisfied:

1. Boundedness 有界性
2. Consistency 一致性
3. Complete State Decoding
4. Output Persistency 持久性

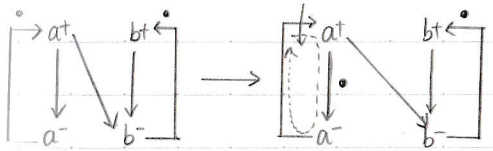
Safety:
- STG is bounded (safe) if no place or arc can ever contain more than one token
- Unbounded是 is always caused by one-side dependency

EXAMPLE: the STG below is unbounded (not safe)

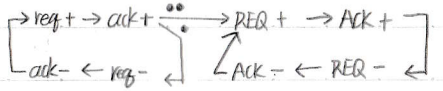


If left cycle goes fast and right cycle lags, then arc ack+ → req+ (REQ) accumulates tokens (REQ+ depends on both ack+ and Ack-)

Boundedness



Boundedness ⟷ Finite reachability graph



左边比右边快，右边会同时有N个token在运行
无限累积

KOKUYO

No.

Date

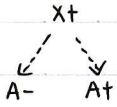
Consistent State Assignment (CSA)

The subset of an STG makes no sense

INCONSISTENT

A+ ----> A+ Two subsequent up-transitions without a down-transition in between

A- ----> A- Two subsequent down-transitions without a up-transition in between



Parallel transitions of the same signal

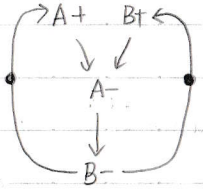
CONSISTENT

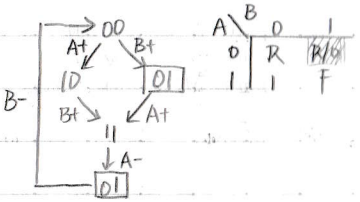
A+ ----> A- ----> A+

Two up-transitions with a down-transition in between (and vice versa)

Boundedness + CSA ⇔ Finite Binary State Graph

Complete State Coding





CSC - conflict :

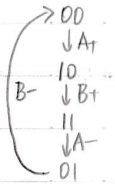
Two states have the same binary codes and different set of enabled non-input signals

Solving CSC by concurrency reduction

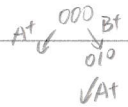
A+ → B+

⇑

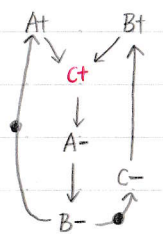
Add ARC

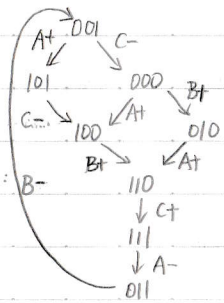


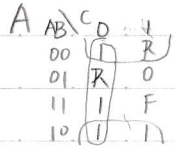
No equivalent transformation - concurrency is reduced!



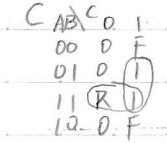
Solving CSC by Signal Insertion





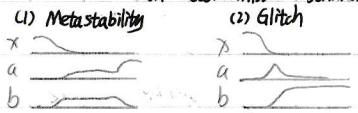


$A = \bar{B} + \bar{C}$



$C = B\,(C+A)$

Non-deterministic behaviour



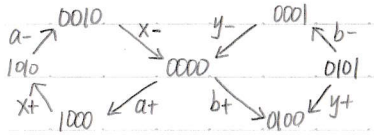
Output Persistency

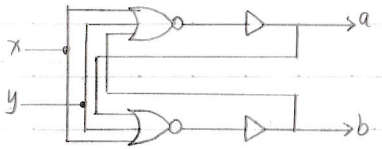
Output disables output

Assume : a, b are outputs and x, y inputs

Depend on which buffer is stronger

Speed Dependent





A state graph is a said to be output-persistent if no event can be disabled by another event (unless they are both ~~a~~ inputs).

This property determines whether or not the circuit is speed independent

Violations of Implement ability

Violations	Meaning	How to correct	Type of Implementation
Boundedness	Infinite Reachability Graph	Change Spec	N/A
Consistency	No binary state Graph	Change Spec	N/A
CSC (reducible)	No Logic	Add Signals or Change Spec	Same inputs and outputs
Persistency	Hazards	Change Spec	No speed independent

No.

Date

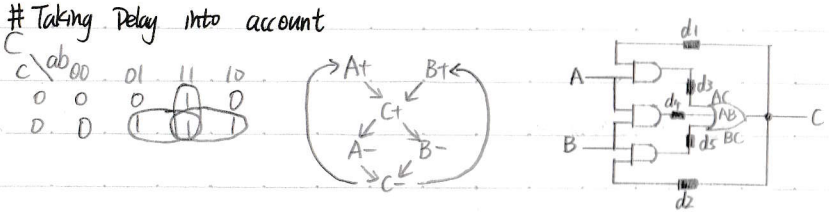
Verify functional correctness under different delay models.

Gate vs wire delay models

Gate delay model: delays in gates, no delay in wires

Wire delay model: delays in gates and wires

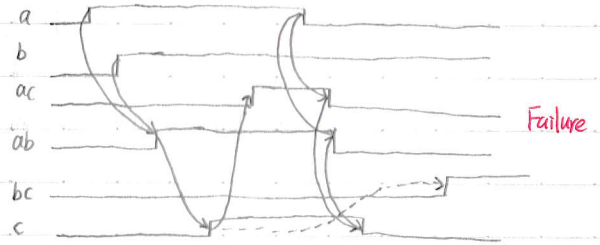
Taking Delay into account



Delay assumption:

d1 = d2 = 2 time units
d3 = d4 = 1 time units
d5 = 10 time units
Gate: 1 time unit

(The logic inside ↗)



Failure

This design function correctly for unbounded (complex) gate delay models
This design is incorrect for un-bounded wire delay models
For bounded delay model, correctness depends on delay constraints

Summary

This Synthesis method produces speed independent implementations that function correctly under the assumption of unbounded gate delay models

Wire delay are ignored, in real system this can be achieved by ~~de~~ divided the system smaller modules within which wire delay are ignored. These modules communicate using delay insensitive interfaces or interfaces satisfying certain timing constraints

It is your duties as a designer to make sure that these assumptions are correct

Asynchronous Design Tools

[1] Work craft http://www.workcraft.org/

Developed by Newcastle University

Used for synthesis of asynchronous circuits from Petri net and Transition graph specification

[2] The Balsa Asynchronous Synthesis System

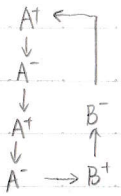
Developed over a number of years at the University of Manchester.

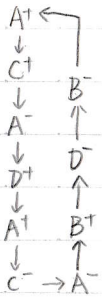
Balsa is built around the HandShake Circuits methodology and can generate gate level netlist from high-level descriptions in the Balsa language.

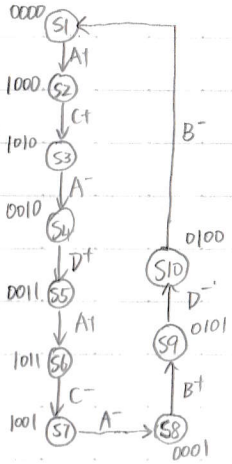
Both dual-rail (QDI) and signal-rail (bundled data) circuits can be generated

Latest release support Xilinx FPGA technology targets

http://apt.cs.manchester.ac.uk/projects/tools/balsa/







"[out]" indicate a feedback to input "out"

Inorder = A B C D

Outorder = [B] [C] [D]

[B] = A'C'D

[C] = A'C + AD'

[D] = B'D + A'C

B

AB \ CD	00	01	11	10
00	0	1	0	0
01	0	1		
11				
10	0	0	0	0

B = A'C'D

C

AB \ CD	00	01	11	10
00	0	0		1
01	0	0		
11				
10	1	0	0	1

✓C = A'C + AD'

C = A'B'C + AB'D'

D

AB \ CD	00	01	11	10
00	0	1	1	1
01	0	0		
11				
10	0	1	1	0

✓D = B'D + A'C

D = B'D + A'B'C

1000	1011
1010	1001
0010	0001
0011	0101
	0100
	0000

No.

Date

Metastability and Synchronizers

Describe the need for synchronization in modern systems

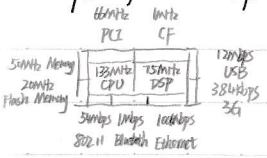
Why Synchronization : 1. Sampling Asynchronous Signals

Before an asynchronous signal can be input to a synchronous module, it must first be synchronized with the local clock to ensure that it makes transitions only during allowed portions of the clock cycle. Such synchronization is required, for example, when sampling real-world input devices whose output can change at any time asynchronously. If such signals that were input directly into a synchronous system, they might cause a synchronization failure.

Why Synchronization : 2. Crossing Clock Domains

Increasing complexity of modern SoC requires the use of many different IPs with different clocking requirements

Example of a smartphone :



① ∿20 IPs are getting usual
② Physical interface peripherials ∿ 200MHz
③ Modern DSP processing ∿ 400MHz
④ MAC processing ∿ 600MHz
⑤ GP Host ∿600MHz → 1GHz
⑥ Graphical & Video accelerators ∿ 500MHz
⑦ Interconnect ∿ 400MHz

The use of a single clock for such systems is becoming less feasible, therefore modern systems have multiple clock domains.

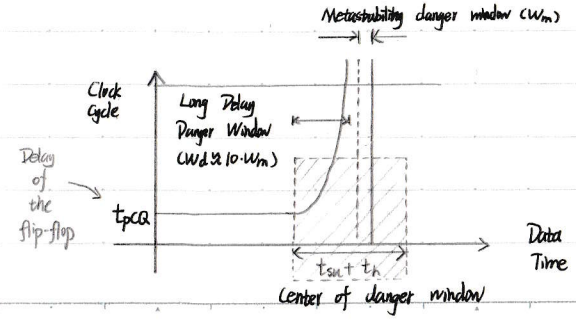
When transferring data between different clock domains, it must first be synchronized with the local clock of the destination to ensure that it makes transitions only during allowed portions of the clock cycle otherwise this may cause a synchronization failure

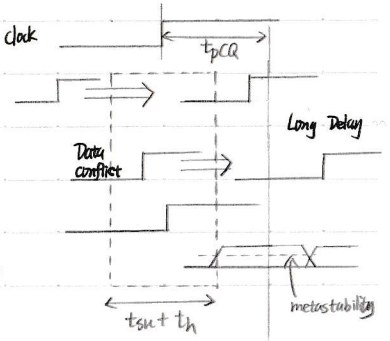
Describe causes and effects of synchronization failures.

Synchronization Failures

Timing violation of flip-flops or latches could cause a metastable output, in which the voltage level may be indetermine for an unbounded length of time before settling to a valid level.

Metastability is a condition where the voltage level of a signal is at an intermedia level (neither 0 or 1) and which may persist for an indetermine amount of time.





Long Delay / Metastability windows

Both Long Delay and Metastability ~~windows~~ are bad

We will consider the combined window in our calculation: $W = W_d + W_m$ (either long delay or metastability)

If data arrives during the danger window W than we will get a synchronous failure

W is Experimentally, less than 4 gate delays (FO4)

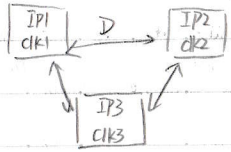
How often does metastability happen?

Assuming that: ① Data arrives or changes uniformly over clock cycle T
② W is the width danger zone

The probability of entering Metastability assuming a clock period is given by:

$$P = \frac{W}{T} = W \times F_C$$

Multiplying the probability of becoming metastable during a cycle that has a data event by the rate of data events yields the rate of entering into metastability (MR)



$$MR = W \times F_C \times F_D$$

[EXAMPLE]

Calculate the rate of Metastability that is caused by moving one data item ~~for~~ from IP1 to IP2, assuming the clock frequency of IP1 is 800MHz, the clock frequency of IP2 is 400MHz
The Data transfer rate is 200KHz
The width of the danger zone for all flip-flops in the system is 132ps 10^{-12}s

$F_C = 400MHz$ $F_D = 200kHz$ $W = 132ps$

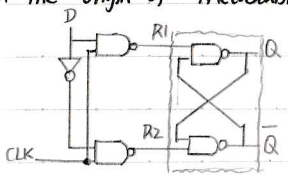
$MR = W \cdot F_C \cdot F_D = 400 \times 10^6 \times 200 \times 10^3 \times 132 \times 10^{-12} = 10560$

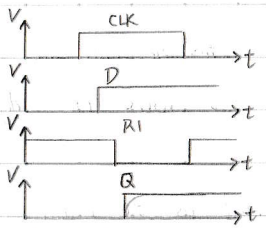
$800000 \times 132 \times 10^{-3}$

$132 \times 80 = 10560$

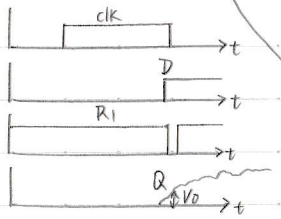
10,000 synchronization failure occur per second

The Origin of Metastability





Normal Operation of NAND Latch



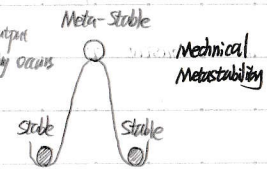
D changes at a very close time to the falling edge of the clock, so the output (Q) does not have sufficient time to make transition from 0 to 1. The SR Latch in this case will act as a cross coupled sense amplifier.

Assuming metastability happened at (t=0), the output voltage of (Q) at time t will be given as:

$$V(t) = V_0 \cdot e^{\frac{t}{\sigma}}$$

V_0: Value of the output when metastability occurs

What happens after metastability



Latch goes 1 or 0 randomly

Noise can have effect on the final output

Metastability does not usually last very long, but it can cause system failure

How long does the metastability last?

Metastability ends approximately when the output voltage of the latch reaches the threshold voltage of the NAND gate, by substituting V with Vth in the previous equation we get:

$$V_{th} = V_0 \cdot e^{\frac{t}{\sigma}}$$

Therefore Metastability will approximately lasts: $t_m = \sigma \cdot \ln \frac{V_{th}}{V_0}$

Probabilistic Analysis shows that given a metastability event at (t=0, the probability of metastability at t>0 (P_m) can be calculated as:

$$P_m(t) = e^{\frac{-t}{\sigma}}$$

Mean Time Between Failures (MTBF)

Failure means that flip-flop became metastable after the clock's sampling edge, and that it is still metastable S time later. The two events are independent, so we can multiply their probabilities.

The rate of failure (FR) is calculated as the product of the metastability rate and the probability of staying in metastable state for S time.

$FR = MR \cdot P_m(S)$ ← time between t_0

$FR = W \times F_c \times F_d \times e^{-\frac{S}{\tau}}$

$MTBF = \frac{1}{Rate\ (failure)} = \frac{e^{\frac{S}{\tau}}}{W \times F_c \times F_D}$

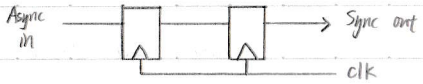
$\frac{e^{26}}{3 \times 10^9} \approx 10^2$ years

W = 4 FO4
J = 2 FO4
S = 100 FO4

* To design synchronous synchronizers for multiple clock domain SoCs

How to avoid Metastability

Allow more time for recovery : by forcing a full clock cycle between the two flip-flops

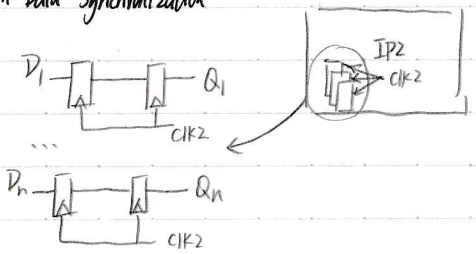


Use more than 2 flip-flops you can.

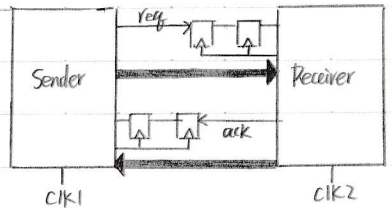
$MTBF = \frac{e^{\frac{S}{\tau}}}{W \times F_D \times F_c}$

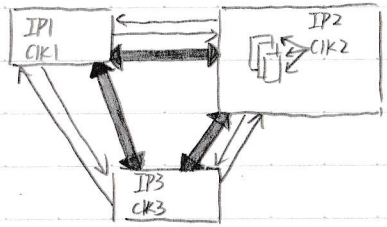
- S : predetermined time S for metastability resolution
- f_c, f_d : clock & data frequencies
- J, W : technology parameters (J ≈ 2 FO4 & W ≈ 4 FO4 delays)

Data Synchronization



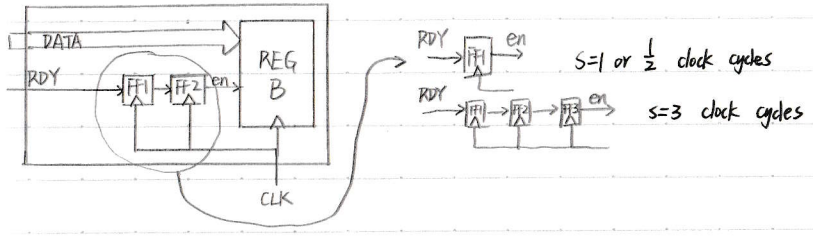
Control Synchronization





Control Synchronization : How to implement

S=2 clock cycles

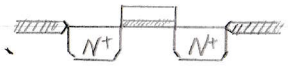


R	W
1400 M/s	180M/s
X2 380 M/s	X10 1900M/s

No.

Date

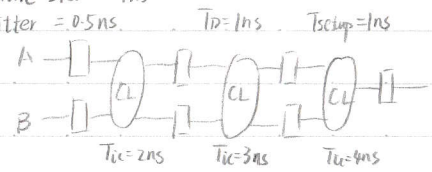
Twin-Tub



↑ NMOS ~~with~~ for N-well process

也是 NPN 三极管
以 P 为底，所以上面
的 PNP 有一个 N-well process
由于这个 NPN 没有 P-well，
所以不是 Twin-Tub
工艺为 N-well process

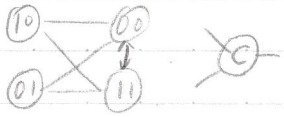
Positive Skew = 1ns
Jitter = 0.5ns



$T_{CLK} \geq T_{ic\text{-}max} + T_D + T_{setup} - Skew + 2\,Jitter = 6ns$

00	0
10	0
01	0
11	1

Acknowledge the change
From 0→1 (00,10,01→11)
1→0 (11→00,10,01)



L-Edit V15.24

15th Nov

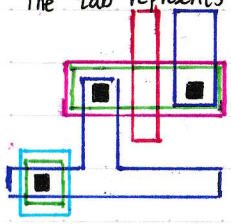
No.

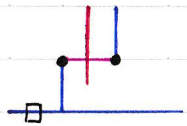
Date

Tabs ~~unFilled black square~~ ✓ □

not shown on stick diagram

The tab represents a connection to something we can't see. either the N-well or the wafer substrate



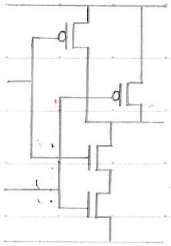


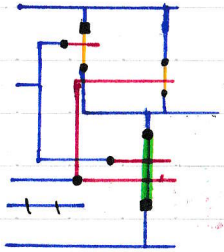
✓ ~~Filled Black Square~~ ■

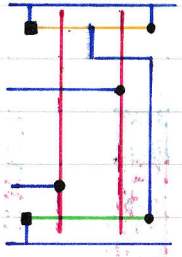
※ Combined contact & Tops

We can often save space by using a combined contact and tap.
Here the tap shares the same active region area as the contact.

[Stick Diagram]



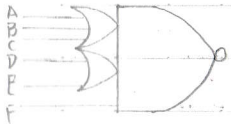


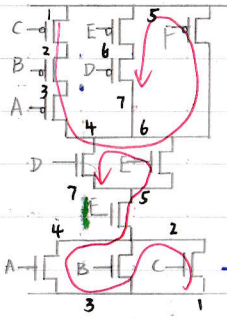


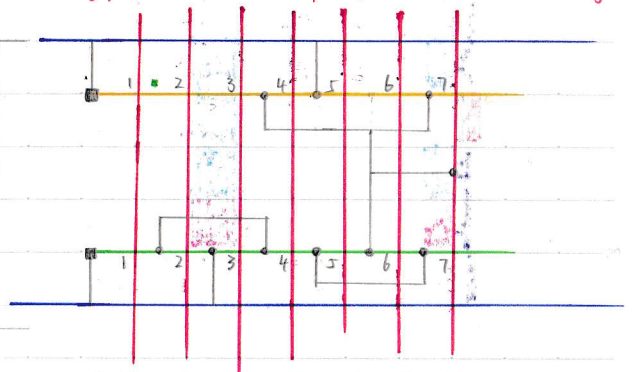
① Find Euler path
② Label poly columns
③ Route power nodes
④ Route output node
⑤ Route remaining ~~nodes~~ nodes
⑥ Add taps for PMOS and NMOS

[Euler Path] $Z = \overline{(A+B+C)\cdot(D+E)\cdot F}$

Vertices of odd degree
A graph will contain an Euler path if it contains at most two
A graph will contain an Euler path if all vertices have even degree

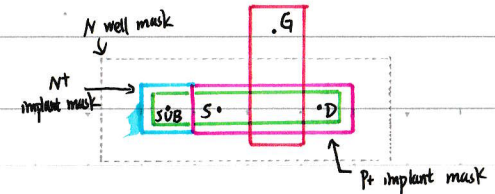






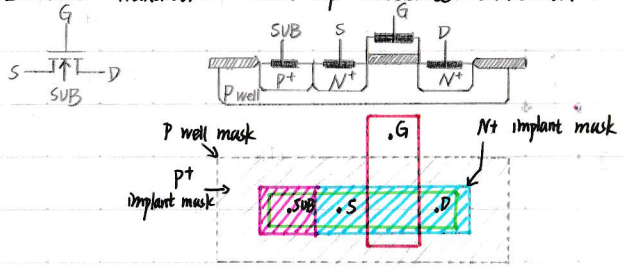
相连一个标留一个标记数字

※ A combined ~~or~~ contact and tap, ■, may be used only where a power contact exists at the end of a line diffusion. Where this is not the case a simple tap ⊟ should be used



CMOS

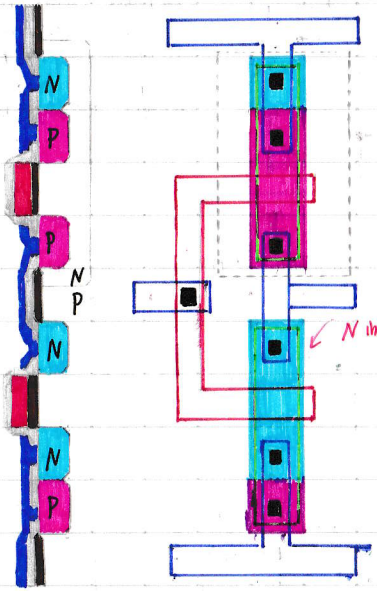
[NMOS Transistor] - with top substrate connection

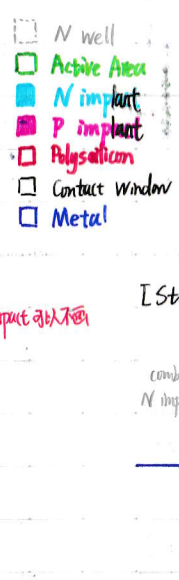


Five masks must be used to define the transistor
- P well (for isolation)
- Active Area
- Polysilicon
- N+ implant
- P+ implant

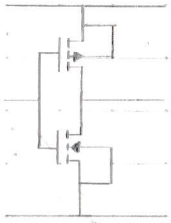
Top substrate connection
P+/N+ implants produce good ohmic contacts

[CMOS Inverter]

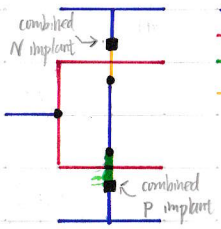




N well
Active Area
N implant
P implant
Polysilicon
Contact window
Metal

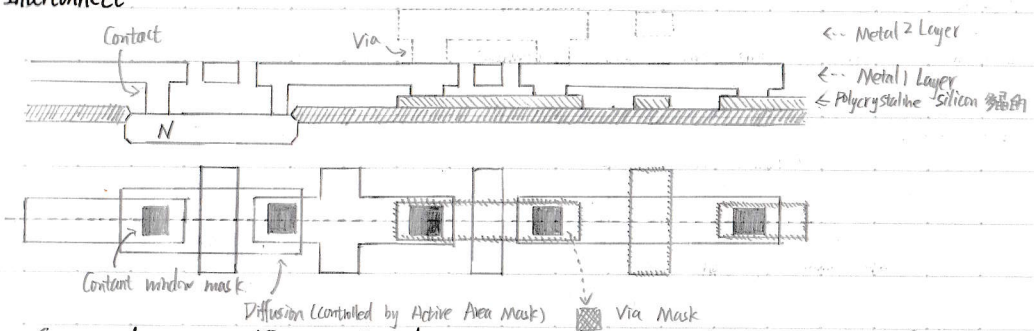


[Stick Diagram]



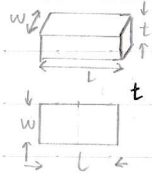
Metal
Polysilicon
N+
P+
Contact
Tap
Combined Contact & Tap

Interconnect



Crossing conductors on different masks do not interact → Explicit contact/via is required for connection
Crossing conductors on the same mask are always connected.

[Resistance]



$R = (\frac{\rho}{t})(\frac{L}{W})$ ρ is the resistivity constant { $3.2\times10^{-8}\,\Omega m$ for aluminium; $1.7\times10^{-8}\,\Omega m$ for copper

t and ρ are fixed for a particular mask layer

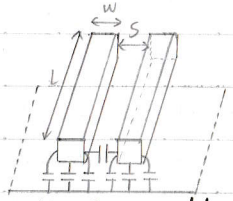
$R = R_s(\frac{L}{W})$ R_s is the sheet resistance ~~0.1Ω/□~~ for 1mm thick copper

(wrong)
Serpentine shape to save on area

~~Cont~~

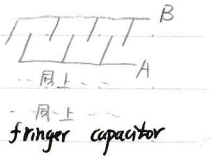
Corner squares count as half squares

[Capacitors]



parallel tracks over underlying conductor

metal-insulator-metal capacitor



Coupling capacitance to adjacent track

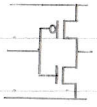
$C = C_a \cdot W \cdot L + 2C_f \cdot L$ $C = C_c \cdot \frac{1}{S}$

C_a, C_f, C_c are constants for a given layer and process
In digital designs our only aim is to minimise parasitic capacitance

No.

Date

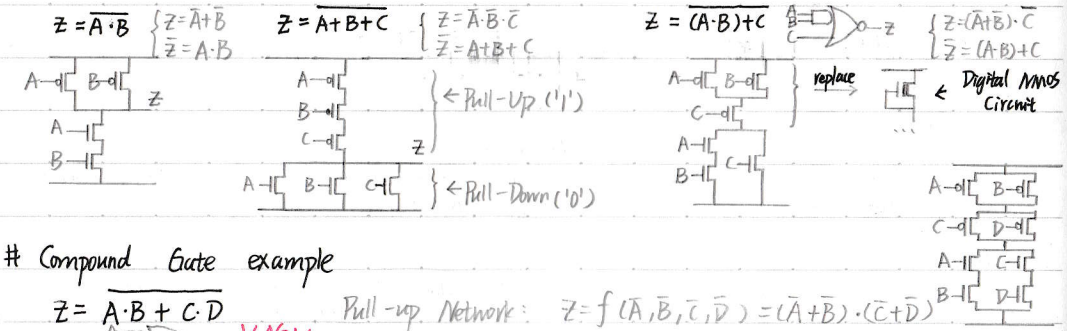
CMOS (Complementary Metal-Oxide-Semiconductor) ALL CMOS devices are enhancement mode



An active PMOS device complements the NMOS device giving
- rail to rail output swing
- negligible static power consumption

ALWAYS Inverting

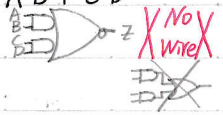
$Z = \overline{A \cdot B}$ { $Z = \bar{A} + \bar{B}$, $\bar{Z} = A \cdot B$ } $Z = \overline{A + B + C}$ { $Z = \bar{A} \cdot \bar{B} \cdot \bar{C}$, $\bar{Z} = A + B + C$ } $Z = \overline{(A \cdot B) + C}$ { $Z = (\bar{A} + \bar{B}) \cdot \bar{C}$, $\bar{Z} = (A \cdot B) + C$ }



Compound Gate example

$Z = \overline{A \cdot B + C \cdot D}$

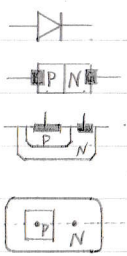
Pull-up Network: $Z = f(\bar{A}, \bar{B}, \bar{C}, \bar{D}) = (\bar{A} + \bar{B}) \cdot (\bar{C} + \bar{D})$

Symbol: 

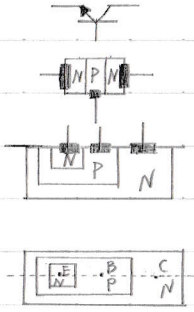
Pull-down Network: $\bar{Z} = f(A, B, C, D) = A \cdot B + C \cdot D$

Components for IC Design

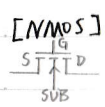
[Diode]



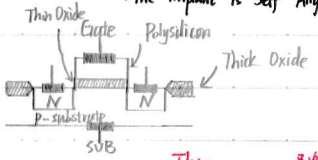
[NPN Transistor]

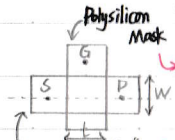


[NMOS]



N-type implant has no extra mask
→ It's blocked by thick oxide and by polysilicon
→ The implant is Self Aligned (자기)

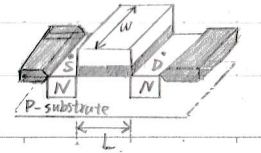




→ defines extent of ~~thick~~ Thin oxide (also Gate Oxide)

→ defines extent of Thick Oxide

Substrate connection is to bottom of wafer (웨이퍼)
- All substrates to ground



Campus

No.

Date

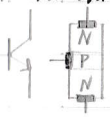
Overview of Techologies

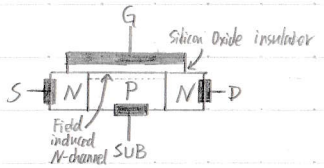
Moore's Law

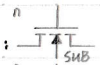
(1965-1975) → The number of components per chip had doubled every year
(1975-2017?) → Doubling every two years

Components for Logic

Diode: 

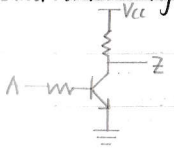
Bipolar Transistor: 

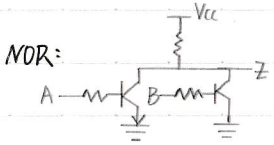


MOSFET:
- Enhancement:  ← off at zero gate-source voltage

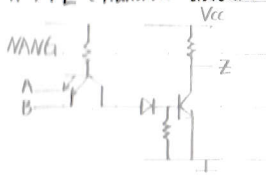
- Depletion:  ← the device is normally ON at zero gate-source voltage (for N-type, the threshold voltage might be -3V)

RTL (Resistor transistor logic)

Inverter: 

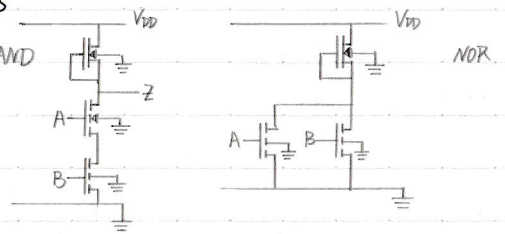
NOR: 

TTL (Transistor-transistor logic)



TTL gives faster switching than RTL at the expense of greater complexity.

NMOS



Depletion transistor acts as non-linear load resistor
↳ Resistance increases as the enhancement device turns on, thus reducing power consumption

KOKUYO