Chap 4 Arithmetic function

4.1 Iterative combinational circuits

Cell – sub-function block

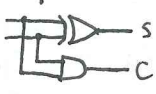
4.2 Binary adders

Half-Adder

X + Y = CS (C = carry, S = sum)

X	Y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

most common implementation: $S = X \oplus Y$, $C = XY$



NAND only: $S = (X \uparrow Y) \uparrow Z$, $C =$ 立马

Full-Adder

Z + X + Y = CS (Z = carry-in, C = carry, S = sum)

X	Y	Z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$S = X\bar{Y}\bar{Z} + \bar{X}Y\bar{Z} + \bar{X}\bar{Y}Z + XYZ$

$C = XY + XZ + YZ$

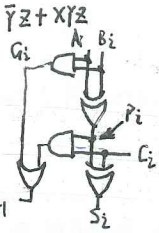
$\Downarrow$

$S = X \oplus Y \oplus Z$

$C = XY + (X \oplus Y)Z$

$X \cdot Y$ = carry generate

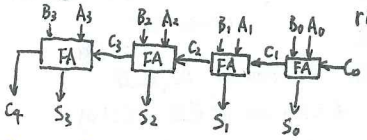
$X \oplus Y$ = carry propagate



Binary Adders – Ripple-Carry

$A(3:0) + B(3:0) = S(3:0)$

FA = full adder



Long path: A_0/B_0 到 S_3

读实现法 – 大问题便是贯通着bit数增加，从最低位向最高位传播 ripple carry 的时间也越长

Binary Adders – Carry Lookahead

当 A、B 都为 1 时，不管是否有在来自上一位的进位，必然会向下一位产生进位。

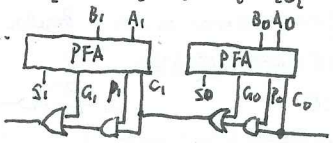
Generate，指由 A、B 产生的进位。 propagate，指由上位进位与 AB 进行计算产生的进位

$P_i = A_i \oplus B_i$ $G_i = A_i B_i$ $\Rightarrow$ $S_i = P_i \oplus C_i$ $C_{i+1} = G_i + P_i C_i$

I. $C_1 = G_0 + P_0 C_0$

II. $C_2 = G_1 + P_1 G_0 + P_1 P_0 C_0$

⋮



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

Example:

16-bit CLA

Delay: NOT = 1, XOR = Isoland AND = 2+1, AND-OR = 2

Longest Delays: Ripple carry adder = 3+(5×2+3 = 36

CLA = 3+3×2+3 = 12

4.3 Binary adder-subtractors

Subtraction – Unsigned subtraction M-N

if no end borrow occurs: then $M \geq N$

if an end borrow occurs: then $N > M$. $2^n - (M-N) = -(N-M)$

a minus sign is appended to the result.

Numerical Coding

相减是通过加减数的补实现的。

例：$N = 0111\,0011_2$ 的 1's complement 为

$$\begin{array}{r} 1111\ 1111 \\ -\ 0111\ 0011 \\ \hline 1000\ 1100 \end{array}$$

$2^n - 1 + X \quad X < 0$

因为 1-1=0 1-0=1，所以 1's complement 是用按位 not 做到的 (bitwise not)

Binary 2's Complement

$$\begin{array}{r} 1\,0000\ 0000 \\ -\ 0111\ 0011 \\ \hline 1001\ 1101 \end{array}$$

$2^n + X \quad X < 0$

是 1's complement 的结果加 1。 (也是取反加 1 的来源)

另一种 2's complement 的实现法：

10010 100 → 01101 + 100

步骤 1：copy 右边的 0 和最低位的 1

步骤 2：对左边其余位取反

Signed Integers

$S\ a_{n-2} \cdots a_2\ a_1\ a_0$

↑ 0 为正 1 为负

Number	Sign-Mag.	1's Comp.	2's Comp.
+3	011	011	011
+2	010	010	010
+1	001	001	001
+0	000	000	000
-0	100	111	—
-1	101	110	111
-2	110	101	110
-3	111	100	101
-4	—	—	100

Adder – Subtractor (with 2's Complement)

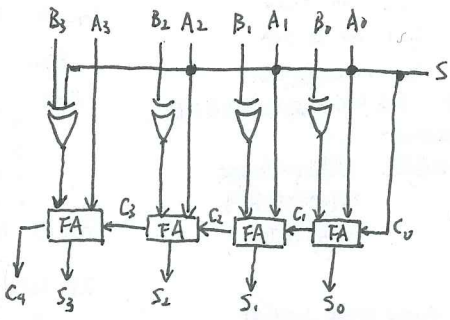
$$\begin{array}{r} 01010100 \\ -\ 01000011 \end{array} \xrightarrow{2\text{'s comp}} \begin{array}{r} 01010100 \\ -\ 10111101 \\ \hline 00010001 \end{array}$$

$$\begin{array}{r} 0100\ 0011 \\ -0101\ 0100 \end{array} \xrightarrow{2\text{'s comp}} \begin{array}{r} 0\,0100\ 0011 \\ +\ 1010\ 1100 \\ \hline 1110\ 1111 \end{array}$$

2's comp → 0001 0001

1 carry 表明无需对结果修正

0 carry 表明需要对结果修正



S=1 时就是减。

2's implement 是通过 XORs 与 $C_0 = 1$ 实现

S=0 时就是加

B 原样不动通过

Overflow 检测：$V = C_n \oplus C_{n-1}$

4.4 Other arithmetic function

压缩 (contraction)：将已有电路简化成一个简单电路。

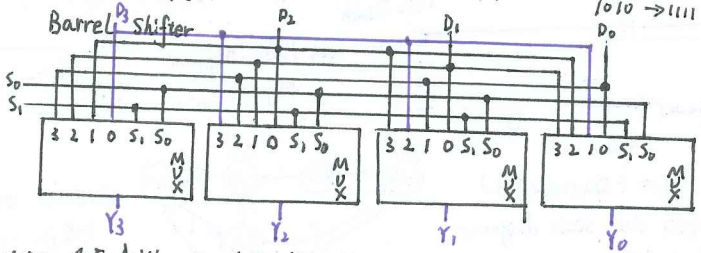
Incrementing 向一个数加上固定值

Decrementing 一个数减去固定值

Zero fill 补 0 至预期位数

Extension 0110 → 0000 0110; 1010 → 1111 1010

Barrel Shifter 循环左移



S=00 不动

S=01 左移一位

S=10 左移2位

S=11 左移3位

4.5 Arithmetic Logic Unit (ALU)

ALU 三大部分：① an arithmetic circuit ② A logic circuit ③ selector

- an arithmetic circuit
  - n bit 一行加法器
  - 选择四种一般的 input
  - B、$\bar{B}$、1、0
- A logic circuit
  - AND
  - OR
  - XOR
  - NOT
- selector 选择是算数运算结果还是逻辑运算结果

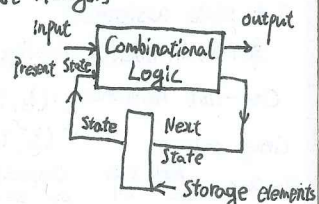
Chap 5 Storage Elements and Sequential Circuit Analysis

Next state function: Next State = f(Inputs, State)

Output function (Mealy): Outputs = g(Inputs, State)

Output function (Moore): Outputs = h(state)

Types of Sequential Circuits: Synchronous / Asynchronous

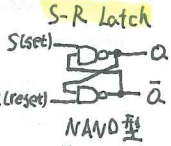


5.1 Storage element

Latch

满足下列条件：① 可长久保存一给定状态 ② 状态为 0、1 其中一种 ③ 在特定状态下可随时改变状态

S-R Latch



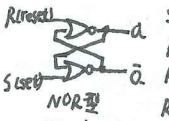
NAND 型 $\bar{S}\bar{R}$ Latch

S=0, R=0 is forbidden

R=1 S=0 将设置 Q 为 1

R=0 S=1 将设置 Q 为 0

R=1 S=1 将保存状态



NOR 型 SR Latch

S=1, R=1 is forbidden

R=0, S=0 保存状态

R=1, S=0 将设置 Q 为 0

R=0, S=1 将设置 Q 为 1

Clocked Latch

D-Latch

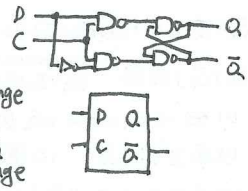
Q	D	Q(t+1)
0	0	0
0	1	1
1	0	0
1	1	1

No change

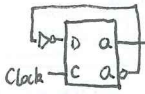
Set Q

Clear Q

no change



The Latch Timing Problem

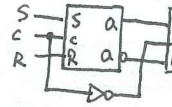


如图：clock Y

只要时钟一直为高，Y就会一直震荡！

解决法为不要组成一个在一个 storage 部件内从为到为的闭环

方案 1：S-R 主从 Flip-Flop



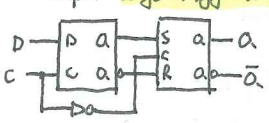
存在的问题："1s catching" 问题

input 在 C=1 时会被第1个 Latch 观察到 output 则在 C=0 时改变。

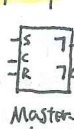
R 一直为 0 那么 1 会传给从 Latch

如果 S 先高后低 R 紧跟着更高后低 那么 0 会传给从 Latch

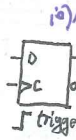
方案 2：Edge-Triggered D Flip-Flop



Negative-edge triggered flip-flop (上边沿触发时钟信号起始接一个反向器)



Master-slave SR



Edge-Triggered D Flip-Flop