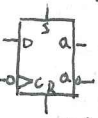
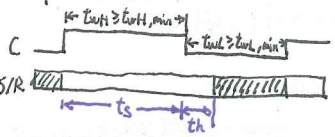
Direct input



可以用 S 设定一个初始值.
用 R 重设为 0



5.2 Sequential circuit analysis

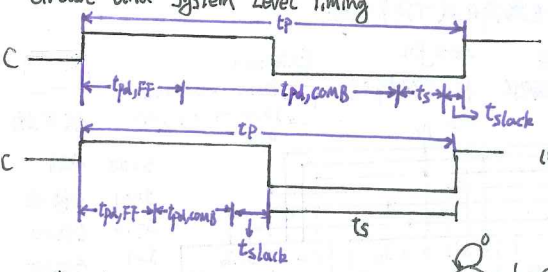
State table characteristics

① Present State ③ next-state
② Input ④ output

FSM 两大分类:

Moore: Output 只与 state 有关 Mearly: Output 与 state、input 有关

Circuit and System Level Timing



(a) Edge-triggered (positive edge)

(b) Pulse-triggered (negative edge)

$t_p = t_{slack} + (t_{pd,FF} + t_{pd,comB} + t_s)$

5.3 state analysis and state diagrams

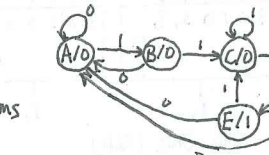
为建立序列识别状态图

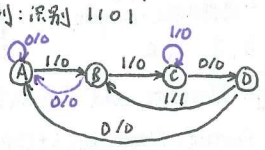
① 画出起始态, 表明从 Reset 后没有任何序列输入 Moore Model

② 添加一个状态, 表明识别了第一个符号 ③后续状态依次表明识别后续符号

④ 最终状态代表输入最终符号.

例: 识别 1101





同时还要添加状态处理非法输入.

5.4 State assignment

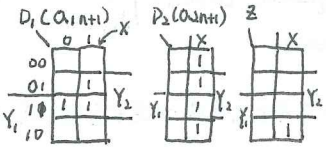
最小 bit 用于分配状态 $n \geq \lceil \log_2 m \rceil$ $2^n - m$ 个无用状态 Mearly Model

One-Hot Assignment: $(Y_3, Y_2, Y_1, Y_0) = 0001, 0010, 0100, 1000$

or $(Y_3, Y_2, Y_1, Y_0) = 1000, 0100, 0010, 0001$

Gray code

$Q_m Q_n$	Next state x=0	Next state x=1	Output(z) x=0	Output(z) x=1
0 0	0 0	0 1	0	0
0 1	0 0	1 1	0	0
1 1	1 0	1 1	0	0
1 0	0 0	0 1	0	1



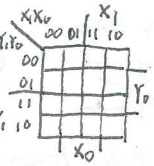
$\therefore D_1 = Y_1Y_2 + XY_2 \quad D_2 = X \quad z = \bar{Y}_1 Y_2 X \quad Y_1, Y_2$: D flip-flop output

$D_D = \bar{Q}_A Q_D + \bar{Q}_B Q_D + \bar{Q}_C Q_D + Q_A Q_B Q_C \bar{Q}_D$ 状态数 m D Flip-flop 数 $n \geq \lceil \log_2 m \rceil$

$= (\bar{Q}_A + \bar{Q}_B + \bar{Q}_C) Q_D + \overline{Q_A Q_B Q_C} \bar{Q}_D$

$= (\bar{Q}_A + \bar{Q}_B + \bar{Q}_C) Q_D + \overline{(\bar{Q}_A + \bar{Q}_B + \bar{Q}_C)} \bar{Q}_D \quad (A \cdot \bar{B} + \bar{A} \cdot B)$

$= (\bar{Q}_A + \bar{Q}_B + \bar{Q}_C) \oplus \bar{Q}_D = \overline{(\bar{Q}_A + \bar{Q}_B + \bar{Q}_C) \oplus \bar{Q}_D} = \overline{(\bar{Q}_A + \bar{Q}_B + \bar{Q}_C \oplus Q_D)}$



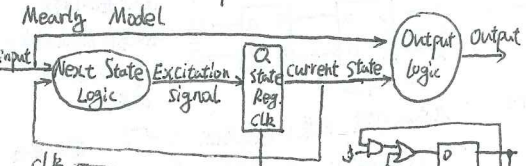
设计思路: (1) 给 input、output 以及状态分配 bit 用于表示

(2) 画出状态图 (3) 根据图画出状态表

(3) 画出 K-map, 化简表达式 (4) 根据表达式画出电路图

5.5 State machine implementation

Mearly Model



Moore 与 Mearly 差别: 就在最上面的 input 输入不存在

JK T flip-flop

5.6 Other Flip-Flop Types

JK 触发器

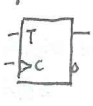
行为与 SR 主从触发器相同, 都有 "Is Catching" 的问题.

不同的是允许 J=K=1. 此时 Flip-flop 会切换到相反态

行为与 J=K 的 JK flip-flop 相同.

T=0 状态不变

T=1 状态相反



Flip-Flop Timing Parameters

t_s - setup time

t_h - hold time

t_w - clock pulse width

t_{px} - propagation delay

t_{PHL} High-to-Low

t_{PLH} Low-to-High

t_{pd} max(t_{PHL}, t_{PLH})

t_p - clock period 两次上下边沿间隔

$t_{pd,COMB}$ -

=> 状态图. 如果 input/output sequence 均相同 则两状态相同

Descriptors

D Flip-Flop:

Equation: $Q(t+1) = D$

Table: D	Q(t+1)	Operation
0	0	reset
1	1	set

T Flip-flop:

Equation: $Q(t+1) = T \oplus Q$

Table T	Q(t+1)	Op
0	Q(t)	No change
1	$\bar{Q}(t)$	Complement

S-R Flip-Flop

Equation: $Q(t+1) = S + \bar{R}Q$, $S \cdot R = 0$

table S	R	Q(t+1)	OP
0	0	Q(t)	No change
0	1	0	Reset
1	0	1	Set
1	1	?	Undefine

J-K Flip-flop

Equation: $Q(t+1) = J\bar{Q} + KQ$

table J	K	Q(t+1)	OP
0	0	Q(t)	No change
0	1	0	reset
1	0	1	set
1	1	$\bar{Q}(t)$	complement

Chap 7 Registers, Microoperations & Implementations

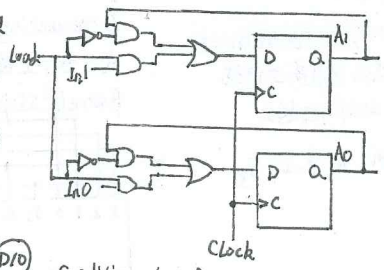
7.1 What is Registers

Register: 常被认为储存了一个 binary value 向量.

Operating: 数据存储、移动, 执行操作

要求: (1) 能多个时间 cycle 存储数据

(2) Load 操作应该由一个 Load 信号控制.



Load 0: 维持数据不变 Load 1: 读取并储存新数据

Elementary Operations: load, count, shift, add, bitwise "OR"

Register 记号 微操作

① 字符、数字: 表示为 Register R2、PC、IR

② 括号 ( ): 表示 Register bit 范围 PC(7:0)

③ 箭头 ←: 表明数据转输 $R1 \leftarrow R_2$

④ 方括号 []: 表明 memory 地址 $R_0 \leftarrow M[PC]$

Condition transfer

如果 (K1=1) 时, 执行 ($R_2 \leftarrow R_1$). 简写: K1: ($R_2 \leftarrow R_1$)

Arithmetic operations: + - * /

Logical operations: ∨ Logical OR ∧ Logical AND ⊕ Logical Exclusive OR − Not

Microoperations

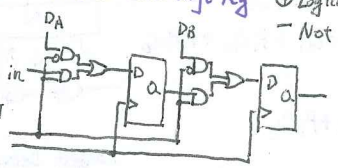
① Transfer: 从一个 register 将运数据到另一个

② Arithmetic: perform arithmetic on data

③ Logic: 操纵数据或用 bit 级的逻辑操作

④ Shift: 左/右移数据

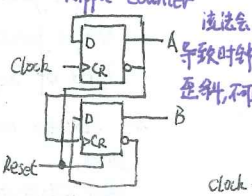
Parallel Load Shift Reg



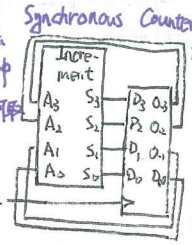
7.2 Microoperations arithmetic & logic SHIFT 寄存器

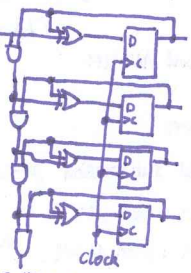
7.3 Microoperations on a single register-2 CP

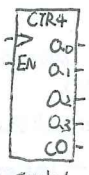
Ripple Counter 连续会导致时钟歪斜, 不可靠



Synchronous Counter







Symbol

还可以加入 Parallel load

Other Counter

Down Counter 向下计时

Up-Down Counter 根据 up/down 信号 决定正数时还是倒计时

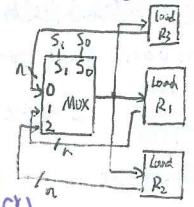
Parallel Load Counter 可以读入一特定值后开始计数

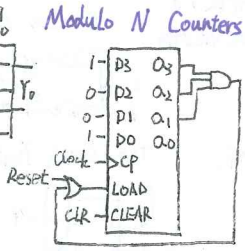
BCD Counter 能使计数器在 2 个 Clock Cycle 后回归正常数据.

Modulo N Counters

不要使用计数器结果作为 clear 的输入

Asynchronously clear 又称 suicide counter





从 9 计数到 14 的计数器

9, 10, 11, 12, 13, 14, 9, 10 …

7.4 Register Cell design

一个 Register Cell 有:

① 一个 Register

② 作为输入的 data input 线 B (CX, CY) (0,0) (0,1) (1,0)

③ 控制 input 的组合逻辑 CX: A ← B ∨ A (0,0)

④ 一组 Register 函数

⑤ 保持状态描述

Three State Bus