多位加法器(CPAs)

进位传播加法器(Carry Propagate Adder, CPA)

行波进位加法器(慢速)　　书本第5章(A)

先行进位加法器(快速)

前缀加法器(更快速)

N/k块 { k个行波进位(串行)
TPG + TPGblock + (N/k-1)tAND-OR + ktFA

减法器

[A-B]补 = [A+(-B)]补 = [A]补 + [-B]补 = [A]补 + $\overline{[B]补}$ + 1

assign {Cout, S} = Sub ? (a + ~b + 1) : (a + b);

比较器

算术逻辑单元(ALU) 第6周. 实验一：多数表决器

ALU 受F(操作码)控制

F2:0	000	001	010	011	100	101	110	111
	A&B	A|B	A+B	-	A&~B	A|~B	A-B	SLT

SLT(小于则置位)操作，当A<B时，Y=1，否则Y=0

F2 决定B与~B

移位器和循环移位器

逻辑移位、算术移位

循环移位器

11001 ROR 2 = 01110

11001 ROL 2 = 00111

可以用复用器(MUX)实现

乘法器

无符号二进制数乘法 可以通过移位与加法实现

有符号乘法 处理符号 (提取符号位，作异或运算)

加法器溢出条件

1. 正数+正数 结果为负数
2. 负数+负数 结果为正数

FPGA 半定制电路

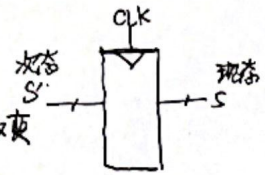
有限状态机（同步时序电路的设计）

组成：

状态寄存器：

存储当前时刻状态

状态在下一个有效时钟沿发生改变



组合逻辑

计算下一状态（次态方程）

计算电路输出

Moore Mealy 型有限状态机

Moore 输出 $Z=F(m_1, m_2, m_3 \cdots m_k)$

m_i 触发器状态

Mealy型 触发器与输入共同决定

例：友通灯控制器设计

输入：两个友通传感器 T_A，T_B

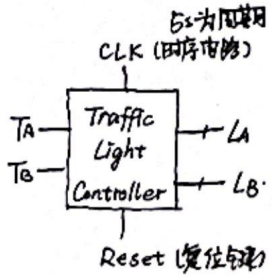
路上有人出现时，传感器返回 TRUE

输出：两个友通灯 L_A 和 L_B

红色，绿色加黄色

时钟周期为 5s

复位按钮



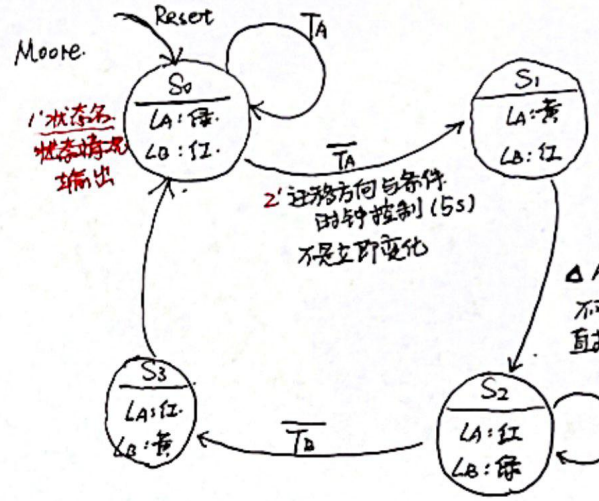
输入：CLK，Reset，T_A，T_B

输出：L_A，L_B

（有限状态机黑盒）

S_0：状态名称

Moore



1' 状态名、状态编码与输出

2' 迁移方向与条件，时钟控制（5s）

不是立即变化

Mealy 输入/下一个输出

△ A变黄以后，不可逆，直接变红（下一个周期）

状态转换表

触发器 Q_1	S_1	Q_0	S_0	现态 S	T_A	T_B	次态 S'	Q_1^+	Q_0^+
0		0		S_0	0	X	S_1	0	1
0		0		S_0	1	X	S_0	0	0
0		1		S_1	X	X	S_2	1	0
1		0		S_2	X	1	S_2	1	0
1		0		S_2	X	0	S_3	1	1
1		1		S_3	X	X	S_0	0	0

状态	编码
S_0	00
S_1	01
S_2	10
S_3	11

最优方案可简化电路

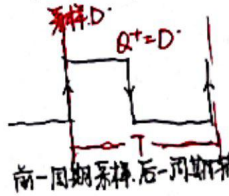
[多种方案]　次优最佳编码

状态编码（任意指出）

求次态方程

采样 D

$Q^+ = D$



前一周期采样，后一周期输出

$Q^+ = D = Delay$

求出 $Q_1^+ Q_0^+$ ← 画卡诺图

Q_1Q_0 \ T_AT_B	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

$$Q_1^+ = \overline{T_A} T_B + T_A \overline{T_B}$$

Q_1Q_0 \ T_AT_B	00	01	11	10
00	1	1	0	0
01	0	0	0	0
11	0	0	0	0
10	1	0	0	1

$$Q_0^+ = \overline{T_A} Q_1 \overline{Q_0} + Q_1 \overline{Q_0}\, \overline{T_B}$$

输出表

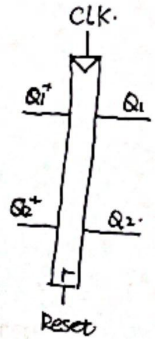
输出	编码
绿	00
黄	01
红	10

现态		输出			
Q_1	Q_0	L_{A1}	L_{A0}	L_{B1}	L_{B0}
0	0	0	0	1	0
0	1	0	1	1	0
1	0	1	0	0	0
1	1	1	0	0	1

$L_{A1} = S_1$　　$L_{A0} = \overline{S_1} S_0$

$L_{B1} = \overline{S_1}$　　$L_{B0} = S_1 S_0$

最后输出电路



Moore型有限状态机设计方法

① 根据问题对象，确定输入输出以及对应的逻辑含义

② 画出状态转换图

③ 列出状态转换表

④ 对状态进行编码，并列出次态方程

⑤ 列出输出表

⑥ 对输出进行编码

⑦ 绘制原理图

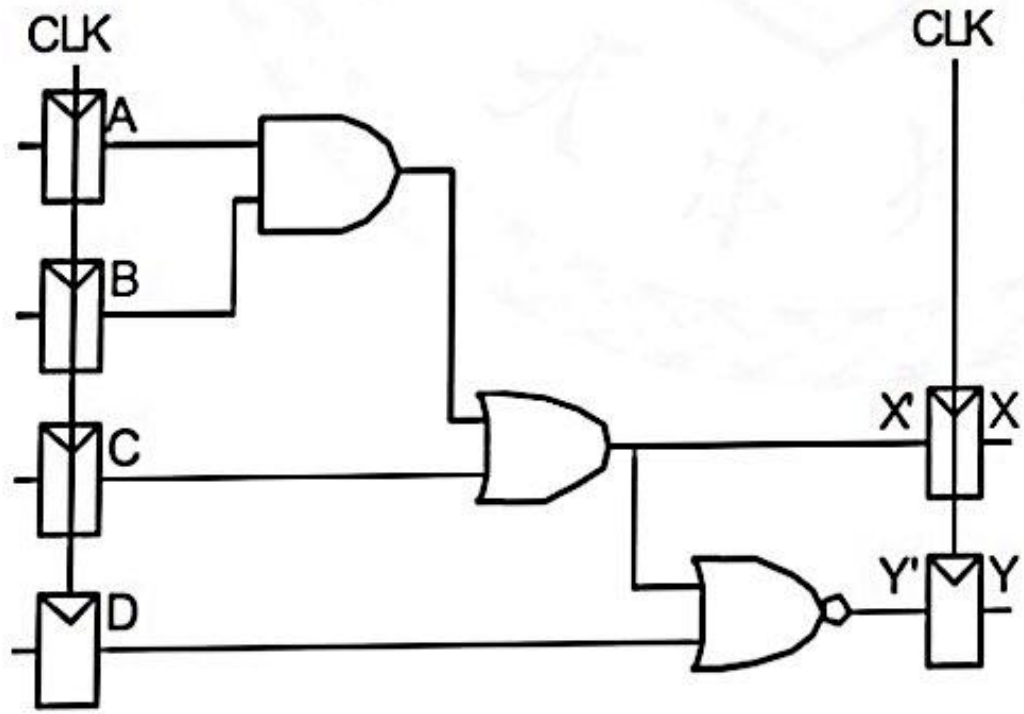
状态编码
- 二进制编码　01　10　11
- 独热编码　0001　0010　0100　1000

系统时序

System Timing

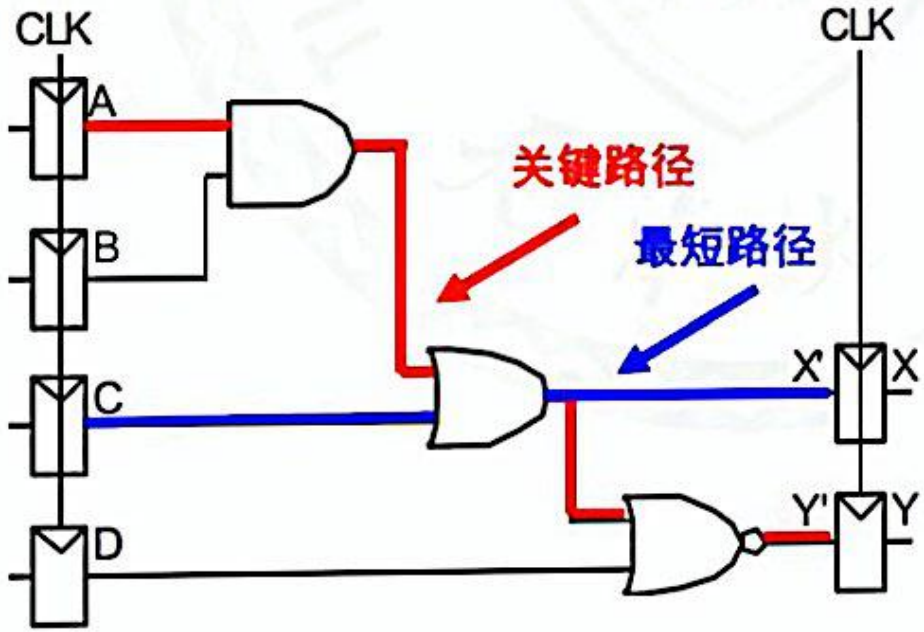
时序分析

- 在右图中
  - 触发器的最小延迟$t_{ccq} = 30ps$
  - 触发器的传播延迟$t_{pcq} = 50ps$
  - 触发器的建立时间$t_{setup} = 60ps$
  - 触发器的保持时间$t_{hold} = 70ps$
  - 逻辑门的最小延迟$t_{cd_gate} = 25ps$
  - 逻辑门的传播延迟$t_{pd_gate} = 35ps$
- 求最短时钟周期，并确定是否满足保持时间约束



时序分析（cont.）

- 已知：
  - $t_{ccq} = 30ps$，$t_{pcq} = 50ps$
  - $t_{setup} = 60ps$，$t_{hold} = 70ps$
  - $t_{cd\_gate} = 25ps$，$t_{pd\_gate} = 35ps$
- 解：
  - $t_{pd} = 35ps \times 3 = 105ps$
  - $t_{cd} = 25ps$
  - $T_c \geq t_{pcq} + t_{pd} + t_{setup} = 50 + 105 + 60 = 215ps$
  - $t_{cd} = 25ps$，$t_{hold} - t_{ccq} = 70 - 30 = 40ps$



$t_{cd} < t_{hold} - t_{ccq}$ 违反保持时间约束

t_{cd_gate} t_{pd_gate}

组合逻辑电路传输延迟和最小延迟

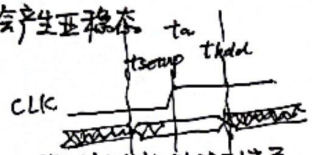
t_{cd} = 最短路径门电路个数 × t_{cd_gate}

t_{pd} = 关键路径门电路个数 × t_{pd_gate}.

时序问题

D触发器在时钟的有效边沿对D采样，并赋值给Q。

采样时，D要稳定，若不稳定，则会产生亚稳态。



输入时序

建立时间 (Setup time)：t_{setup} = 在时钟有效边沿到来前，输入信号需要稳定的时间

保持时间 (Hold time)：t_{hold} = 在时钟 —— 后，输入信号需要保持稳定的时间

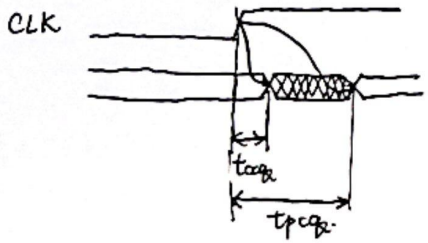
孔径时间 (Aperture time)：t_a = 在时钟边沿附近，保持稳定的时间

$$t_a = t_{setup} + t_{hold}$$

输出时序

传播延迟：t_{pcq} = 时钟有效沿到达后到数据稳定的最长时间

最小延迟：t_{ccq} = 时钟有效边沿到达Q开始改变最短时间



动态约束

同步时序电路中，输入必须在时钟有效边沿附近的孔径内保持稳定。

t_{setup} 与 t_{hold}

在时钟有效沿到达前，至少稳定 t_{setup}。

在时钟有效边沿到达后，至少稳定 t_{hold}。

系统时序

时钟周期 T_c 是两个相邻的上升沿（下降沿）之间的间隔。

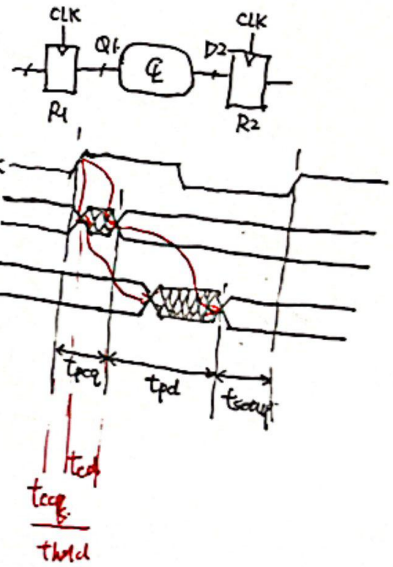
$f_c = 1/T_c$ 表示时钟频率。

建立时间约束 t_{setup}

建立时间约束由路径 R1 至 R2 间的最大延迟决定

寄存器的传播延迟 t_{pcq}

组合逻辑电路传播延迟 t_{pd}



寄存器R2的值必须在下一个时钟有效沿到来之前稳定

$$T_c \geq t_{pcq} + t_{pd} + t_{setup}$$

$$t_{pd} \leq T_c - (t_{pcq} + t_{setup})$$

保持时间约束 t_{hold}

$$t_{hold} \leq t_{ccq} + t_{cd}$$

$$t_{cd} \geq t_{hold} - t_{ccq}$$

在实际设计中，各寄存器的供应商将 t_{hold} 设计成 0，以保证保证保持时间约束在各种情况下都可以满足

教材中如非特别重要，注明，后面会忽略保持时间约束。

亚稳态产生的原因

D在 t_a（孔径时间）内不断变化，导致触发器输出无法确定。

$t_{cd\text{-}gate}$ $t_{pd\text{-}gate}$

组合逻辑电路传输延迟和最小延迟

t_{cd} = 最短路径门电路个数 × $t_{cd\text{-}gate}$

t_{pd} = 关键路径门电路个数 × $t_{pd\text{-}gate}$

~~ala~~ always过程块分为3种类型

always_comb (描述组合逻辑), always_latch, always_ff (后两者用于描述时序逻辑).

always @(sensitivity list). → 敏感事件列表，当列表中的事件产生时，过程块中语句开始工作

    Statement;

寄存器建模

```
module flop ( input logic clk,
              input logic[3:0] d,
              output logic [3:0] q );
always_ff @(posedge clk)
    q<=d;
endmoudle
```

表示触发器　表示clk信号上升沿　非阻塞赋值.

带复位端的寄存器建模

```
module flopr (input logic clk,
              input logic reset,
              input logic[3:0] d,
              output logic[3:0] q);
always_ff @(posedge clk)  当时钟同期上升沿到来.
    if (reset) q<=4'b0;  如果reset为1. 则q为4'b0.
    else q<=d;
endmodule
```

同步复位.(复位必须在时钟周期上升沿)

带使能端的寄存器建模

```
module flopren (input logic clk,
                input logic reset,
                input logic en,
                input logic[3:0] d,
                output logic [3:0] q);
always_ff @(posedge clk, posedge reset)
    if (reset) q<=4'b0;
    else if(en) q<=d;
endmodule.
```

```
module flopr (input logic clk,
              input logic reset,
              input logic[3:0] d,
              output logic[3:0] q);
always_ff @(posedge clk, posedge reset)
    if(reset) q<=4'b0;
    else q<=d;
endmodule
```

异步复位

当时钟同期上升沿到达 或 reset上升沿(由0变为1)时触发.

也就是复位在任意时刻进行

锁存器

```
module latch (input logic clk,
              input logic[3:0] d,
              output logic[3:0] q);
always_latch
    if(clk) q<=d;
endmodule
```