《数字电路设计》期末串讲

Digital Circuit Design

李晨阳

6 月 11 日

目录

1 Number System **3**
1.1 Conversion . . . 3
1.2 Arithmetic . . . 4
1.3 Negative Number . . . 4
1.3.1 Conversion: Binary –2' s Complement . . . 5
1.4 Floating Point Numbers . . . 5
1.5 Parity Checking . . . 6

2 Logic Function and Switching Algebra **6**
2.1 Some Gates . . . 6
2.2 Theorems . . . 7

3 Karnaugh Maps **8**
3.1 Minterm and Maxterm . . . 8
3.2 Karnaugh Map . . . 9
3.2.1 Prime Implicants . . . 11

4 Hazards and Glitches **12**

5 Latches and Flip-Flop **13**
5.1 SR Latch(NOR gate) . . . 13
5.2 $\overline{S}\,\overline{R}$ Latch . . . 14
5.3 D Latch . . . 15
5.4 Master-Slave D Flip-Flop . . . 15
5.5 Master-Slave JK Flip-Flop . . . 16
5.6 Edge-Triggered Flip-Flops . . . 16
5.7 Timing parameters . . . 17

6 Sequential Circuit Analysis **17**
6.1 Structure . . . 17
6.2 3 equations and 3 tables . . . 18

6.3 Process 19

7 Autonomous Sequential Circuit Design **20**

7.1 Process 21

1 Number System

1.1 Conversion

- **Hexadecimal (base 16):** Digits in base 16 range from 0 to 15: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F .
- **MSB:**Most Significant Digit.(rightmost)

  LSB: Least Significant Digit.(leftmost)

$$53_{10} = 110101_2$$

- A large binary number to be converted to decimal.**(Binary number-Hex-Decimal)**.
- Non-integer conversion: **Integer part:** from LSD to MSD. **Fractional Part:** from MSD to LSD.

Exercise 1:

Convert 46 to binary.

Convert 265.7810 to hexadecimal.

Convert 345_{16} to binary.

Convert 111001001010_2 to Decimal.

1.2 Arithmetic

1. Carry to next place when you exceed base!

2. Borrow the base amount when you need to subtract!

EXAMPLE:

A 7 3 4 8 6 5

+ 1 2 5 9 D 6

A 8 5 A 2 3 B

1.3 Negative Number

- **Sign-Magnitude Representation:** Magnitude and symbol representing +/ −.
- **Two' s Complement:**n bits create a range over $[-2^n - 1 \ldots 2^n - 1 - 1]$.

Code	Simple	Signed	2's Comp
0000	0	+0	0
0001	1	1	1
0010	2	2	2
0011	3	3	3
0100	4	4	4
0101	5	5	5
0110	6	6	6
0111	7	7	7
1000	8	-0	-8
1001	9	-1	-7
1010	10	-2	-6
1011	11	-3	-5
1100	12	-4	-4
1101	13	-5	-3
1110	14	-6	-2
1111	15	-7	-1

1.3.1 Conversion: Binary –2' s Complement

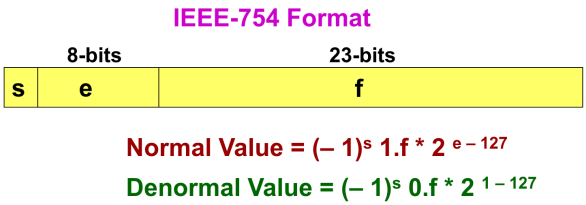
1. Obtain the n-bit simple binary equivalent.
2. Invert the bits of that representation.
3. Add 1 to the result.
4. Copy the sign bit into all the additional bits in the new format if you need to make extension.

Exercise 2:

Convert -46 to 8-bit 2' s complement.

Calculate 28-32.(in 2' s Complement).

1.4 Floating Point Numbers



Exercise 3:

Represent –0.75 in floating point format.

1.5 Parity Checking

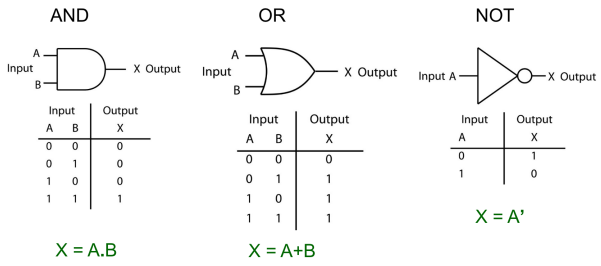
- Even-parity code: set the parity bit to 0 if there' s an even number of 1s; set the parity bit to 1 otherwise.
- Odd-parity code: set the parity bit to 0 if there' s an odd number of 1s; set the parity bit to 1 otherwise.

2 Logic Function and Switching Algebra

Switching Algebra:

- Two-valued boolean algebra.
- Basic operations are AND, OR and complement.

2.1 Some Gates



NAND

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

F = (A.B)'

NOR

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

F = (A+B)'

XOR

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

F = (A⊕B)

F = A'B+AB'

XNOR

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

F = (A⊕B)'

F = A'B'+AB

2.2 Theorems

(T1) X+0 = X	**(T1') X·1 = X**
(T2) X+1 = 1	**(T2') X·0 = 0**
(T3) X+X = X	**(T3') X·X = X**
(T4) (X')' = X	
(T5) X+X' = 1	**(T5') X·X' = 0**

De Morgan' s Theorems :Interchanging operations and complement each variable.

$(T13)(X1 \cdot X2 \cdot \ldots \cdot Xn)' = X1' + X2' + \ldots + Xn'$

$(T13')(X1 + X2 + \ldots + Xn)' = X1' \cdot X2' \cdot \ldots \cdot Xn'$

Principle of Duality: Only interchange operations.

Example: The dual of $F = X \bullet Y + Z \bullet W$ *is* $F^D = (X + Y) \bullet (Z + W)$.

Exercise 4:

Two (and Three) Variable Theorems

(T6) X + Y = Y + X	(T6') XY = YX	*(Commutativity)*
(T7) (X + Y) + Z = X + (Y + Z)	(T7') (XY)Z = X(YZ)	*(Associativity)*
(T8) XY + XZ = X (Y + Z)	(T8') (X + Y)(X + Z) = X + YZ	*(Distributivity)*
(T9) X + XY = X	(T9') X(X + Y) = X	*(Covering)*
(T10) XY +XY' = X	(T10') (X + Y)(X + Y') = X	*(Combining)*
(T11) XY +X'Z + YZ = XY + X'Z		*(Consensus)*
(T11') (X + Y)(X' + Z)(Y + Z) = (X + Y) (X' + Z)		

Adsorption Theorem (not in the textbook):
(T*) X + X'Y = X + Y **(T*)'** X(X' + Y) = XY

All theorems can be proved, either *algebraically* or through the use of a *truth table*.

Simplify $F = [AB'(A + C)']' [(A'B)' + (A + B' + C')]$

Simplify $F = (A + B)(A + C')(A + D)(BC'D + E)$

3 Karnaugh Maps

Another approach to represent (and simplify) Boolean equations.

3.1 Minterm and Maxterm

1. **minterm:**
   - A product of n distinctive logic variables (or their complements); e.g., X · Y · Z;
   - The sum of minterms corresponds to the combination of Truth Table rows for which the function produces a 1 output;
   - Each variable is represented by its complement if the variable value is 0.
2. **maxterm:**
   - The sum of n distinctive logic variables or their complements e.g., X+Y+Z.

- The product of maxterms corresponds to the product of Truth Table rows for which the function produces a 0 output;
- Each variable is represented by its complement if the variable value is 1.

EXAMPLE: Write the boolean function in the form of Sum of Minterms and Product of maxterms.

X	Y	Z	F	$\overline{F}$	
0	0	0	1	0	m_0
0	0	1	0	1	m_1
0	1	0	1	0	m_2
0	1	1	0	1	m_3
1	0	0	0	1	m_4
1	0	1	1	0	m_5
1	1	0	0	1	m_6
1	1	1	1	0	m_7

3.2 Karnaugh Map

Tips: The minterm numbers do not follow the normal binary counting order sequence.Only 1 bit changes from one adjacent column to the next.

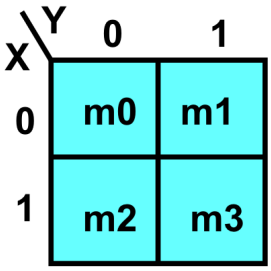


图 1: 2-variable k-map

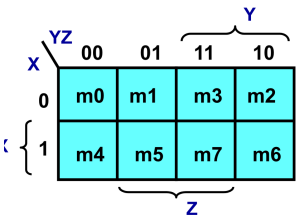


图 2: 3-variavle k-map

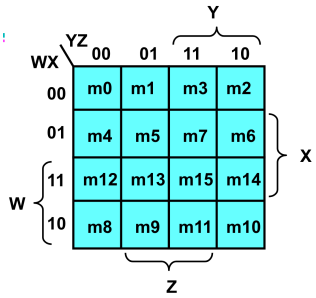


图 3: 4-variable k-map

Process of simplifying using k-map:

1. Put 1' s in boxes of corresponding terms and put 0' s in all other boxes.

2. group adjacent 1' s (in powers of 2) and eliminate unnecessary variables.

3. Remember to circle the largest groupings possible!

Exercise 5:

Simplify $F = X' \bullet Y' \bullet Z' + X' \bullet Y' \bullet Z + X' \bullet Y \bullet Z' + X \bullet Y' \bullet Z' + X \bullet Y' \bullet Z$ using k-map and draw the **NAND** gate implementation.

3.2.1 Prime Implicants

When we do grouping, we should always ensure that the size of each group is maximised and the number of groups is minimised to get **minimal sum of products**.

Prime Implicants: Largest possible grouping of 1' s.(for each 1)

Essential Prime Implicants: If any group contains a minterm that isn' t also covered by another overlapping group, then that is an EPI.

Exercise 6:

Find PIs, EPIs, and the equation.

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

Exercise 7: Find the minimal sum of products for F where, $F = X'Y'Z + X'YZ' + X'YZ + XY'Z' + XY'Z$.

X \ YZ	00	01	11	10
0	0	0	0	0
1	0	0	0	0

Don't care:

W	X	Y	Z	F
0	0	0	0	x
0	0	0	1	1
0	0	1	0	x
0	0	1	1	1
0	1	0	0	0
0	1	0	1	x
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

F(W,X,Y,Z) = Σm(1, 3, 7, 11, 15)
d(W,X,Y,Z) = Σm(0, 2, 5)

Answer:
F = YZ + W'X'

WX \ YZ	00	01	11	10
00	x	1	1	x
01	0	x	1	0
11	0	0	1	0
10	0	0	1	0

4 Hazards and Glitches

Output glitch: A momentary unexpected output change (short pulse) when an input changes; usually caused by gate propagation delays.

Hazards: A timing hazard exists in a combinational circuit when it produces an output glitch when one or more inputs change.

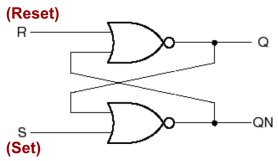
Find adjacent groups!

- A **static-1 hazard** is a pair of input combinations that: (i) differ in only one input variable and (ii) both give a 1 output; such that it is possible for a momentary 0 output to occur during a transition in the differing input variable.
- A **static-0 hazard** is a pair of input combinations that: (i) differ in only one input variable and (ii) both give a 0 output; such that it is possible for a momentary 1 output to occur during a transition in the differing input variable.

Exercise 8: Is there any static hazard on $F = A'D' + A'B'C' + ABC + ACD$?

5 Latches and Flip-Flop

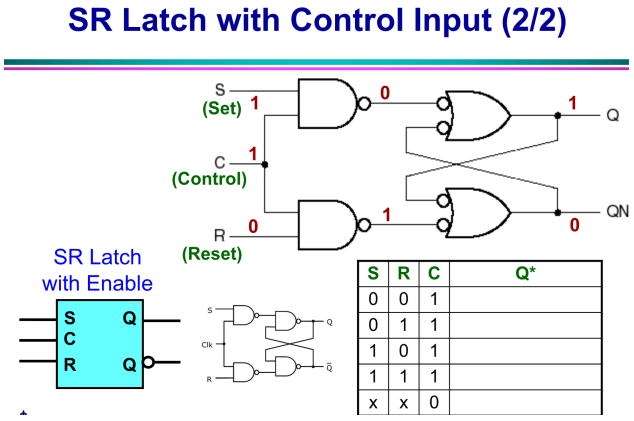
5.1 SR Latch(NOR gate)



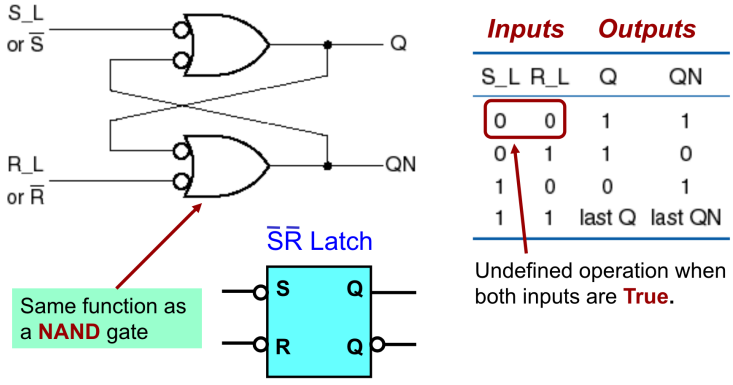
Inputs		*Outputs*	
S	R	Q	QN
0	0	last Q	last QN
0	1	0	1
1	0	1	0
1	1	0	0

Undefined operation when both inputs are *True*.

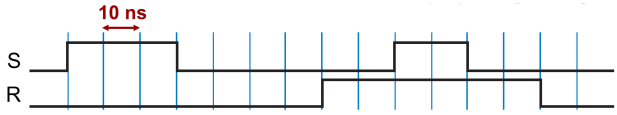
Oscillation occurs if both S and R return to 0 simultaneously!



5.2 $\overline{S}\ \overline{R}$ Latch

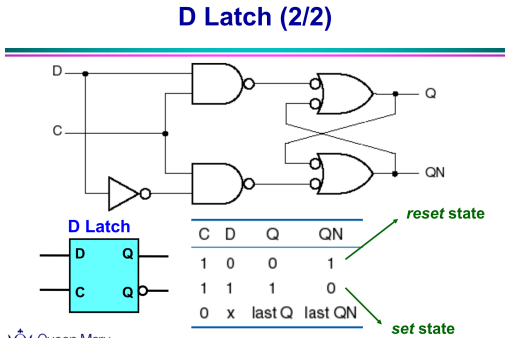


Exercises 9: Draw the outputs of an SR latch for the input waveforms shown below. Assume that the propagation delay of a NOR gate is 10ns.

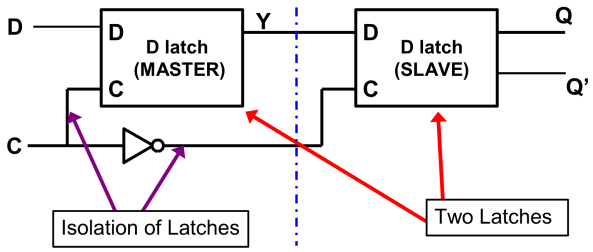


5.3 D Latch

- Can eliminate the indeterminate state by ensuring that both Set and Reset inputs are never equal to '1' at the same time!
- Has only two inputs: Data (D) and Control (C).

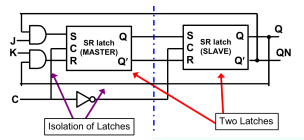


5.4 Master-Slave D Flip-Flop

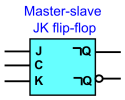


5.5 Master-Slave JK Flip-Flop

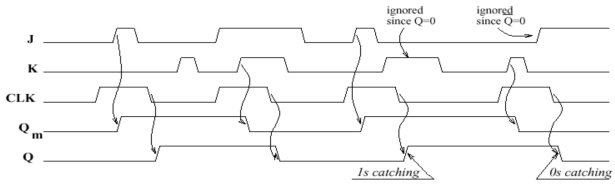
No indeterminate state!



J	K	C	Q	QN
x	x	0		
0	0	⊓		
0	1	⊓		
1	0	⊓		
1	1	⊓		

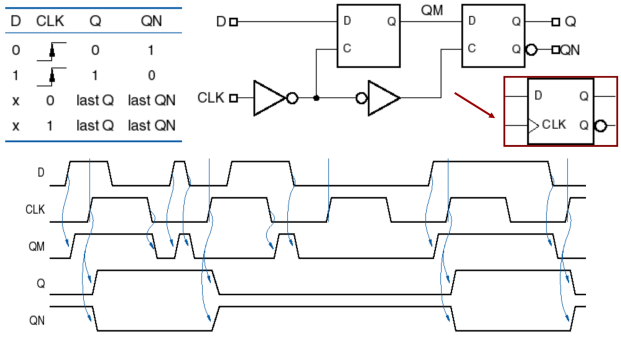


- 1’ s Catching: Output changes to 1 even though K and not J is asserted at the end of the triggering pulse.
- 0’ s Catching: Output changes to 0 even though J and not K is asserted at the end of the triggering pulse.



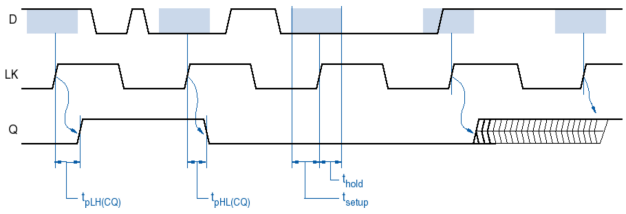
5.6 Edge-Triggered Flip-Flops

Ignore inputs while the clock pulse is at a constant level.Only set outputs on clock pulse transitions. Thus solving the catching problems.



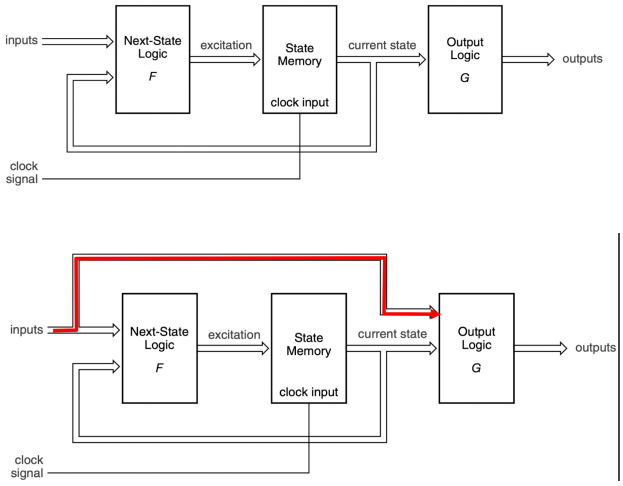
5.7 Timing parameters

- Propagation Delay
- recovery time(SR Latch)
- setup time
- hold time



6 Sequential Circuit Analysis

6.1 Structure



6.2 3 equations and 3 tables

- **Input Equation(excitation)**.A boolean equation expressesed by Next-state logic , in terms of the input to the state memory. equation
- **Characteristic Equation**.Determined by the characteristic equation for the flip-flop type.

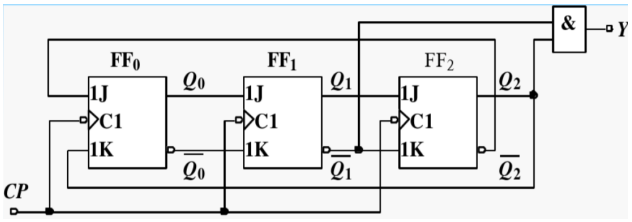
Device Type	*Characteristic Equation*
S-R Latch MS S-R Flip-Flop	$Q^* = S + R'Q$
D Latch	$Q^* = D$
JK Latch MS J-K Flip-Flop **ET J-K Flip-Flop**	$Q^* = JQ' + K'Q$
ET D Flip-Flop	$Q^* = D$
D Flip-Flop w/ Enable	$Q^* = E_N D + E_N'Q$
T Flip-Flop	$Q^* = Q'$
T Flip-Flop w/ Enable	$Q^* = E_N Q' + E_N'Q$

- **Output Equation**
- **Transition Table**:Expresses the next state as a function of the current state and the input.
- **State Table:**Expresses the next state as a function of the current state and the input using alphanumeric state names.
- **State/Output Table:** Similar to the State Table, but includes the output as well.

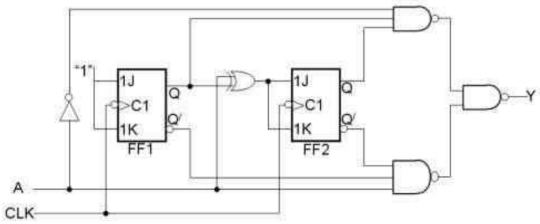
6.3 Process

1. Obtain the *input* (or ***excitation***) ***equations***.
2. Obtain the ***output equations***.
3. Obtain the *next state* (or ***characteristic***) ***equations***.
4. Substitute the *input equations* into the *next state equations* to obtain ***transition equations***.
5. Develop a ***transition table*** from the *transition equations*.
6. Develop a ***state table*** that relates the possible states in terms of the *present* and *next state*.
7. Develop a ***state/output table*** that relates the possible states in terms of the *present* and *next state, together with the outputs.*
8. Draw the ***state diagram***.

Exercise 10: Analyze the following sequential circuit, draw the transition diagram.



Exerciese 11: Analyze the following sequential circuit, draw the transition diagram.



7 Autonomous Sequential Circuit Design

Autonomous circuits do not have primary inputs, they have only secondaries (plus a clock signal).

7.1 Process

1. Draw-up a table of present and next states.Need to take account of the characteristic equation of the flip-flop
2. Draw a Karnaugh map for each next state output(flip-flop input) in terms of the present state.
3. Minimise the logic using Karnaugh map simplification
4. Draw-up the corresponding circuit diagram for the FSM.

Exercise 12: Design a 6-state counter using the first six binary numbers. Use JK flip-flops.

Exercise 13: Design a ring adder with an effective cycle state of 100-010-001-100.

谢谢！

祝大家取得好成绩！