

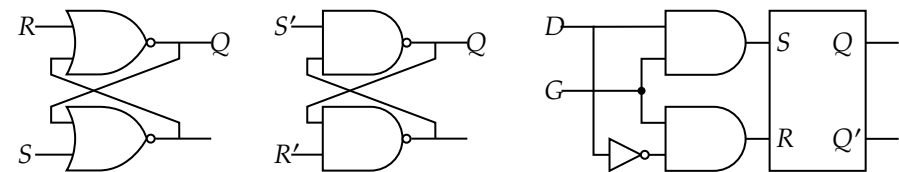
# Digital System Design and Laboratory Final

## Misc Components

- Multiplexer, Three-State Buffers, Decoder, Encoder (one hot, priority based), Read Only Memory, Programmable Logic Arrays
- FPGA: function generator (Shannon’s Expansion)

## Latches — Asynchronous

- SR Latch (Set-Reset),  $SR \neq 1$  — Application: Switch Debouncing
- Gated D Latch (Data)

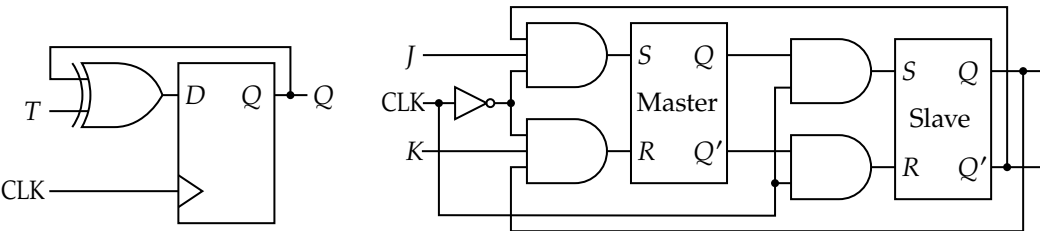
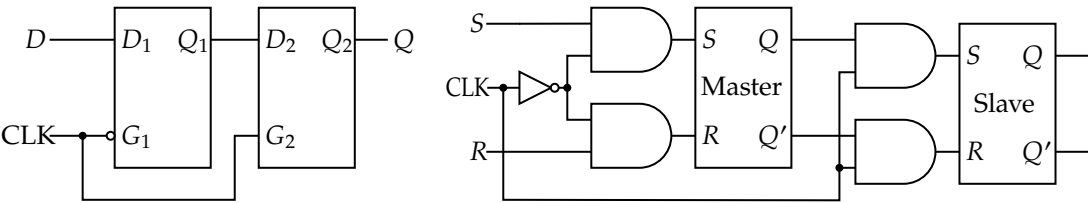


## Flip-Flop — Synchronous

$Q \rightarrow Q^+$	00	01	10	11
D Flip-Flop	0	1	0	1
S Flip-Flop	0	1	0	x
R Flip-Flop	x	0	1	0
T Flip-Flop	0	1	1	0
J Flip-Flop	0	1	x	x
K Flip-Flop	x	x	1	0

Data  
Set-Reset  
(Not desired — S, R can change only at high clock)  
Toggle  
00 Hold, 10 Jump, 01 Clear, 11 Toggle  
(same issue as SR)



- Setup Times (input should be prepared) and Hold Times (input should be hold)
- Additional Inputs:
  - ClrN: clear (often inverted), PreN: set 1 (often inverted), CE: clock enable



## Registers

- Normal, Shift Register, PIPO Shift
- Counter: Ensure don’t care states falls into loop, or reset upon power up

## State Machines

- Moore Machine:  , Mealy Machine:  (prone to glitches and spikes)
- Serial Data Transmission
  - NRZ: 0 = 0, 1 = 1, NRZI: 1 = T, NRZ: 0 = 00, 1 = 10, Manchester: 0 as 01, 1 as 10
- State Equivalence
  - All output of input are the same, or all next states and outputs are equivalent
- State Reduction: Implication Table
  - Draw a table (only half for simplification, full for equivalence test)
  - Each square has #input pairs, with conditions that two state are equivalent
  - For example  $A \rightarrow (B, C)$  and  $D \rightarrow (E, F)$ , put B-E and C-F
  - Delete self implied pairs
  - Cross out if output is already different, or pair is already crossed out
- State Assignment — Heuristic: Put “similar” states adjacent: same next state, output, (one of) previous states

# Digital System Design and Laboratory Midterm

## Switching Circuit

- Asynchronous  $\leftrightarrow$  Synchronous
- Combinational  $\leftrightarrow$  Sequential

## Numbers and Codes

- 1's complement: Overflow carry wrap around
- 2's complement: Overflow carry don't care
- Binary Code: For display

## Boolean Algebra

- Precedence: (brackets), NOT, AND, OR
- ( $2^{nd}$  Distributive)  $X + YZ = (X + Y)(X + Z)$
- ( $1^{st}$  Elimination)  $X + X'Y = X + Y$
- ( $1^{st}$  Consensus)  $XY + X'Z + YZ = XY + X'Z$
- ( $2^{nd}$  Consensus)  $(X + Y)(X' + Z)(Y + Z) = (X + Y)(X' + Z)$
- (Sum Product Exchange)  $(X + Y)(X' + Z) = XZ + X'Y$
- $XOR \oplus, XNOR \equiv$

## Truth Table and Karnaugh Map

- Minterm  $\sum m(6) = ABC'$
- Maxterm  $\prod M(6) = (A' + B' + C)$
- Don't care:  $\sum d$  or  $\prod D$

- Implicant: A product term
- Prime Implicant: A maximal Implicant
- Essential Prime Implicant: A PI with exclusive minterm(s)
- Quine-McCluskey: PI finding, systematic, useless for human

## Circuit Design

- Level counts from output
- Two-level Circuit:  
AND-OR  $\leftrightarrow$  NAND-NAND  $\leftrightarrow$  OR-NAND  $\leftrightarrow$  NOR-OR,  
OR-AND  $\leftrightarrow$  NOR-NOR  $\leftrightarrow$  AND-NOR  $\leftrightarrow$  NAND-AND,  
others are degenerate
- Multi-level: use OR-AND alternating and bubble-pushing to form NAND-NAND and NOR-NOR
- Multiple-output: Pay more attention to essential and common implicants

## Delay and Hazard (SOP)

- Delay causes hazard
- When **analyzing** or **constructing hazard free circuit**, DO NOT assume  $XX' = 0$ ,  $X + X' = 1$ , i.e. using simple factoring and DeMorgan's law.
- Static 1-Hazard: Change from a implicant to another
- Static 0-Hazard: Multi-level, from  $X'X\alpha$ .
- Dynamic Hazard: Multi-level, from  $X'X\alpha$ , single difference propagates  $\geq 3$  different paths.