Digital Logic & Design Study Guide

Logic Gates

- AND x, y → x & y
- OR x, y → x | y
- XOR x, y → x ⊕ y
- NOT x → x'
- NAND x, y → (xy)' or (x & y)'
- NOR x, y → (x+y)' or (x | y)'

* There are many circuit / gate combos which have the same F

Design Levels

① functional
② transistor
③ truth table
④ gate
⑤ verilog / structual / behavioral
⑥ matlab / simulink

Moore's Law

↳ # transistors on a microchip doubles every year

Number Bases

- decimal → anything
  ① divide by that base
  ② remainders give converted #s starting w/ most sig # (left)
- binary ↔ octal
  → break down into 3's
  or
  ← convert by digit
- binary ↔ hex
  → break down into 4's
  or
  ← convert by digit
- hex ↔ binary ↔ octal

Binary math

- Half adder
  S = A ⊕ B (xor)
  C = AB
  A, B → sum, carry
- Full adder
  $C_o = ab + (a+b)C_{in}$
  $S = A \oplus B \oplus C$
  A, B, C_{in} → sum, C_{out}
- overflow (v) = $C_N \oplus C_{N-1}$
  ↳ if v = 1, append C_N carry

Ex
6 → 0110, -3 → 1101 → 0011
v = 1 ⊕ 0 = 1, C_N ignored

-6 → 1010, -3 → 1101 → 10111
v = 1 ⊕ 0 = 1, C_N included

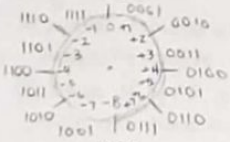
① Convert to binary
② Add w/ carries

1's & 2's Compliment

- if signed ##.## (1 = −, 0 = +)
- 1's comp
  ↳ invert digits
- 2's comp
  ↳ 1's comp + 1

Ex +4 → 0100
1011 1's comp
+ 1
12 ← 1100 2's comp
mod(12, 2^4) = −4
↳ mod to get neg value

- 2's comp addition
  ↳ reg binary add ignoring carries beyond MSB
- subtraction
  ↳ add 1st # to 2's comp of 2nd #
  ↳ ignore carries beyond MSB



↳ if no carry produced, take 2's comp if ans is neg

Fixed & Floating Pt

- floating → decimal
  ① 1st # = sign (0 → +, 1 → −) = s
  ② next 8 # convert bin2dec
  ↳ = e
  ③ remaining # convert bin2dec
  ↳ = m
  ④ $x = (-1)^s(1 + \frac{m}{2^{23}})2^{e-e_{bias}}$
  * $e_{bias} = 127$
- decimal → floating
  ① Split whole . dec
  ② whole dec2bin
  ③ dec dec2bin
  ④ whole Bin . dec Bin (#spots moved)
  ⑤ format ↓ to #.###... ×2^y
  ⑥ s = 1 if neg, = 0 if pos
  ⑦ e_{bias} + # spots moved = e
  ⑧ e dec2bin
  ⑨ | s | e | #'s after decimal |
  bits: 1 | 8 | 23

Duality

- interchange & and | and 1's and 0's
- Boolean / Shannon
  F(x, y, z) = x · F(1, y, z) + x' · F(0, y, z)

De Morgan's

(x·y)' = x' + y'
(x+y)' = x'y'

- complement = dual w/ variable replaced w/ complements

Combinational Circuits

① Analysis (determine F)
↳ Label all branches to find F
↳ Simplify
↳ Truth table for F
↳ MATLAB to make timing plots

② Synthesis
↳ make truth table
↳ SOP (Σ) = eqn where F = 1
↳ POS (Π) = eqn where F = 0
↳ simplify resulting equation

OR use kmaps

* maxterm = complement of midterm

Decoder & Encoders

- often uses 1-hot encoding
- convert all outputs we don't care about into "don't cares" (x's)

① Truth table w/ don't cares
② kmap for minterms (include don't cares BUT, all groups must have ≥ one actual 1)
③ Discovered equations are output equations

* Same process encoder or decoder

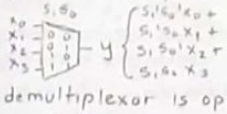
- priority encoder
  ↳ minimize outputs

Ex given $y_3 y_2 y_1 y_0$
$I_3 = y_3$
$H_2 = y_2 y_3'$
$H_1 = y_1 y_2' y_3'$
$H_0 = y_0 y_1' y_2' y_3'$
↳ compressed truth table gives 1-hot encoding

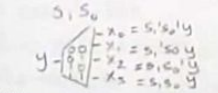
Map: y → H → encoder / decoder

Multiplexing

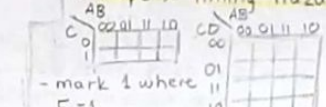
- combine multiple signals into 1 signal
- input S selects which n inputs are transferred



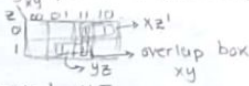
- demultiplexor is opposite



K Maps & Timing Hazards



- mark 1 where F = 1
- group 1's in powers of 2 (ex. 2, 4, 8, 16)
- use groups to calculate SOP



- F = xz' + yz
  but! this will have timing hazards. To fix, add overlap term
  → F = xz' + yz + xy
  ↳ complete equation

Comparator

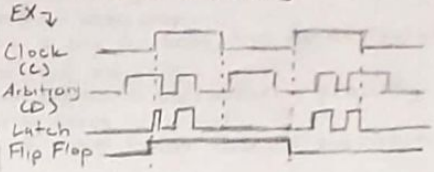
- compares 2 binary #s

$a \oplus b \rightarrow = 1$ if $a \neq b$
$(a \oplus b)' = 1$ if $a = b$
→ $e_n = (a_n \oplus b_n)'$ equivalancy
→ $E = e_n \cdot e_{n-1} \cdots e_1 e_0$
→ $G_{a>b} = a_3 b_3' + e_3 a_2 b_2' + e_3 e_2 a_1 b_1' + e_3 e_2 e_1 a_0 b_0'$
↳ compare $a_3\ a_2\ a_1\ a_0$ with $b_3\ b_2\ b_1\ b_0$ >?
→ $L_{a<b} = (E + G)'$

Latches & Flip Flops

- flip flop → time intervals
- latches → continous

EX ↓

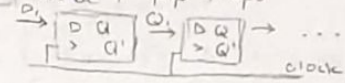


→ Latch
↳ C = 1, latch = D
C = 0, latch = 0
→ Flip flop
↳ C switches ↓ (up edge)
ff = D

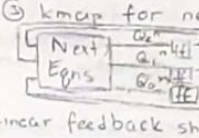
* See ff summary in Seq. circuits for more

Registers & Counters

- shift register → adds delay t/c made w/ flip flops



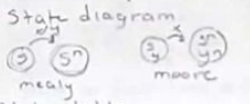
- counter
  ↳ n bits for length N $n = \lceil \log_2 N \rceil$
  ① given transition table
  ② make state table
  ③ kmap for next eqns



- linear feedback shift register
  ↳ gives pseudorandom sequence
  Form: $Q_2^{next} = Q_1 \oplus Q_0$
  $Q_1^{next} = Q_2$
  $Q_0^{next} = Q_1$

Finite State Machines

- Design ↓
  ① Choose state & reset state from description
  ② State diagram



  ③ State table

Present	Next x=0	Next x=1	Out y
S_0	S_0 S_0		?
S_1	S_1 S_0		?
S_2	S_2 S_0		?

  ④ Encode AB → S
  ⑤ Implement w/ gates & simulink
  ↳ equations from kmaps of state tables

Synthesis ↑
Analysis ↓