CS207 Digital logic

number

base convert

- Notation: $(value)_{base}$
- 10 to other
- other base r to 10
  - For fraction, multiply r untill the fractional part is 0。
- between binary, octal, hexadecimal

complement

- r's
- r-1's

subtraction

- Steps of dong M-N:
  - make M and N have the same number of digit by adding zero on more significant bit of the smaller number
  - calculate the r's complement of N, resulting in A
  - do M+A
  - If $M \geq N$, remove the carry
  - if $M < N$, the result will not have a carry. Instead the result is the r's complement of N-M. So to get the real result of M-N, you need to
    - get the r's complement of M+A,
    - then add a minus sign before it.

Signed binary number

- For positive number the sign bit value is 0, for negative the value is 1.
- The r and r-1's complement representation of signed binary numbers are the same as the sign-magnitude representation.
- For negative number are NOT. But the sign bit will not change during complement.
- For 0, +0 is always represented by 0000, but for the -0, 2's complement representation not exist, other need calculate.

Binary codes

BCD (binary coded decimal)

- denote: $(value)_{BCD}$
- If a number more than 9(1001) appeared, then add 6(0110)
- If a carry appeared, add 3 zero before it. As every number in BCD should be represented by FOUR digits.
- In addition, if a number more than 9 appeared, after adding 6, the carry appeared should be add to more significant bit.

Gray code

- A number changes by only one bit as it proceeds from one number to the next
- 000-001-011-010-110-111-101-100
- Always let the LSB change first.

Binary storage and registers

- Binary cell
- Register

Boolean algebra

operators

- XOR

universal

- NAND
- NOR

properties

- $A + 0 = A, A \cdot 1 = A$
- $A + 1 = 1, A \cdot 0 = 0$
- $A \cdot A' = 0, A + A' = 1$
- $A \cdot A = A, A + A = A$
- $(A + B)(A + C) = A + BC$
- $A + A'B = A + B$
- $(A + B)(A + B') = A$
- $AB + AB' = A$

postulates

- Distributive law
  - NOT just for AND, OR is also valid
  - $A + (BC) = (A + B)(A + C)$
- DeMorgan
- Absorption
  - $A + AB = A = A(1 + B) = A \cdot 1$
  - $A(A + B) = A$

Boolean function

Canonical forms

SOP POS

- POS is different from SOP.
  - As pos are product of sum, in the Karnaugh map, you should group value 0 and add them before multiply them.

minterm maxterm

- involve ALL number
- Min: and
- Max: or,
- obtain sum of minterm/ product of maxterm
  - use$\sum$ / Π to represent. And they are mutually exclusive
  - Get product of maxterm:
    $(A + B) = (A + B + 0) = (A + B + CC') = (A + B + C)(A + B + C')$
    - Using $(A + BC) = (A + B)(A + C)$

Digital logic gates

- universal : NAND and NOR

Gate-level Minimization

- using boolean algebar or Kmap
- be careful about the requirement. SOP and POS is different from sum of minterm and product of max term

Karnaugh map

- The index is 00,01,11,10, which is quite different from the regular sequence.
- When the POS is the target, group the cell of 0, and for digit equal to 1 do the complement rather than 0.
- When drawing Karnaugh Map, de not forget to determine the represent of input

grouping tricks

- group 4 cell in the edges
- be careful about where are don't care condition

Two-level implementation

NAND and NOR

- adding bubbles.

XOR

- The result of $A \oplus B$ is whether A and B are different.
- If $A = B'$, they are different, the result is 1.
- If $A = B$, they are similar the output is 0.
- indentities
  - $X \oplus 0 = X$
  - $X \oplus 1 = X'$
  - $X \oplus X = 0$
  - $X \oplus X' = 1$
- Odd function
  - Can be even function by adding a bubble before the final result F.
- Parity generation and checking
  - A parity bit is an extra bit included with a binary message to make the number of 1's either odd or even.

Combinational logic

Design

- steps
  - determine the number and representation of input and output
  - draw truth table
  - Obtain simplified equation of every output by kmap

multiplexer

- The device has two control or selection lines A and B and an **enable line E**.
- can be used to implement boolean function

Interconnecting multiplexers

- enable line E can be used as an input as well
- **two levels of multiplexing**

Multiplexer as a Boolean function generator

- The residue variables can be formed into residue functions which can then be applied at the data inputs.
- The input of MUX can be NOT just 0/1, it can be a real input like C

decoder

- n to 2^n
- **minterm generator**:
  - bubbles after decoder and the and gate
  - It can be used as the representation of sum of product with a OR gate after the decoder.
- Decoder network
  - tree
  - Coincident

incoder

- 2^n to n

Sequencial logic

latches

flip-flop

- For EVER ff, the input will always contain a clk.

	JK	T	D
input equation	00:Q, 01:0, 10:1,11: Q'	0:Q, 1:Q'	0:0, 1:1
characteristic equation			

Analysis

- state table
    - input of circuit
    - present state
    - next state
    - output of curcuit
- state diagram
    - diagram is NOT table
    - The number on arrow separated by / is input/output
    - state is in circule
- input equation of ff
    - represent J/K/T/D by current state and input
- output of circuit

Finite state machine

- Mearly model : output may not synchronous
- Moore model

state reduction

- the state table should be written in the form that output and next are together with input
- If the output and the next state of 2 state on EVERY input of 2 **states** are the same, replace one **state** with another.

Design

- steps
    - draw state diagram
    - reduce state
    - State table
    - choose ff type
    - ff input equation, outptu equation
- When draw the circuits, do not forget the clk for EVERY ff

Arithmetic Circuits

Adder

- The difference between half-adder and full-adder is **the number of inputs**. The half-adder only have 2 inputs, while the full-adder have 3 inputs, **including the previous carry, denoted by x**.

half-adder (HA)

- $S = A \oplus B$
- $C = AB$

full-adder (FA)

- $S = X'A'B + X'AB' + XA'B' + XAB = (A \oplus B) \oplus X$: Use the property of XOR gate as the odd function. Only the odd number of 1 exist, the output is 1.
- $C = AB + BX + AX$

Binary Adder

- It is constructed with full adder connected in cascade, the carry is sent from LSB to MSB.
- For the n_{th} FA, there are 2 input A_{n-1} and B_{n-1}
- This method using FAs are more simple compared with the one using classical method.

Carry propergation

- The carry propagation time is an important attribute of the adder because it limits the speed with which two numbers are added.
  - Since all other arithmetic operations are implemented by successive additions, the time consumed during the addition process is critical.
- A solution is to increase the complexity of the equipment in such a way that the carry delay time is reduced.
  - The most widely used technique employs the principle of **carry lookahead logic**.

Implementation

- For the i_{th} FA (i = 0,1,...n-1):
  - $P_i = A_i \oplus B_i$
  - $G_i = A_i B_i$
  - $S_i = P_i \oplus C_i$
  - $C_{i+1} = G_i + P_i C_i$
  - In this representation, C_{i+1} is the output carry of the i_{th} FA, C_i is the previous carry X of the FA.

Subtraction

- D: difference
- B: borrow from more significant bit

half-subtractor

- $D = X \oplus Y$
- $B = X'Y$: only when the minuend is 1 and the subtrahend is 0 will the borrow be 1

full-subtractor

- It has one more input Z, generated from the subtraction operation of previous significant digits
- $D = X\prime Y\prime Z + X\prime YZ\prime + XY\prime Z\prime + XYZ$
- $B = X\prime Z + X\prime Y + YZ$

Binary subtractor

- The addition and subtraction operations can be combined into one circuit with one common binary adder by **including an exclusive-OR gate** with each full adder.
    - It is consist of a series of FA
    - The output c of every FA is connected to the more significant bit as the input x

overflow

- To solve it, add a XOR gate on the last 2 carries. When it is 1, an oerflow appeared.

decimal adder

binary multiplier

- It is similar to the decinal multiplier.
- For one bit binary number, $A * B = AB$, as only when both of them are 1, the result is 1.

Register

- An n-bit register consists of a group of n flip-flops, capable of **storing n bits of binary information**.
- In addition to the flip-flops, a register may have combinational gates that perform certain data-processing tasks.
- **Register vs counter**: Essentially **a counter are a register** that goes through a predetermined sequence of binary states.

Shift register

- A register capable of shifting its binary contents either to the left or to the right is called a shift register.
- Unidirectional shift register: A register capable of shifting in one direction only is a unidirectional shift register.
- Bidirectional shift register: One that can shift in both directions is a bidirectional shift register.
- Universal shift register: If the register has both shifts and parallel-load capabilities, it is referred to as a universal shift register.

serial/paraellel - in/out register

- For serail-in parallel-out register, n-bit of number need n clock pulses to into the register.

universal shift

- cl
- clk
- Shift-right&left control

Shift register counter

- For ALL counter, the state of **ALL ff are part of the output.**
- But for a sequence generator, only the left-most ff (usually a D ff) is the output.

Ring counter

- Init at L from 0001 -> 1000 -> 0100
- When PR is 0, then the output is 1.
- And when CLR is 0, then the output is 0.
- Both PR and CLR are active low signal that always works in value 0.
- So before pr and clk all have a bubble.

Johnson counter

- Init at L from 0000 -> 1000 ->1100
- Disadvantage

Sequence generator

- A sequence generator is a circuit that generates a desired sequence of bits in synchronization with a clock.
- It is implemented by D ff.
- For a sequence generator with n D ffs, the seqnence generator is the Q_{n-1}

Design

- steps
  - As N is the modulus, length of the sequence, using the formula $2^n - 1 \geq N$ to get the valid n_{min}.
  - Name the state of sequence generator ff to Q_{n-1}.
  - Use n_{min} to draw a table of n_{min} column, N rows. You can draw a table of $n_{min} + 1$ column, reserving one column for z.
  - Check whether there are repeat rows. If so, $n++$
  - Add the column of z.
  - draw Karnaugh Map of z
  - draw the circuit:
    - Draw D ff.
    - draw clk
    - mark Q_{n-1} to Q_0
    - draw sequence generator at Q_{n-1}
    - draw z

Serial addition

- It consist of 2 shift registers, a full adder and a D ff.
- The D ff and 2 shift register share the same clk input.

Counter

- The meaning of counter here is more than a counter, although it can be used as an insreument of measuring time.
- It is usually constructed from one or more flip-flops that **change state in a prescribed sequence** when input pulses are received.

ripple/asynchronous/serial counter

- The simplest one using T ff to implement.
- The T input of each flip-flop is connected to a constant 1.
- The clock inputs of the three flip-flops are connected in cascade. And as before EACH clk input of T ff there exist a BUBBLE, so the ff will toggle at teh negative edge of its clk.
- Only the FIRST ff is driven by a real clk as the purpose of this circuit is to count the number of pulses that occur on the primary input CLK (Clock).
- MOD-Number: The modulus of a counter is the total number of unique states it passes through.
- N **vs** n: N is the modulus, n is the number of T ff.
    - But in sequence generator, the N represent the length of sequence, n represent the number of ff. $2^n - 1 \geq N$
- **Frequency divider**: The frequency of the n^{th} ff is $\frac{f}{2^n}$, where f is the frequency of the clk. So the clk does NOT equal to the frequency of any ff.
- The maximum binary number that can be counted by the counter is 2^n–1: Becase the maximum of **state** it can represent is 2^n, and the begning of the count is 0, so the maximum of **number** it can count is the maximum number of state minus 1.
- When the modulus is 2^n, the output Q of the end ff is useless. It is NOT connected to anything.
- The left-most T ff represent the LSB, as it is connected to the clk.
- sequence of draw the circuit:
    - Draw the required number of **T** ff, which is represented by little n.
    - Draw input T of each t ff (all is 1).
    - Draw the clk of each ff. The first is clk, other are state of previous ff. Do NOT forget the bubble before each clk.
    - Draw cl(clear) if $N < 2^n$. The not of state(Q') may be used. Do NOT forget the bubble before the cl.

asynchoronous counter with modulus $< 2^n$

- To achieve this, skip some states.
- To skip some states, the **cl**(clear, make the state of a ff to be 0) input is needed. The **first useless** state(NOT the last valid state) that will be reach should let the cl of EVERY ff to be 1. This can be implemented by a NAND gate.

Asynchronous down-counter

- 3 way to implement.

	1	2	3
reverse of Clk	Q	Q'	Q=Q'' (remove the bubble before clk.)
State	Q'	Q	Q

Multimode/up-down counter

Synchronous

- The ripple or asynchronous counter is the simplest to build, but its highest operating frequency is limited because of **ripple action**.
- Compared with the asynchronous counter, the T ffs of synchronous counter have same clk, and the input T of each T ff is the state of previous T ff except the first T ff, which is 1. But in asynchronous counter, the input T of each T ff is 1.

Design a synchronous counter

- The steps are similiar to design a sequencial circuit.
- Steps:
  - Draw state diagram. (NOT the table)
  - Find the N(number of ff required).
  - Decide the ff type. (NOT only T ff is possible).
  - Determine the ff input.
  - Draw Karnaugh Map of EACH ff input.
- This is different from the counter implemented by T ffs, as the ff input may not directly connected to the state of previous T ff. It will be a combinational logic consisting of the present state of each ff.

Lock out

- It happend in counter with modulus $< 2^n$
- It may be possible that the counter might go from one unused state to another and never arrive at a used state.A counter whose unused states have this feature is said to suffer from **lock out**.
- To ensure that lock out does not occur, we design the counter assuming the next state to be the initial state, from each of the unused states.
- Solution:
  - Connect each invalid state to a valid one and try to make the different digit between 2 connected state as small as possible.

- In the state table, write row of these invalid states.

#####