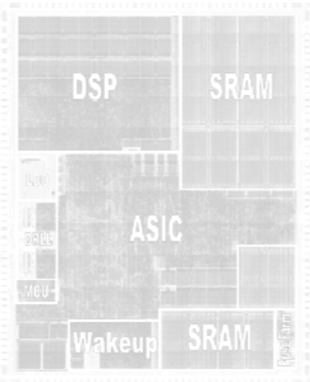
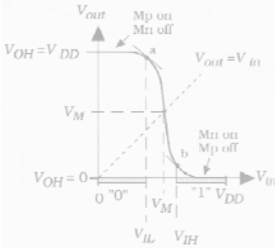
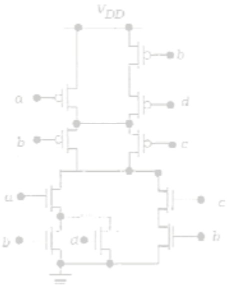


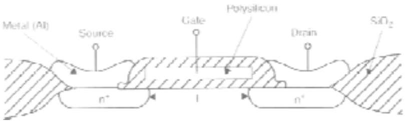
$$
\begin{aligned}
(a+b)\cdot(a+c) &= a + a\cdot b + a\cdot c + b\cdot c \\
&= a\cdot(1+b) + a\cdot c + b\cdot c \\
&= a\cdot(1+c) + b\cdot c \\
&= a + b\cdot c
\end{aligned}
$$

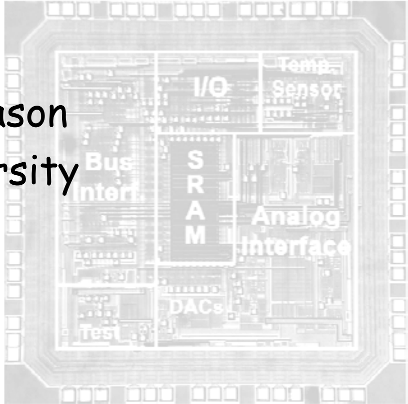


ECE 331: N0
ECE230 Review

Professor Andrew Mason
Michigan State University
Spring 2013







Opening Remarks

- Announcements
  - HW1 due next Mon
  - Labs begin in week 4
  - No class next-next Mon –MLK Day
  - ECE230 Review HO_0 posted on web
  - Anyone need a syllabus?

Questions?

- Today's Objectives
  - Perform number base conversions for Dec, Hex, Bin
  - Identify value range as function of number of bits; identify out of range overflow for signed and unsigned binary numbers
  - Express numbers in signed 2's complement (S2C) form, perform 2's complement operation, and evaluate subtraction using S2C.
  - Identify value range in S2C and determine 2C overflow
  - Perform minimization of logic expressions using min/max terms, K-maps, and Boolean arithmetic

Homework Guidelines

- Be neat!
  - should represent a professional work product
- Show work
- Clearly indicate answers
- Give units in answer
- Grading
  - effort more than results
  - may not be "corrected" but solutions will be posted
- Homework Questions
  - come to office hours!

Number Systems

- Digital Bases
  - Decimal (Dec), base 10
  - Binary (Bin), base 2
  - Hexadecimal (Hex), base 16
- Base Conversions
  - Bin → Dec
  - Hex → Dec
  - Bin → Hex

EX $1010_2 \rightarrow ?_{10}$, $\underset{8}{1\times2^3}+\underset{4}{0\times2^2}+\underset{2}{1\times2^1}+\underset{1}{0\times2^0}$

8421

$= 8+0+2+0 = \underline{\underline{10_{10}}}$

EX $2EB5_{16} \rightarrow ?_{10}$

show work! $= 2\times16^3+\overset{14}{E}\times16^2+\overset{11}{B}\times16+5$

4096 256

use calculator $= 2(4096)+14(256)+11(16)+5$

$= \underline{\underline{11,957_{10}}}$

EX $101101001101_2 = ?_{16}$

$\Downarrow$

group by 4 from LSB

$\underset{\underset{11}{8\,21}}{1011}\ \underset{\underset{4}{4}}{0100}\ \underset{\underset{13}{84\,1}}{1101} = \underline{\underline{B4D_{16}}}$

TPS: Convert 111101101 to hex.

Binary Addition

DEMO: using PC Calculator to check conversions & math
- only for *checking* answers!
- always show work in homework

- Addition Examples

Decimal		Binary		Hex
7	7 → 0111	0111	0111_2 → 7_{16}	7
+ 4	4 → 0100	+ 0100	0100_2 → 4_{16}	+ 4
11_{10}		1011_2		B_{16}

Decimal		Binary		Hex
7	7 → 0111	0111	0111_2 → 7_{16}	7
+ 9	9 → 1001	+ 1001	1001_2 → 9_{16}	+ 9
16_{10}		$1]0000_2$		10_{16}

- Value Range
  - *n* unsigned binary bits can only express values 0 to 2^n-1
    - *n* unsigned hex bit express values 0 to 16^n-1
  - larger values generate "overflow" → carry_out bit

Binary Subtraction

- Microprocessors do not subtract, they only add!
- 2's Complement = 2C
  - X-Y = X+(-Y) = X+[Y]*, where [ ]* means 2's complement
  - read extensive notes in HO_0a
- To subtract…
  - use numbers in signed 2's complement (S2C) form
  - use 2C operation to negate subtracted/negative values
  - then ADD

EX: Evaluate 5 – 3 using 4-bit S2C #'s

$$\begin{array}{r} 5 \\ -3 \\ \hline ? \end{array} \rightarrow \begin{array}{r} 0101 \\ -0011 \end{array} \xrightarrow{2C} 1100+1 \quad \begin{array}{r} 0101 \\ =\ 1101 \\ \hline 10010 \end{array}$$

value = 2

sign bit 0 = + (positive)

carry_out, ignore for subtraction

2's Complement Overflow

EX: Evaluate -7 – 6 using 4-bit S2C #'s

$$\begin{array}{r} -7 \\ -6 \\ \hline ? \end{array} \rightarrow \begin{array}{r} -0111 \\ -0110 \end{array} \begin{array}{c} \xrightarrow{2c} \\ \xrightarrow[2c]{} \end{array} \begin{array}{r} 1001 \\ +1010 \\ \hline 10011 \end{array}$$

value = 3

sign bit 0 = + (positive)

carry_out, ignore for subtraction

but does -7 – 6 = 3? No! should = -13

- -13 is out of range for 4-bit S2C #

- 2's Complement Overflow: when result of arithmetic is outside the value range of n-bit signed binary number
- Signed binary range: -2^{n-1} to $2^{n-1}-1$
- Detecting 2C Overflow: only overflow if sign of both numbers is same & is opposite sign of *addition* result

$$\boxed{Overflow = \overline{A_{n-1}} \cdot \overline{B_{n-1}} \cdot S_{n-1} + A_{n-1} \cdot B_{n-1} \cdot \overline{S_{n-1}}}$$

Arithmetic with 2C Overflow

EX: Evaluate using 3-bit S2C #'s, detect 2C overflow

$$\begin{array}{r} 011 \\ -101 \\ \hline ? \end{array} \quad \left| \begin{array}{c} 3 \\ -(-3) \\ \hline 6 \\ \text{DEC} \end{array} \right| \begin{array}{l} \rightarrow 011 \\ \xrightarrow[2c]{} 010+1 \end{array} = \begin{array}{r} {}^{1}0^{1}11 \\ 011 \\ \hline 110 \end{array} \rightarrow -001+1 = -010 = -2$$

negative↑

- 2C Overflow?
  - sign of both #s (at addition) = 0
  - sign of result = 1 (opposite of 0)
  - → 2C overflow!

- More examples in HO_0a

Bin/Hex Math Examples

EX: Evaluate using 4-bit S2C #'s, detect 2C overflow

$$\begin{array}{r}0110\\-1110\end{array}\rightarrow\begin{array}{r}0110\\0[001+1]\end{array}\rightarrow\begin{array}{r}0110\\+0010\\\hline 1000\end{array}\rightarrow\begin{array}{r}+6\\+2\\\hline \underline{8}\end{array}\leftarrow \text{no match}$$

$$\hookrightarrow -[111+1] = \underline{-8}$$

- meets conditions for 2C overflow, so answer is not reliable

EX: Evaluate using 4-bit S2C #'s, check answer

$$\begin{array}{r}E5\\-EE\end{array}\rightarrow\begin{array}{r}1110\ 0101\\-1110\ 1110\end{array}\rightarrow +[110\ 1110]^{*} = 001\ 0001+1 = 001\ 0010$$

$$\downarrow$$

$$\begin{array}{r}1110\ 0101\\+\ 0001\ 0010\\\hline 1111\ 0111\end{array}\quad\begin{array}{l}\rightarrow -(001\ 1011) = -(16+11) = -27\\\rightarrow -(0010010) = -(16+2) = -18\\= -[111\ 0111]^{*} = -[000\ 1001] = \underline{-9}\end{array}$$

$$\begin{array}{r}-27\\-(-18)\\\hline \underline{-9}\end{array}\ \checkmark$$

Logic Minimization

- Often need to reduce a complex logic expression to it smallest form (i.e., fewest number of logic operations)
- Methods of logic minimization
  - Boolean arithmetic using Boolean properties
    - EX $F = \overline{X}\cdot\overline{Y}\cdot Z + \overline{X}\cdot\overline{Y}\cdot\overline{Z} = \overline{X}\cdot\overline{Y}\cdot\left(Z+\overline{Z}\right) = \overline{X}\cdot\overline{Y}$
  - Karnaugh maps
    - EX: Find the minimal SoP expression for $F = \sum_{XYZ}(1,2,5,7)$

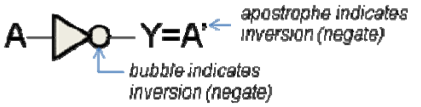
		XY			
		00	01	11	10
Z	0		m_2		
	1	m_1		m_7	m_5

- then reduce using Boolean arithmetic

F = XZ + Y'Z + X'YZ' = Z(X+Y') + X'YZ' is the minimal form.

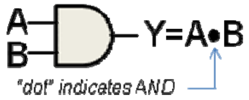
Basic Combinational Logic Gates

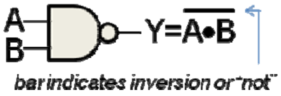
- INV


A
Y=A'
apostrophe indicates inversion (negate)
bubble indicates inversion (negate)

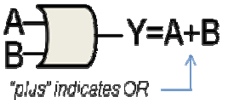
A	Y
0	1
1	0

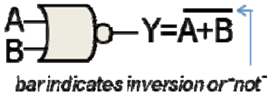
- AND
- NAND





- OR
- NOR



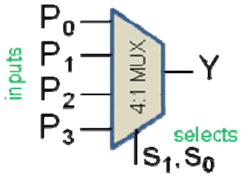


- inversion "bubbles"

inputs		AND	NAND	OR	NOR
A	B	A•B	$\overline{A\bullet B}$	A+B	$\overline{A+B}$
0	0	0	1	0	1
0	1	0	1	1	0
1	0	0	1	1	0
1	1	1	0	1	0

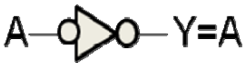
More Combinational Logic

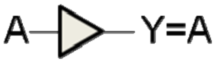
- MUX: select 1 from many
  - example 4:1 MUX



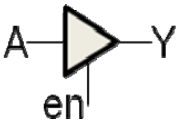
se ects		output
S_1	S_0	Y
0	0	P_0
0	1	P_1
1	0	P_2
1	1	P_3

- Buffer: signal isolation & drive
  - inverter without the inversion





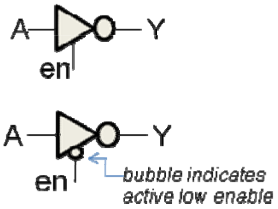
- Tri-State (buffer/inverter)
  - buffer (or inverter) with disable (high impedance) state



en	A	Y
0	0	hi Z
0	1	hi Z
1	0	0
1	1	1

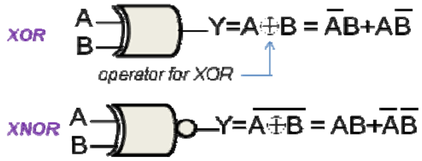
high impedance = open circuit

Y=A when en = 1



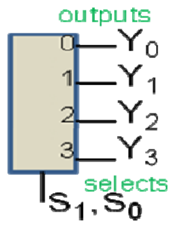
And More…

- XOR & XNOR
  - true of both different (XOR) or both same (XNOR)


XOR
A
B
$Y=A\oplus B=\bar{A}B+A\bar{B}$
operator for XOR
XNOR
A
B
$Y=\overline{A\oplus B}=AB+\bar{A}\bar{B}$

A	B	$A\oplus B$	$\overline{A\oplus B}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

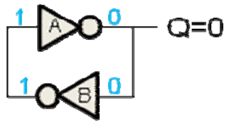
- Decoder
  - only one "active" output
  - example 4:1 decoder
  - difference from MUX?
  - active low concept

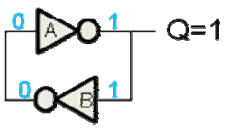


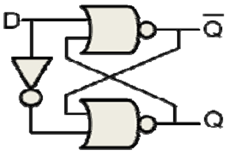
S_1	S_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

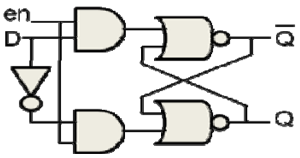
Sequential Logic & Latch

- Sequential logic
  - difference from combinational?
  - output based on input value at prior time
- Latch
  - 2-state storage circuit
    - can "hold" data
  - bi-stable circuit
  - D-type latch
  - D-type latch w/ enable
  - SR latch

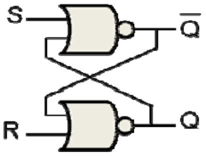








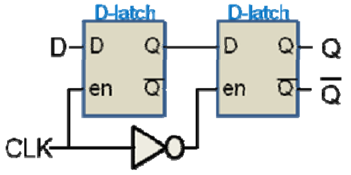
en	D	Q_1
0	0	Q_0
0	1	Q_0
1	0	0
1	1	1



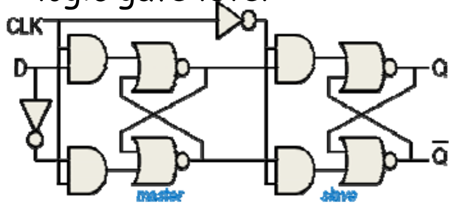
S	R	Action
0	0	$Q_1=Q_0$
0	1	$Q_1=0$
1	0	$Q_1=1$
1	1	*restricted*

Flip Flop

- Flip flop is a synchronous circuit
  - which means?
- D-type master-slave flip flop
  - latch level



logic gate level



- JK flip flop
- Toggle flip flop
- Convertability

D	Q_1
0	0
1	1

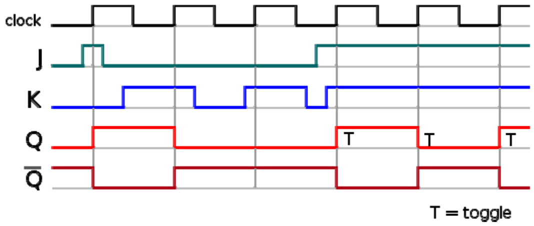
J	K	Q_1
0	0	Q_0
0	1	0
1	0	1
1	1	$\overline{Q}_0$

T	Q_1
0	Q_0
1	$\overline{Q}_0$

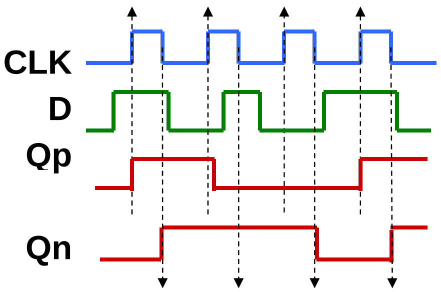
State tables for D-type, JK, and T-type flip flops

Flip Flop Timing Diagrams

- Timing diagram: map of digital output (& input) vs. time
  - use state table to find output
  - look at proper clock transition
  - ignore input changes outside clock transitions

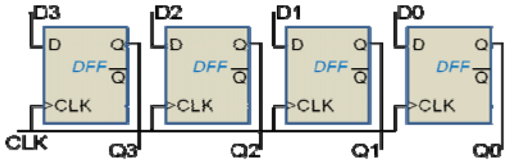


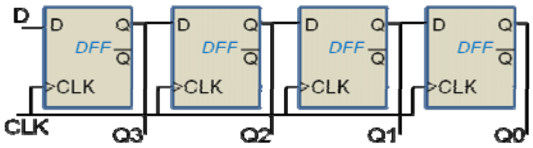
- Rising- vs. falling-edge triggered
  - triggered by clock
  - rising = positive = "up"
  - falling = negative = "down"



Registers

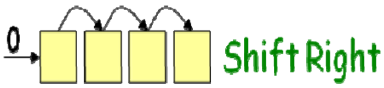
- Data registers
  - stores block (byte/word) of digital data
  - composed of flip flops; used as static memory
  - example: 4b parallel-in parallel-out register
- Shift register
  - can move data laterally between register bits
  - can input or output (or both) data serially
  - example: 4b serial-in parallel out shift register

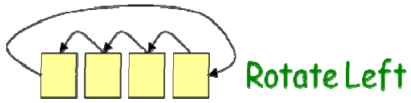




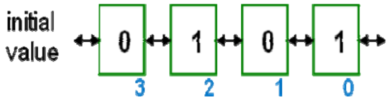
Shift & Rotate

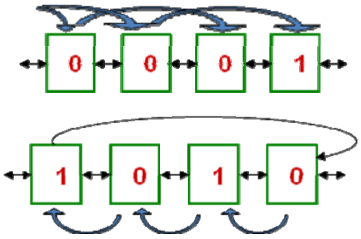
- ***Shift***
  - move each bit (left or right) to adjacent register, load in preset value (normally 0) into open registers
- ***Rotate***
  - move each bit (left or right) to adjacent register, rotate exiting bits back into other side of register





- Example:
  - shift right by 2
  - rotate left by 1



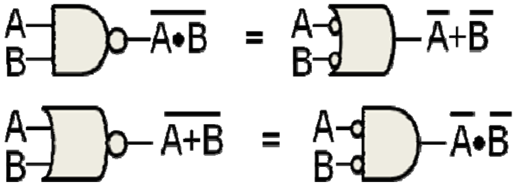


DeMorgans & Complete Set

- DeMorgans Relations

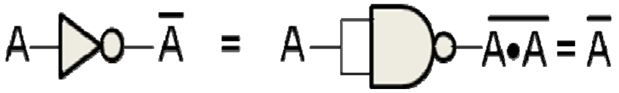
$$\overline{A \bullet B} = \overline{A} + \overline{B}$$

$$\overline{A + B} = \overline{A} \bullet \overline{B}$$



A
B
$\overline{A \bullet B}$
=
A
B
$\overline{A}+\overline{B}$
A
B
$\overline{A+B}$
=
A
B
$\overline{A}\bullet\overline{B}$

- Complete Set Concept
  - all logic functions can be implemented with ONLY NAND (or ONLY NOR)
  - example: INV with NAND

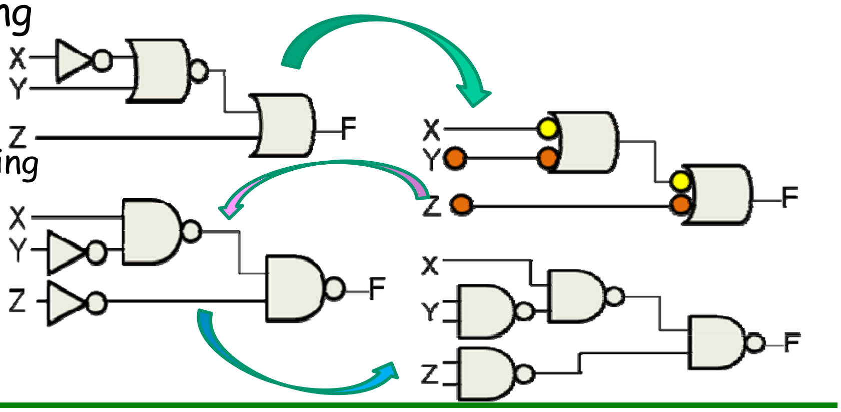


    - similarly, can make AND, OR, NOR with only NAND

Logic Schematic Manipulation

- Bubble pushing technique
  - bubbles can be moved from input of a gate to the output of an attached gate (or from output to input)
  - 2 bubbles can be added to any node (like double negative)

- Example: convert to all NANDs using DeMorgans & bubble pushing
  - initial F
  - bubble pushing
  - DeMorgans
  - final



X
Y
Z
F
X
Y
Z
F
X
Y
Z
F
X
Y
Z
F