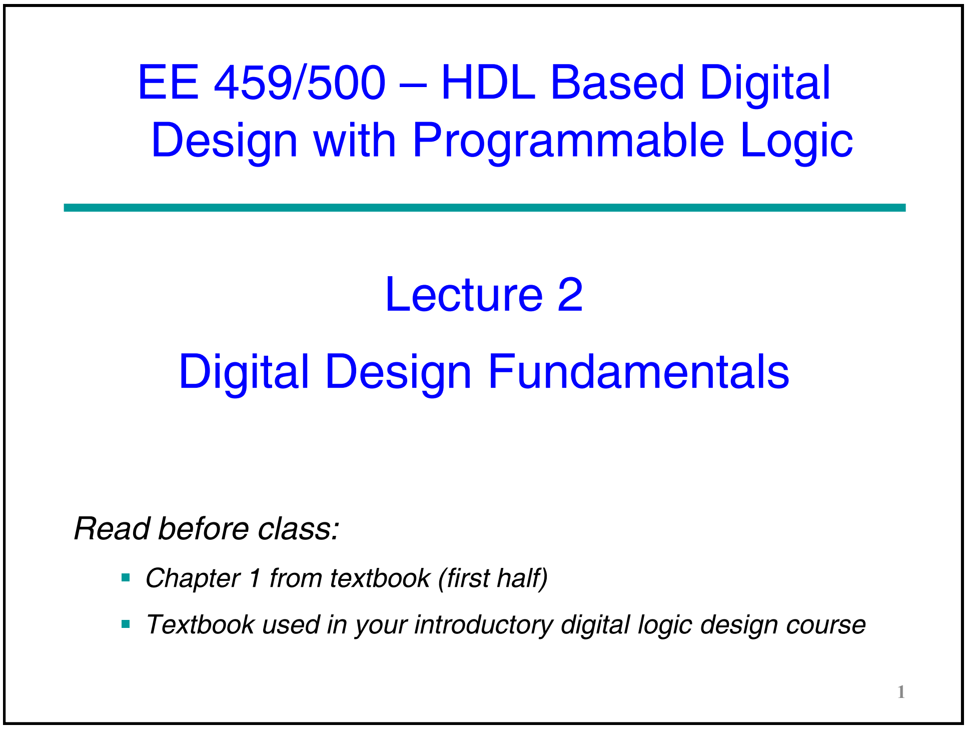


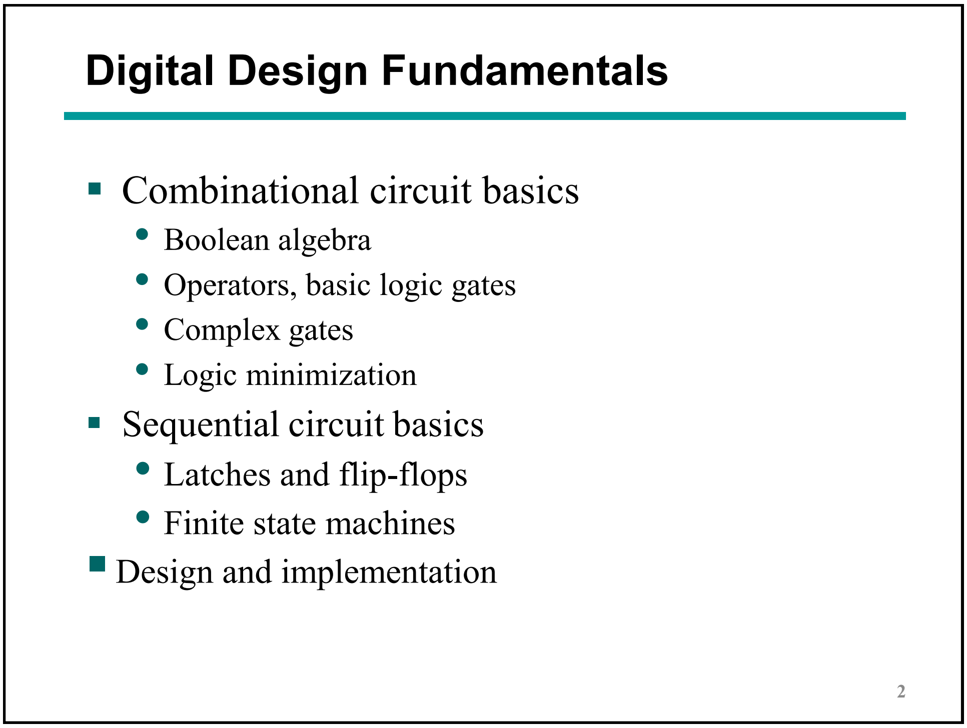
EE 459/500 – HDL Based Digital Design with Programmable Logic

Lecture 2

Digital Design Fundamentals

Read before class:

- *Chapter 1 from textbook (first half)*
- *Textbook used in your introductory digital logic design course*



Digital Design Fundamentals

- Combinational circuit basics
  - Boolean algebra
  - Operators, basic logic gates
  - Complex gates
  - Logic minimization
- Sequential circuit basics
  - Latches and flip-flops
  - Finite state machines
- Design and implementation

Part 1

- Combinational circuit basics

Boolean Algebra

- Unity operators $A + 0 = A$
  $A \cdot 1 = A$
- Complement $A + \overline{A} = 1$
  $A \cdot \overline{A} = 0$
- Commutativity $A + B = B + A$
  $A \cdot B = B \cdot A$
- Associativity $A + (B + C) = (A + B) + C$
  $A \cdot (BC) = (AB) \cdot C$
- **Distributive Law** $A \cdot (B + C) = AB + AC$
  $A + BC = (A + B) \cdot (A + C)$

Boolean Algebra

- Duality $f(A,B,1,0,\cdot,+ = \overline{f(\bar{A},\bar{B},0,1,+\ )}$

$$A+A=A \qquad A\cdot A=A$$

$$1+A=1 \qquad 0\cdot A=0$$

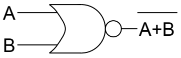
$$A+AB=A \qquad A\cdot(A+B)=A$$

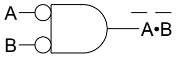
$$A+\bar{A}B=A+B \qquad A\cdot(\bar{A}+B)=A\cdot B$$

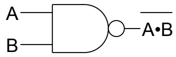
- DeMorgan's Theorem

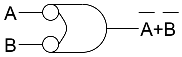
$$\overline{A}+\bar{B}=\bar{A}\overline{B}$$

$$\overline{A}\cdot\overline{B}=\bar{A}+\bar{B}$$





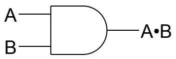




Operators, Basic Logic Gates

- AND

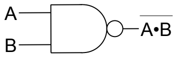
$$f(A,B)=A\cdot B=A\bigcap B$$



A	B	A•B
0	0	0
0	1	0
1	0	0
1	1	1

- NAND

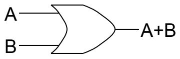
$$f(A,B)=\bar{A}\cdot\bar{B}=\overline{A\bigcap B}$$



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

- OR

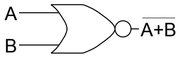
$$f(A,B)=A+B=A\bigcup B$$



A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

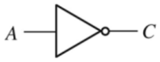
- NOR

$$f(A,B)=\overline{A+B}=\overline{A\bigcup B}$$



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

- NOT

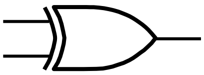


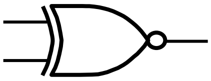
$$C=NOT(A)=A'=\bar{A}$$

Operators, Basic Logic Gates

- Exclusive OR (XOR)/Exclusive NOR (XNOR)

  $X \oplus Y = X\overline{Y} + \overline{X}Y$ $\overline{X \oplus Y} = XY + \overline{X}\,\overline{Y}$



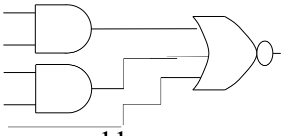


- Uses for the XOR and XNOR gate include:
  - Adders/subtractors/multipliers
  - Counters/incrementers/decrementers
  - Parity generators/checkers
- The XOR function may be implemented
  - directly as an electronic circuit (truly a basic gate)
  - interconnecting other gate types (used as a convenient representation; can be seen as a complex gate)

X	Y	$X \oplus$
0	0	0
0	1	1
1	0	1
1	1	0

Complex Logic Gates

- SOP or POS structures with and without an output inverter.
  - SOP: $F = \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\,\overline{C} + A\overline{B}C + AB\overline{C}$
  - POS: $F = (A + B + C)\cdot(A + \overline{B} + \overline{C})\cdot(\overline{A} + \overline{B} + \overline{C})$
- Naming:
  - A – AND, O – OR, I - Inverter
  - Numbers of inputs on first-level "gates" or directly to second-level "gates" – AOI 221

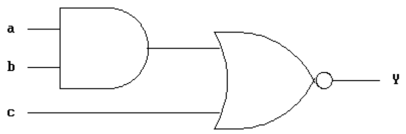


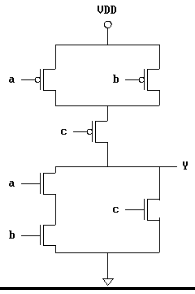
- These gate types are used because:
  - The number of transistors needed is fewer than required by connecting together primitive gates
  - Potentially, the circuit delay is smaller, increasing the circuit operating speed

Complex Logic Gates

- Common forms of two-level complex logic gates:
  - **and-or-invert (AOI)**
  - **or-and-invert (OAI)**

and-or-invert	sum of products	AOI-21 = [AB+C]'	2+1 = 3 inputs
		AOI-231 = [AB+CDE+F]'	2+3+1 = 6 inputs
or-and-invert	product of sums	OAI-21 = [(A+B)(C)]'	2+1 = 3 inputs
		OAI-231 = [(A+B)(C+D+E)(F)]'	2+3+1 = 6 inputs

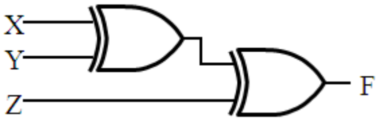




Odd/Even Function

- The XOR function of ≥ 3 variables is called an *odd function* or *modulo 2 sum* (*Mod 2 sum*)

$$X \oplus Y \oplus Z = \overline{X}\,\overline{Y}\,Z + \overline{X}\,Y\,\overline{Z} + X\,\overline{Y}\,\overline{Z} + X\,Y\,Z$$



$$X \oplus 0 = X \qquad X \oplus 1 = \overline{X}$$

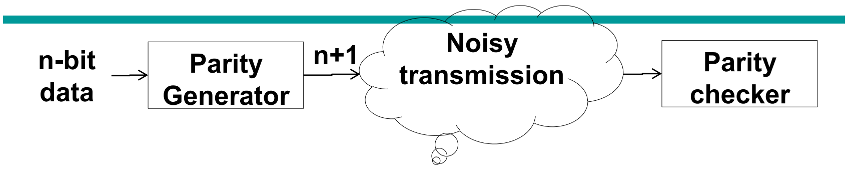
$$X \oplus X = 0 \qquad X \oplus \overline{X} = 1$$

$$X \oplus Y = Y \oplus X$$

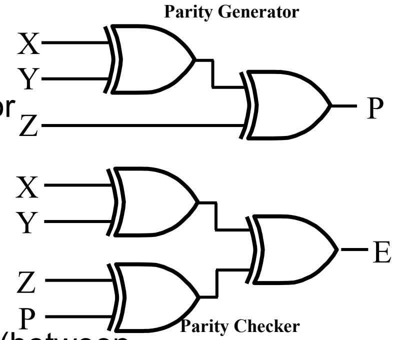
$$(X \oplus Y) \oplus Z = X \oplus (Y \oplus Z) = X \oplus Y \oplus Z$$

- The XNOR function of >3 variables is the *even function*

Parity Generators and Checkers



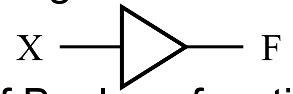
- Example: n = 3
  - Generate a parity code word of length 4 with odd parity generator
  - Check the parity code word of length 4 with odd parity checker:
  - $P=X\oplus Y\oplus Z$, $E=X\oplus Y\oplus Z\oplus P$



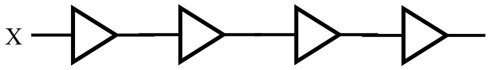
  - If Y changes during transmission (between generator and checker), then E = 1 indicates an error.

Buffer

- A buffer is a gate with the function F = X



- In terms of Boolean function, a buffer is the same as a connection!
- So why use it?
  - A buffer is an electronic amplifier used to improve circuit voltage levels and increase the speed of circuit operation.



Hi-Impedance Outputs

- Logic gates introduced thus far
  - have 1 and 0 output values
  - cannot have their outputs connected together
  - transmit signals on connections in only one direction
- Three-state logic adds a third logic value: Hi-Impedance (Hi-Z)
- The presence of a Hi-Z state makes a gate output as described above behave quite differently:
  - "1 and 0" → "1, 0, and Hi-Z"
  - "cannot" → "can"
  - "only one" → "two"

Hi-Impedance Outputs (Contd.)

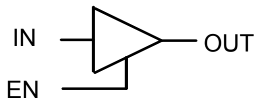
- The Hi-Z value behaves as an open circuit
  - looking back into the circuit, the output appears to be disconnected
- Hi-Z may appear on the output of any gate, but we restrict to gates:
  - a 3-state buffer
  - a transmission gate

  Each of which has one data input and one control input

The 3-State Buffer

- For EN = 0, the OUT is Hi-Z regardless of the value on IN
- For EN = 1, the OUT follows the input value

Symbol

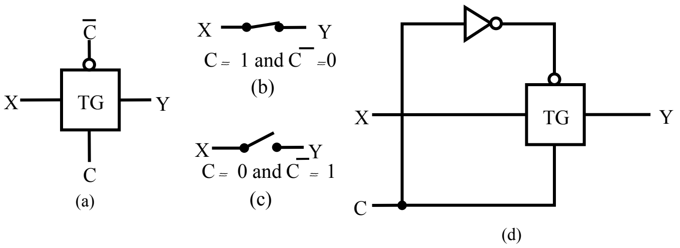


Truth Table

EN	IN	OUT
0	X	Hi-Z
1	0	0
1	1	1

Transmission Gates

- The transmission gate is an electronic switch for connecting and disconnecting two points in a circuit
  - C = 1, Y = X (X = 0 or 1)
  - C = 0, Y = Hi-Z

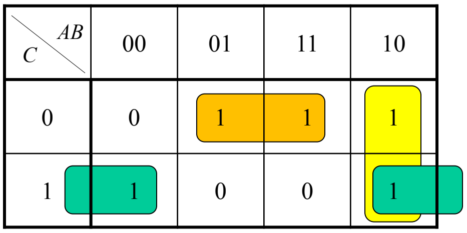


- Since X and Y as input and output are interchangeable, and signals can pass in both directions

Logic Minimization

- Minimizing SOP representation to MSP
- Using Karnaugh Map (K-Map)

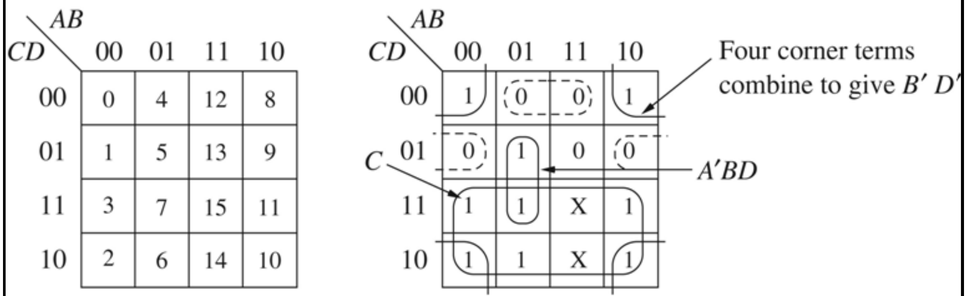
$$F = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + A\bar{B}C + AB\bar{C}$$



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

$$F = B\bar{C} + A\bar{B} + \bar{B}C$$

K-Map Example



CD \ AB	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	3	7	15	11
10	2	6	14	10

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

$$F = \Sigma m\,(0, 2, 3, 5, 6, 7, 8, 10, 11) + \Sigma d\,(14, 15)$$
$$= C + B'\,D' + A'\,BD$$

(a) Location of minterms (b) Looping terms

Logic Minimization

- Minimizing POS representation to MPS
- Using K-Map

$$F=(A+B+C)\cdot\left(A+\overline{B}+\overline{C}\right)\cdot\left(\overline{A}+\overline{B}+\overline{C}\right)$$

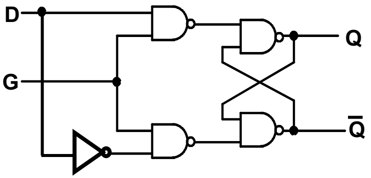
$C\backslash AB$	00	01	11	10
0	0	1	1	1
1	1	0	0	1

$$F=(A+B+C)\cdot\left(\overline{B}+\overline{C}\right)$$

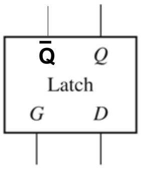
Part 2

- Sequential circuit basics

D-Latch (transparent D-latch)



D latch

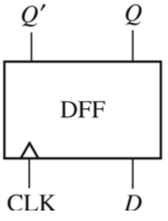


G	D	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

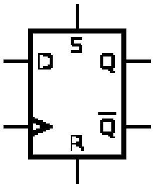
“Latch” is an important concept: input is controlled by a gate input G; when G stays low, the state of the device holds (latches) previous value; when G is high, Q follows D

D Flip-Flop (DFF)

- Rising edge triggered D Flip-Flop
- Timing parameters:
  - Setup time t_{su}: input must be stable before the clock edge
  - Hold time t_h: input must stay stable after the clock edge
  - Clock to Q $t_{c\text{-}q}$: maximum time for output to be stable after the clock edge

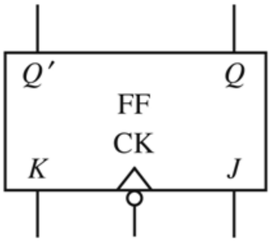


D	Q	Q^+
0	0	0
0	1	0
1	0	1
1	1	1



JK Flip-Flop (JKFF)

- Clocked JK flip-flop
- All state changes occur following the falling edge of the clock, CK, input. This is indicated by the “bubble” at the CK input
- **Exercize**: Use K-Map to derive equation of Q^+



J	K	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Summary

- Boolean algebra essential for digital circuits
- Logic minimization K-Map based
- Be aware of difference in operation between latch and flip-flop