BLM5207
Computer Organization

Prof. Dr. Nizamettin AYDIN

naydin@yildiz.edu.tr

http://www3.yildiz.edu.tr/~naydin

Digital Logic

1

1

Introduction

- In the latter part of the nineteenth century, George Boole incensed philosophers and mathematicians alike when he suggested that logical thought could be represented through mathematical equations.
  - How dare anyone suggest that human thought could be encapsulated and manipulated like an algebraic formula?
- Computers, as we know them today, are implementations of Boole's Laws of Thought.
  - John Atanasoff and Claude Shannon were among the first to see this connection

2

2

Introduction

- In the middle of the twentieth century, computers were commonly known as thinking machines and electronic brains.
  - Many people were fearful of them.
- Nowadays, we rarely ponder the relationship between electronic digital computers and human logic.
- Computers are accepted as part of our lives.
  - Many people, however, are still fearful of them.
- In this lecture, you will learn the simplicity that constitutes the essence of the machine.

3

3

Boolean Algebra

- Boolean algebra is a mathematical system for the manipulation of variables that can have one of two values.
  - In formal logic, these values are true and false.
  - In digital systems, these values are on and off, 1 and 0, or high and low.
- Boolean expressions are created by performing operations on Boolean variables.
  - Common Boolean operators include AND, OR, and NOT.

4

4

Boolean Algebra

- A Boolean operator can be completely described using a truth table.
  - The truth table for the Boolean operators AND, OR, and NOT are shown at the right.
    - The AND operator is also known as a Boolean product.
    - The OR operator is the Boolean sum.
    - The NOT operation is most often designated by an overbar.
      - It is sometimes indicated by a prime mark (″) or an "elbow" (¬).

X AND Y

X	Y	XY
0	0	0
0	1	0
1	0	0
1	1	1

X OR Y

X	Y	X+Y
0	0	0
0	1	1
1	0	1
1	1	1

NOT X

X	$\overline{X}$
0	1
1	0

5

5

Boolean Algebra

- A Boolean function has:
  - At least one Boolean variable,
  - At least one Boolean operator, and
  - At least one input from the set {0,1}.
- It produces an output that is also a member of the set {0,1}.
    - Now you know why the binary numbering system is so handy in digital systems.

6

6

Boolean Algebra

- The truth table for the Boolean function:
  $F(x,y,z) = x\bar{z}+y$

$F(x,y,z) = x\bar{z}+y$

x	y	z	$\bar{z}$	$x\bar{z}$	$x\bar{z}+y$
0	0	0	1	0	0
0	0	1	0	0	0
0	1	0	1	0	1
0	1	1	0	0	1
1	0	0	1	1	1
1	0	1	0	0	0
1	1	0	1	1	1
1	1	1	0	0	1

  – To make evaluation of the Boolean function easier, the truth table contains extra (shaded) columns to hold evaluations of subparts of the function.

7

Boolean Algebra

- As with common arithmetic, Boolean operations have rules of precedence.

$F(x,y,z) = x\bar{z}+y$

x	y	z	$\bar{z}$	$x\bar{z}$	$x\bar{z}+y$
0	0	0	1	0	0
0	0	1	0	0	0
0	1	0	1	0	1
0	1	1	0	0	1
1	0	0	1	1	1
1	0	1	0	0	0
1	1	0	1	1	1
1	1	1	0	0	1

  – The NOT operator has highest priority, followed by AND and then OR.
  – This is how we chose the (shaded) function subparts in our table.

8

Boolean Algebra

- Digital computers contain circuits that implement Boolean functions.
- The simpler that we can make a Boolean function, the smaller the circuit that will result.
  – Simpler circuits are cheaper to build, consume less power, and run faster than complex circuits.
- With this in mind, we always want to reduce our Boolean functions to their simplest form.
- There are a number of Boolean identities that help us to do this.

9

Boolean Algebra

- Most Boolean identities have an AND (product) form as well as an OR (sum) form.
- First group is rather intuitive:

Identity Name	AND Form	OR Form
Identity Law	$1x = x$	$0 + x = x$
Null Law	$0x = 0$	$1 + x = 1$
Idempotent Law	$xx = x$	$x + x = x$
Inverse Law	$x\bar{x} = 0$	$x + \bar{x} = 1$

10

Boolean Algebra

- Second group of Boolean identities:

Identity Name	AND Form	OR Form
Commutative Law	$xy = yx$	$x+y = y+x$
Associative Law	$(xy)z = x(yz)$	$(x+y)+z = x + (y+z)$
Distributive Law	$x+yz = (x+y)(x+z)$	$x(y+z) = xy+xz$

- Thirdgroup of Boolean identities:

Identity Name	AND Form	OR Form
Absorption Law	$x(x+y) = x$	$x + xy = x$
DeMorgan's Law	$\overline{(xy)} = \bar{x} + \bar{y}$	$\overline{(x+y)} = \bar{x}\bar{y}$
Double Complement Law	$\overline{(\bar{x})} = x$	

11

Boolean Algebra

- We can use Boolean identities to simplify the function $F(X,Y,Z) = (X + Y)(X + \bar{Y})\overline{(X\bar{Z})}$ as follows:

$(X + Y)(X + \bar{Y})\overline{(X\bar{Z})}$	Idempotent Law (Rewriting)
$(X + Y)(X + \bar{Y})(\bar{X} + Z)$	DeMorgan's Law
$(XX + X\bar{Y} + XY + Y\bar{Y})(\bar{X} + Z)$	Distributive Law
$((X + Y\bar{Y}) + X(Y + \bar{Y}))(\bar{X} + Z)$	Commutative & Distributive Laws
$((X + 0) + X(1))(\bar{X} + Z)$	Inverse Law
$X(\bar{X} + Z)$	Idempotent Law
$X\bar{X} + XZ$	Distributive Law
$0 + XZ$	Inverse Law
XZ	Idempotent Law

12

Boolean Algebra

- Sometimes it is more economical to build a circuit using the complement of a function (and complementing its result) than it is to implement the function directly.
- DeMorgan's law provides an easy way of finding the complement of a Boolean function.
- Recall DeMorgan's law states:

$$\overline{(xy)} = \bar{x} + \bar{y} \quad \text{and} \quad \overline{(x+y)} = \bar{x}\bar{y}$$

13

13

Boolean Algebra

- DeMorgan's law can be extended to any number of variables.
- Replace each variable by its complement and change all ANDs to ORs and all ORs to ANDs.
- Thus, we find the complement of:

$$F(X,Y,Z) = (XY) + (\bar{X}Z) + (Y\bar{Z})$$

as:

$$\begin{aligned}\bar{F}(X,Y,Z) &= \overline{(XY) + (\bar{X}Z) + (Y\bar{Z})} \\ &= \overline{(XY)}\,\overline{(\bar{X}Z)}\,\overline{(Y\bar{Z})} \\ &= (\bar{X}+\bar{Y})(X+\bar{Z})(\bar{Y}+Z)\end{aligned}$$

14

14

Boolean Algebra

- Through our exercises in simplifying Boolean expressions, we see that there are numerous ways of stating the same Boolean expression.
  - These synonymous forms are logically equivalent.
  - Logically equivalent expressions have identical truth tables.
- In order to eliminate as much confusion as possible, designers express Boolean functions in standardized or canonical form.

15

15

Boolean Algebra

- There are two canonical forms for Boolean expressions:
  - sum-of-products
  - product-of-sums
    - Recall the Boolean product is the AND operation and the Boolean sum is the OR operation.
- In the sum-of-products form, ANDed variables are ORed together.
  - For example: $F(x,y,z) = xy + xz + yz$
- In the product-of-sums form, ORed variables are ANDed together.
  - For example: $F(x,y,z) = (x+y)(x+z)(y+z)$

16

16

Boolean Algebra

- It is easy to convert a function to sum-of-products form using its truth table.

$F(x,y,z) = x\bar{z}+y$

x	y	z	$x\bar{z}+y$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

  - We are interested in the values of the variables that make the function true (=1).
  - Using the truth table, we list the values of the variables that result in a true function value.
  - Each group of variables is then ORed together.

17

17

Boolean Algebra

- The sum-of-products form for our function is:

$$F(x,y,z) = \bar{x}y\bar{z}+\bar{x}yz+x\bar{y}\bar{z}+xy\bar{z}+xyz$$

$F(x,y,z) = x\bar{z}+y$

x	y	z	$x\bar{z}+y$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

  - We note that this function is not in simplest terms.
  - Our aim is only to rewrite our function in canonical sum-of-products form.

18

18

Logic Gates

- We have looked at Boolean functions in abstract terms.
- In this section, we see that Boolean functions are implemented in digital computer circuits called gates.
- A gate is an electronic device that produces a result based on two or more input values.
  - In reality, gates consist of one to six transistors, but digital designers think of them as a single unit.
  - Integrated circuits contain collections of gates suited to a particular purpose.

19

19

Logic Gates

- The three simplest gates are the AND, OR, and NOT gates.

X AND Y

X	Y	XY
0	0	0
0	1	0
1	0	0
1	1	1

X OR Y

X	Y	X+Y
0	0	0
0	1	1
1	0	1
1	1	1

NOT X

X	$\overline{X}$
0	1
1	0

- They correspond directly to their respective Boolean operations, as you can see by their truth tables.

20

20

Logic Gates

- Another very useful gate is the exclusive OR (XOR) gate.
  - The output of the XOR operation is true only when the values of the inputs differ.

X XOR Y

X	Y	X ⊕ Y
0	0	0
0	1	1
1	0	1
1	1	0

  - Note the special symbol ⊕ for the XOR operation.

21

21

Logic Gates

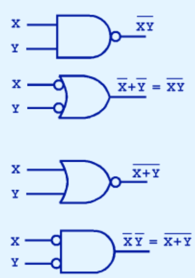
- NAND and NOR are two very important gates.

X NAND Y

X	Y	X NAND Y
0	0	1
0	1	1
1	0	1
1	1	0

X NOR Y

X	Y	X NOR Y
0	0	1
0	1	0
1	0	0
1	1	0

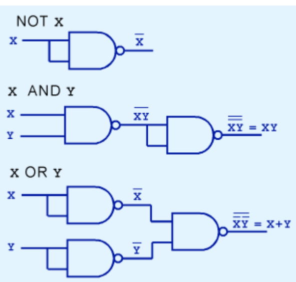


22

22

Logic Gates

- NAND and NOR are known as universal gates because they are inexpensive to manufacture, and any Boolean function can be constructed using only NAND or only NOR gates.

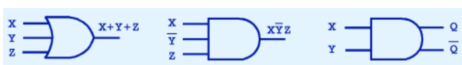


23

23

Logic Gates

- Gates can have multiple inputs and more than one output.
- A second output can be provided for the complement of the operation.

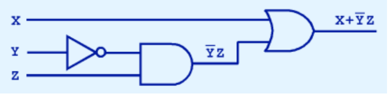


24

24

Digital Components

- The main thing to remember is that combinations of gates implement Boolean functions.
- The circuit below implements the Boolean function: $F(X,Y,Z) = X+\overline{Y}Z$



– We simplify our Boolean expressions so that we can create simpler circuits.

25

25

Combinational Circuits

- We have designed a circuit that implements the Boolean function:

$$F(X,Y,Z) = X+\overline{Y}Z$$

- This circuit is an example of a combinational logic circuit.
- Combinational logic circuits produce a specified output (almost) at the instant when input values are applied.

26

26

Combinational Circuits

- Combinational logic circuits give us many useful devices.

Inputs		Outputs	
X	Y	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

- One of the simplest is the half adder, which finds the sum of two bits.
- We can gain some insight as to the construction of a half adder by looking at its truth table

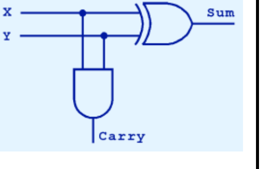
27

27

Combinational Circuits

- As we see, the sum can be found using the XOR operation and the carry using the AND operation.

Inputs		Outputs	
X	Y	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



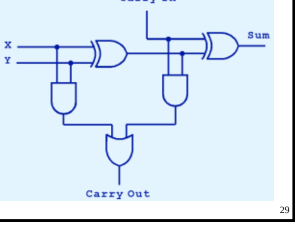
28

28

Combinational Circuits

- We can change our half adder into to a full adder by including gates for processing the carry bit.
- The truth table for a full adder and its implementation:

Inputs			Outputs	
X	Y	Carry In	Sum	Carry Out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

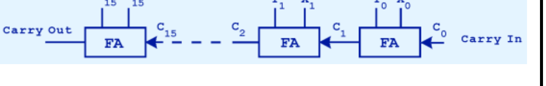


29

29

Combinational Circuits

- Just as we combined half adders to make a full adder, full adders can be connected in series.
- The carry bit ripples from one adder to the next; hence, this configuration is called a ripple-carry adder.



– Today's systems employ more efficient adders.

30

30

Combinational Circuits

- Decoders are another important type of combinational circuit.
  - Among other things, they are useful in selecting a memory location according a binary value placed on the address lines of a memory bus.

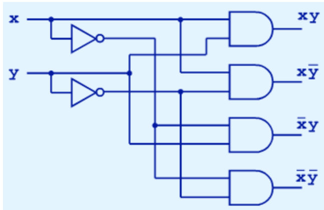
- Address decoders with n inputs can select any of 2^n locations.

31

31

Combinational Circuits

- This is what a 2-to-4 decoder looks like on the inside.



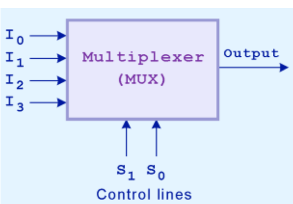
  - If x = 0 and y = 1, which output line is enabled?

32

32

Combinational Circuits

- A multiplexer does just the opposite of a decoder.
- It selects a single output from several inputs.



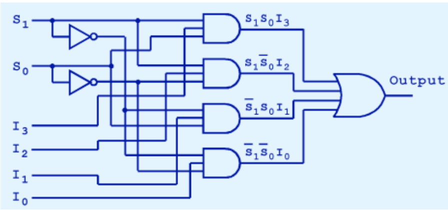
  - The particular input chosen for output is determined by the value of the multiplexer's control lines.
  - To be able to select among n inputs, $\log_2 n$ control lines are needed.

33

33

Combinational Circuits

- This is what a 4-to-1 multiplexer looks like on the inside.



  - If $S_0 = 1$ and $S_1 = 0$, which input is transferred to the output?

34

34

Sequential Circuits

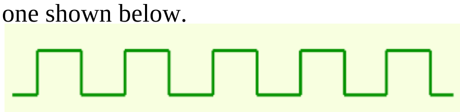
- Combinational logic circuits are perfect for situations when we require the immediate application of a Boolean function to a set of inputs.
- There are other times, however, when we need a circuit to change its value with consideration to its current state as well as its inputs.
  - These circuits have to remember their current state.
- Sequential logic circuits provide this functionality for us.

35

35

Sequential Circuits

- As the name implies, sequential logic circuits require a means by which events can be sequenced.
- State changes are controlled by clocks.
  - A clock is a special circuit that sends electrical pulses through a circuit.
- Clocks produce electrical waveforms such as the one shown below.

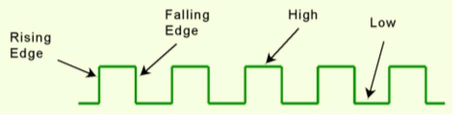


36

36

Sequential Circuits

- State changes occur in sequential circuits only when the clock ticks.
- Circuits can change state on the rising edge, falling edge, or when the clock pulse reaches its highest voltage.

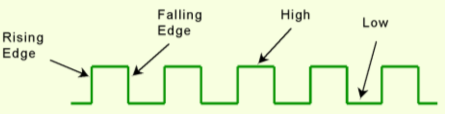


37

37

Sequential Circuits

- Circuits that change state on the rising edge, or falling edge of the clock pulse are called edge-triggered.
- Level-triggered circuits change state when the clock voltage reaches its highest or lowest level.

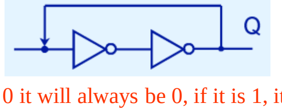


38

38

Sequential Circuits

- To retain their state values, sequential circuits rely on feedback.
- Feedback in digital circuits occurs when an output is looped back to the input.
- A simple example of this concept is shown below.



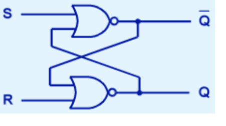
  - If Q is 0 it will always be 0, if it is 1, it will always be 1. Why?

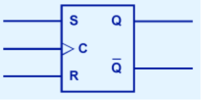
39

39

Sequential Circuits

- You can see how feedback works by examining the most basic sequential logic components, the SR flip-flop.
  - The SR stands for set/reset.
- The internals of an SR flip-flop are shown below, along with its block diagram.



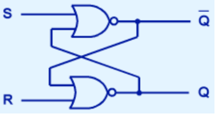


40

40

Sequential Circuits

- The behavior of an SR flip-flop is described by a characteristic table.



S	R	Q(t+1)
0	0	Q(t) (no change)
0	1	0 (reset to 0)
1	0	1 (set to 1)
1	1	undefined

  - Q(t) means the value of the output at time t.
  - Q(t+1) is the value of Q after the next clock pulse.

41

41

Sequential Circuits

- The SR flip-flop actually has three inputs:
  - S, R, and its current output, Q.

Present State			Next State
S	R	Q(t)	Q(t+1)
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	undefined
1	1	1	undefined

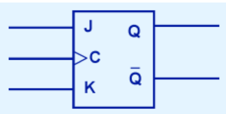
- Thus, we can construct a truth table for this circuit, as shown at the left.
- Notice the two undefined values.
  - When both S and R are 1, the SR flip-flop is unstable.

42

42

Sequential Circuits

- If we can be sure that the inputs to an SR flip-flop will never both be 1, we will never have an unstable circuit. This may not always be the case.
- The SR flip-flop can be modified to provide a stable state when both inputs are 1.



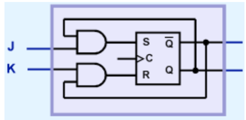
  – This modified flip-flop is called a JK flip-flop, shown at the left.
    - The JK is in honor of Jack Kilby.

43

43

Sequential Circuits

- At the left, we see how an SR flip-flop can be modified to create a JK flip-flop.
- The characteristic table indicates that the flip-flop is stable for all inputs.



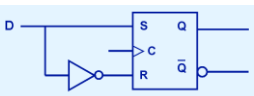
J	K	Q(t+1)
0	0	Q(t) (no change)
0	1	0 (reset to 0)
1	0	1 (set to 1)
1	1	Q̄(t)

44

44

Sequential Circuits

- Another modification of the SR flip-flop is the D flip-flop, shown below with its characteristic table.
  – You will notice that the output of the flip-flop remains the same during subsequent clock pulses.
  – The output changes only when the value of D changes.



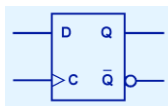
D	Q(t+1)
0	0
1	1

45

45

Sequential Circuits

- The D flip-flop is the fundamental circuit of computer memory.
  – D flip-flops are usually illustrated using the block diagram shown below.
- The next slide shows how these circuits are combined to create a register.



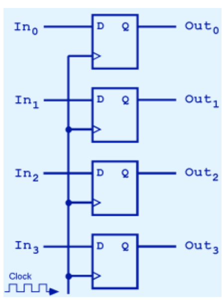
D	Q(t+1)
0	0
1	1

46

46

Sequential Circuits

- This illustration shows a 4-bit register consisting of D flip-flops.
- You will usually see its block diagram (below) instead.

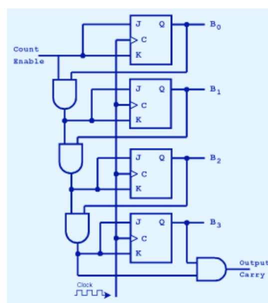


47

47

Sequential Circuits

- A binary counter is another example of a sequential circuit.
- The low-order bit is complemented at each clock pulse.
  – Whenever it changes from 0 to 1, the next bit is complemented, and so on through the other flip-flops.



48

48

Designing Circuits

- We have seen digital circuits from two points of view:
  - Digital analysis
    - explores the relationship between a circuits inputs and its outputs.
  - Digital synthesis
    - creates logic diagrams using the values specified in a truth table.
- Digital systems designers must also be mindful of the physical behaviors of circuits to include minute propagation delays that occur between the time when a circuit's inputs are energized and when the output is accurate and stable.

49

49

Designing Circuits

- Digital designers rely on specialized software to create efficient circuits.
  - Thus, software is an enabler for the construction of better hardware.
- Of course, software is in reality a collection of algorithms that could just as well be implemented in hardware.
  - Recall the Principle of Equivalence of Hardware and Software.

50

50

Designing Circuits

- When we need to implement a simple, specialized algorithm and its execution speed must be as fast as possible, a hardware solution is often preferred.
- This is the idea behind embedded systems, which are small special-purpose computers that we find in many everyday things.
- Embedded systems require special programming that demands an understanding of the operation of digital circuits, the basics of which you have learned in this chapter.

51

51

Conclusion

- Computers are implementations of Boolean logic.
- Boolean functions are completely described by truth tables.
- Logic gates are small circuits that implement Boolean operators.
- The basic gates are AND, OR, and NOT.
- The XOR gate is very useful in parity checkers and adders.
- The "universal gates" are NOR, and NAND.

52

52

Conclusion

- Computer circuits consist of combinational logic circuits and sequential logic circuits.
- Combinational circuits produce outputs (almost) immediately when their inputs change.
- Sequential circuits require clocks to control their changes of state.
- The basic sequential circuit unit is the flip-flop:
  - The behaviors of the SR, JK, and D flip-flops are the most important to know.

53

53