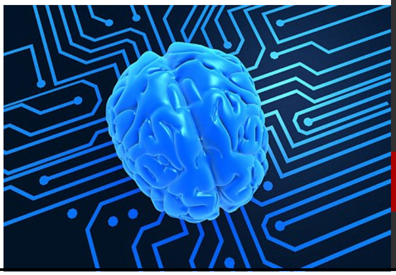
Tutorial-2: Floating point and Computer Logic

AbdulWahab Kabani
PhD Student – Computer Science

1

Why is this topic important?

- Understand Computer Circuits will help us understand how computers perform operations
- Build foundation for the next chapters



2

Outline

- 32 IEEE Floating Point
- Computer Logic
  - Combinational Circuits vs. Sequential Circuits
  - Gates
  - Combinational Circuits
  - Sequential Circuits
- Conclusion

3

Outline

- **32 IEEE Floating Point**
- Computer Logic
  - Combinational Circuits vs. Sequential Circuits
  - Gates
  - Combinational Circuits
  - Sequential Circuits
- Conclusion

4

IEEE Floating-Point (32 bit)

S

+/-

E

127 + e

For example:

e=10 → 127 + 10 = 137
e = 5 → 127 + 5 = 132
e =-5 → 127 - 5 = 122

F

Only bits **AFTER** decimal
For example:

1.**01010101000000000000000**

1.**11011101001000000000000**

5

IEEE Floating-Point (32 bit)

0

- **2200.25$_{10}$**
- Converting 2200.25$_{10}$ to 32 IEEE floating point format
  - Sign: the number is positive → 0
  - Break down the number 2200.25$_{10}$:
    - 2200$_{10}$ = 100010011000$_2$
    - 0.25$_{10}$ = 0.01$_2$

6

IEEE Floating-Point (32 bit)

- **2200$_{10}$ = 100010011000$_2$**
  - 2200/2 = 1100 R = 0
  - 1100/2 = 550 R = 0
  - 550/2 = 275 R = 0
  - 275/2 = 137 R = 1
  - 137/2 = 68 R = 1
  - 68/2 = 34 R = 0
  - 34/2 = 17 R = 0
  - 17/2 = 8 R = 1
  - 8 / 2 = 4 R = 0
  - 4/2 = 2 R = 0
  - 2/2 = 1 R = 0
  - 1/2 = 0 R = 1

7

IEEE Floating-Point (32 bit)

- 0.25$_{10}$ = 0.01$_2$
  - 0.25 *2 = **0**.5 WP = 0
  - 0.5 *2 = **1**.0 WP = 1

8

IEEE Floating-Point (32 bit)

0

- **2200.25$_{10}$**
- Converting 2200.25$_{10}$ to 32 IEEE floating point format
  - Sign: the number is positive → 0
  - Break down the number 2200.25$_{10}$:
    - 2200$_{10}$ = 100010011000$_2$
    - 0.25$_{10}$ = 0.01$_2$
  - Combine Both numbers:
    - 100010011000. 01$_2$

9

IEEE Floating-Point (32 bit)

- 100010011000. 01

11 10 9 8 7 6 5 4 3 2 1

- Unbiased exponent = 11
- Remember our bias: 127
- 127+11 = 138

10

IEEE Floating-Point (32 bit)

0 1 0 0 0 1 0 1 0

- **2200.25$_{10}$**
- Converting 2200.25$_{10}$ to 32 IEEE floating point format
  - Sign: the number is positive → 0
  - Break down the number 2200.25$_{10}$:
    - 2200$_{10}$ = 100010011000$_2$
    - 0.25$_{10}$ = 0.01$_2$
  - Combine Both numbers:
    - 100010011000. 01
  - 138 in binary is 10001010

11

IEEE Floating-Point (32 bit)

0 1 0 0 0 1 0 1 0 0 0 0 1 0 0 1 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0

- **2200.25$_{10}$**
- Converting 2200.25$_{10}$ to 32 IEEE floating point format
  - Sign: the number is positive → 0
  - Break down the number 2200.25$_{10}$:
    - 2200$_{10}$ = 100010011000$_2$
    - 0.25$_{10}$ = 0.01$_2$
  - Combine Both numbers:
    - 100010011000. 01
  - 138 in binary is 10001010
  - Don't store the 1 at the beginning of **1**.0001001100001

12

IEEE Floating-Point (32 bit)

- $0x7F400000_{16}$
- 7 F 4 0 0 0 0 0_{16}
- 0111 1111 0100 0000 0000 0000 0000 0000

13

IEEE Floating-Point (32 bit)

0

- 0x7F400000
- 7 F 4 0 0 0 0 0_{16}
- 0111 1111 0100 0000 0000 0000 0000 0000

14

IEEE Floating-Point (32 bit)

0 | 1 1 1 1 1 1 1 0

- 0x7F400000
- 7 F 4 0 0 0 0 0_{16}
- 0111 1111 0100 0000 0000 0000 0000 0000

15

IEEE Floating-Point (32 bit)

0 | 1 1 1 1 1 1 1 0 | 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

- 0x7F400000
- 7 F 4 0 0 0 0 0_{16}
- **0111 1111 0100 0000 0000 0000 0000 0000**

16

IEEE Floating-Point (32 bit)

0 | 1 1 1 1 1 1 1 0 | 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

+

254 → 2^{127}

1.1_2 → 1.5_{10}

IEEE Floating-Point (32 bit)

0 | 1 1 1 1 1 1 1 0 | 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

- The number is: $+1.5X2^{127}$
- $2^{127} = 10^x$
- Solve for x by taking the log ➔ X = 38.23081
- $2^{127} = 10^x = 10^{38.23081}$
- $= 10^{38} X\ 10^{0.23081} = 10^{38} X\ 1.7014$
- $+1.5\ X \boxed{2^{127}} = 1.5X \boxed{10^{38} X\ 1.7014} = +\ 2.5521\ X\ 10^{38}$

Outline

- **32 IEEE Floating Point**
- Computer Logic
  - **Combinational Circuits vs. Sequential Circuits**
  - Gates
  - Combinational Circuits
  - Sequential Circuits
- Conclusion

Combinational vs. Sequential

- **Combinational** :
  - Gates are the building blocks
  - Output depends **only** on its current inputs

Combinational vs. Sequential

- **Sequential:**
  - Flip-Flops and latches are the building blocks
  - Output depends on current inputs **+ Current State**

21

Combinational vs. Sequential

	Combinational	Sequential
Building Block	Gates	Flip-Flops and latches (set of gates)
Depends on	Input	Input + Current State

22

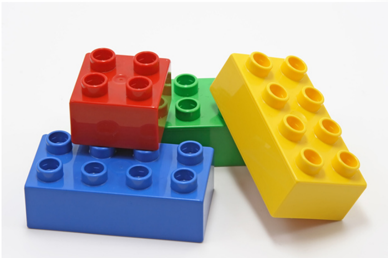
Outline

- **32 IEEE Floating Point**
- Computer Logic
  - **Combinational Circuits vs. Sequential Circuits**
  - **Gates**
  - Combinational Circuits
  - Sequential Circuits
- Conclusion

23

Gates

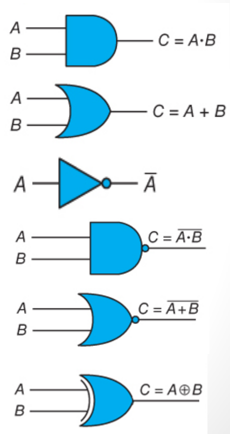
- Our Building Blocks
- Also, used to build flip-flops
- Come in many flavors:
  - AND
  - OR
  - NOT
  - NAND (Not AND)
  - NOR (Not OR)
  - XOR



24

Gates

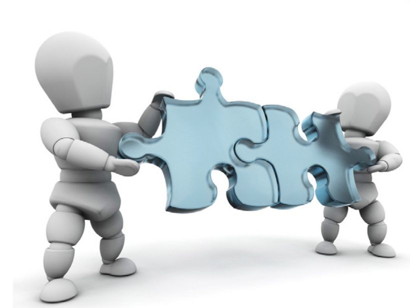
- AND:
  - True only if both input are True
- OR:
  - True if at least one input is True
- NOT:
  - inverts the sign
- NAND:
  - The Opposite of an AND gate
- NOR:
  - The Opposite of an OR gate
- XOR:
  - True only if the inputs are different



25

Gates

- Time to use our blocks to construct digital circuits!



26

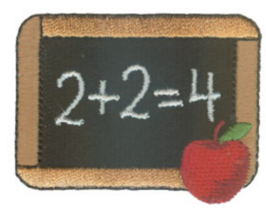
Outline

- **32 IEEE Floating Point**
- Computer Logic
  - **Combinational Circuits vs. Sequential Circuits**
  - **Gates**
  - **Combinational Circuits**
  - Sequential Circuits
- Conclusion

27

Addition Circuit

- Let's construct a circuit that will perform addition

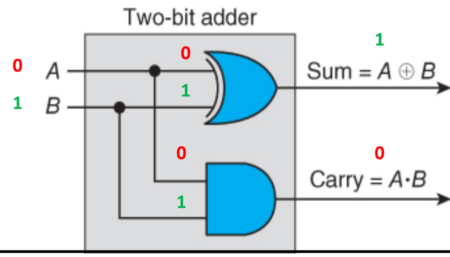


28

Half Adder

- A circuit that adds 1 bit to another
- All possible input/output:
  - 0+0=0
  - 0+1=1
  - 1+0=1
  - 1+1=0 **(with a carry of 1)**

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

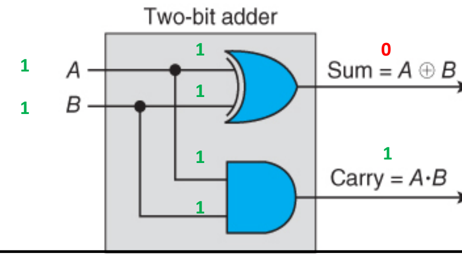


29

Half Adder

- A circuit that adds 1 bit to another
- All possible input/output:
  - 0+0=0
  - 0+1=1
  - 1+0=1
  - 1+1=0 **(with a carry of 1)**

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



30

Half Adder

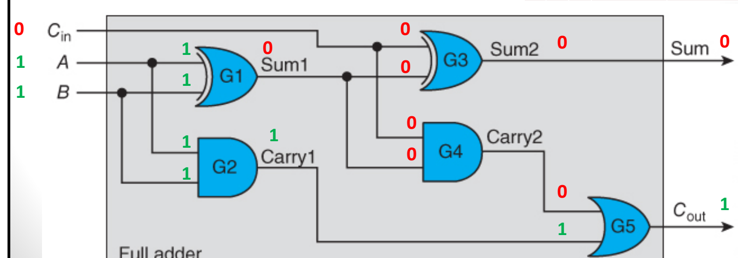
- What if we want to add three bits?
  - Use a Full Adder

31

Full Adder

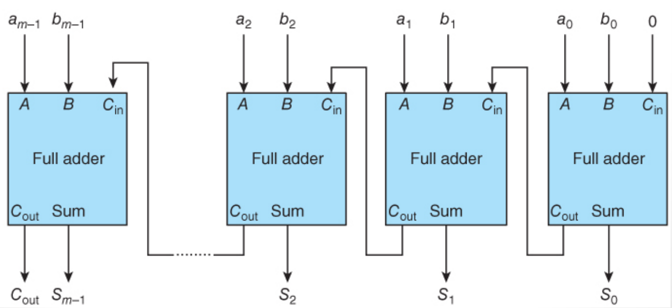
- **2 Half-Adders** combined with an **OR gate**

A	B	C	Sum	Carry
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



32

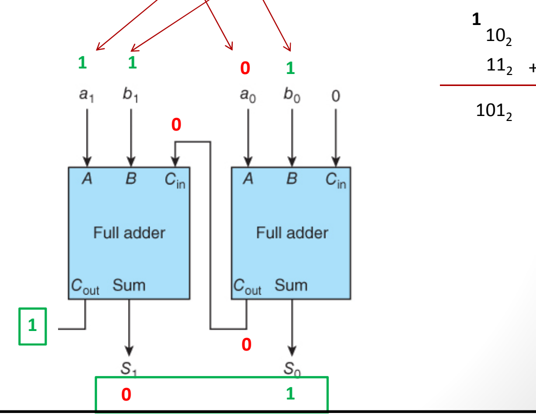
Full Adder



33

Full Adder

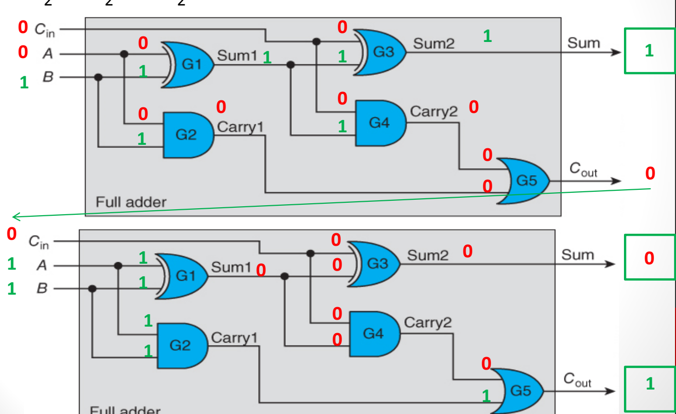
- Say we have want to add $10_2 + 11_2 = 2 + 3 = 5 = 101_2$



34

Full Adder

- $10_2 + 11_2 = 101_2$



35

Full Adder/Subtractor

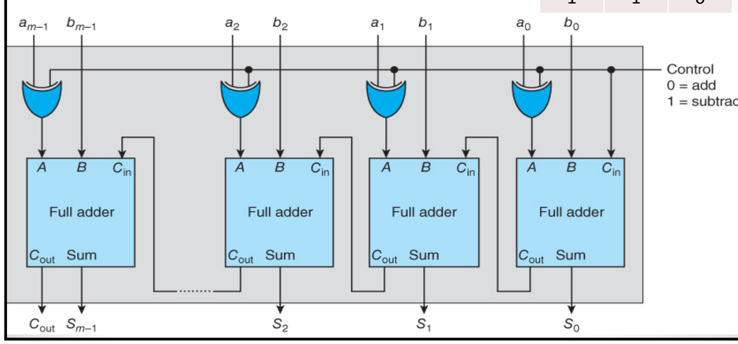
- How can we subtract?
  - Recall Two's complement
  - Subtraction is addition with the two's complement
  - Example: $011_2 - 010_2 = 3 - 2 = 1$
    - N = 3 bits
    - Get the Two's complement
    - $100_2 - 010_2 = 110_2 = -2_{10}$
    - $011_2 + 110_2 = 001_2 = 1$

36

Full Adder/Subtractor

- Why Does this work?
  - Let's review the truth table of the XOR

A	B	XOR
0	0	0
0	1	1
1	0	1
1	1	0

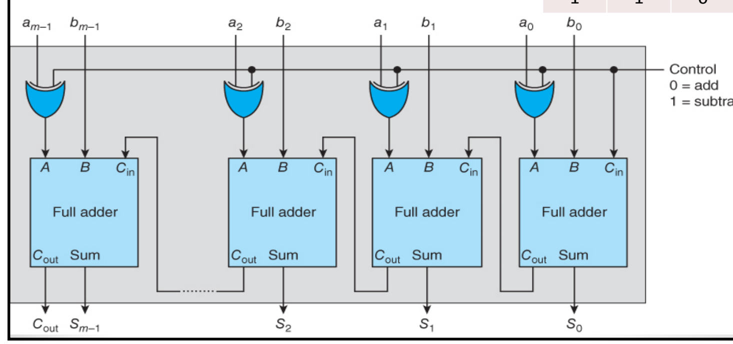


37

Full Adder/Subtractor

- Bit a_i will be flipped if the control is 1

A	B	XOR
0	0	0
0	1	1
1	0	1
1	1	0



38

Decoder

- Say we want a device that does a specific task if the user enters a certain code
- Something like a remote control. You enter a code and go to channel. Few inputs, lots of output (channels)
- Or maybe, Bar code: few input, lots of products
- Computer Instructions

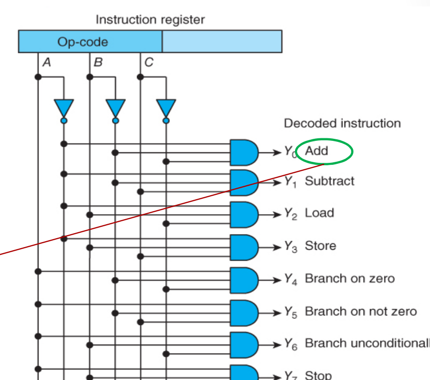
123456

39

Decoder

- In all cases, you go from a small set of possible inputs to a large set of possible outputs.
- Example: 3-8 Decoder
  - 3 input channels
  - 8 possible outputs
- 3 input → 2^3 outputs

You can put a set of full adders here to create an addition circuit



40

Outline

- **32 IEEE Floating Point**
- Computer Logic
  - **Combinational Circuits vs. Sequential Circuits**
  - **Gates**
  - **Combinational Circuits**
  - **Sequential Circuits**
- Conclusion

Sequential Circuits

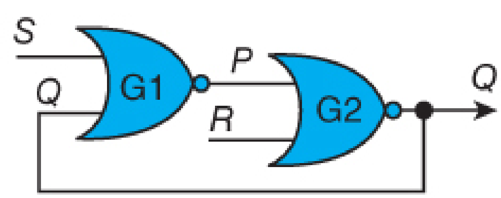
- Flip-Flops and latches are the building blocks
- Output depends on current inputs **+ Current State**

Sequential Circuits

- Three basic types:
  - RS latches
  - D flip-flops
  - JK flip-flops
- These types are the building blocks of sequential Circuits
- What are they composed of?

RS latches

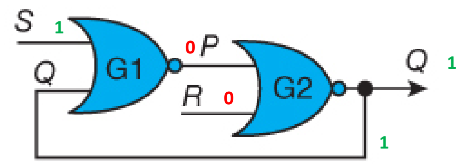
- Convention:
  - Q is the current state
  - Q^+ is the next (new) state
  - R is reset
  - S is set
- This circuit employs *feedback*
  - The value of p is determined by Q (current state)



RS latches

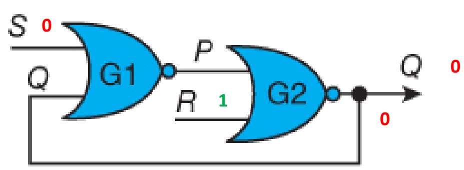
- In Sequential Circuits, it is useful to assume an initial state
- Don't try to think how the previous state was generated

RS latches



Inputs			Output
R	S	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	?
1	1	1	?

RS latches



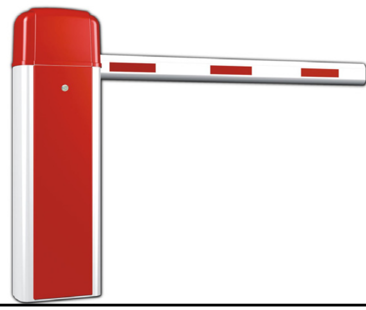
Inputs			Output
R	S	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	?
1	1	1	?

RS latches

Inputs		Output	Description
R	S	Q^+	
0	0	Q	No change
0	1	1	Set output to 1
1	0	0	Reset output to 0
1	1	X	Forbidden

RS latches

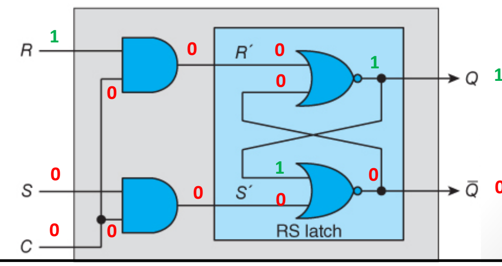
- How can we control **when** to set/reset an RS latch?
  - Use a clock
  - If the clock is **0**, nothing happens to the latch
  - If the clock is **1**, the latch can be set/reset/maintain its state



49

Clocked RS Flip-Flop

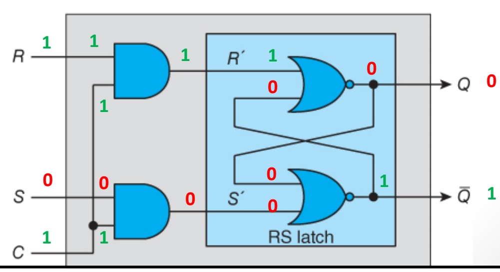
- When clock = 0, nothing happens to the latch
- Assume Q = 1



50

Clocked RS Flip-Flop

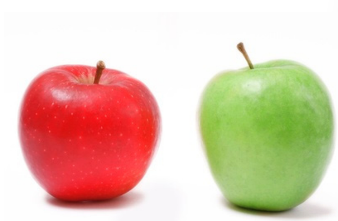
- When clock = 1, nothing happens to the latch
- Assume Q = 1



51

What is the difference between a latch and flip-flop?

- Both are bi-stable
- Flip-Flops are **clocked**



52

D flip-flops

- Similar to the clocked RS flip-flops
- However, no Reset/set Inputs
- Inputs:
  - D: Data
  - Clock
- Similar to a dam

53

D flip-flops

- If clock (C) is 0, no data (D) flows
- If clock (C) is 1, data (D) flows through the flipflop
- Unlike dams, the flow is either 0, or 1 (no quantity concept)

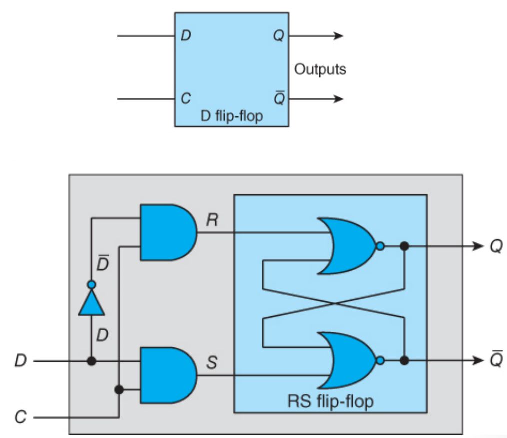
54

D flip-flops

- How to prepare a D-Flip-Flops?
- You need:
  - RS flip-flop
  - 2 AND gates
  - D (Data) and its opposite (inverter)

55

D flip-flops



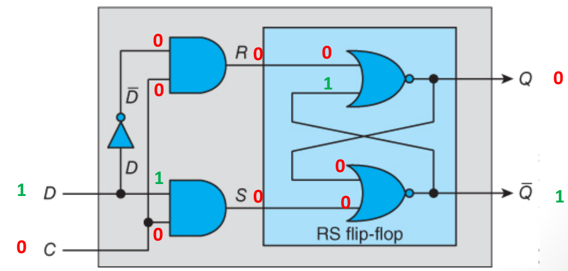
56

D flip-flops

Inputs		Output	Description
C	D	Q^+	
0	0	Q	No change
0	1	Q	No change
1	0	0	D is copied to output
1	1	1	D is copied to output

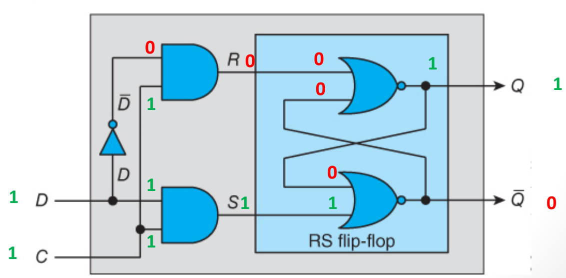
D flip-flops

- When clock is 0, the state remain the state
- Assume $Q_{initial} = 0$



D flip-flops

- When clock is 1, the state changes and have the input of D
- Assume $Q_{initial} = 0$



JK Flip-Flop

Inputs		Output	Description
R	S	Q^+	
0	0	Q	No change
0	1	1	Set output to 1
1	0	0	Reset output to 0
1	1	X	Forbidden

Inputs		Output	Description
K	J	Q^+	
0	0	Q	No change
0	1	1	Set output to 1
1	0	0	Reset output to 0
1	1	Q'	Opposite State

Outline

- **32 IEEE Floating Point**
- Computer Logic
  - **Combinational Circuits vs. Sequential Circuits**
  - **Gates**
  - **Combinational Circuits**
  - **Sequential Circuits**
- **Conclusion**

61

Conclusion

- Introduced an Example of the 32 bit IEEE Floating point format
- Talked about digital circuits
- Combinational and Sequential
- Should be able to understand a circuit

62

Help

- ***Graduate Teaching Assistants:***
  - Abdulwahab Kabani, *akabani5@uwo.ca*
  - Mike Molnar, *mmolnar2@uwo.ca*
  - Sakif Pritom, *spritom@uwo.ca*
- Send us an email or see us after class
- Consultation Room : Middlesex College - 342

63