Lecture 1

Grading system:

- Short tests on every lab lesson from the previous lesson.
- Exercises (during lectures) (recommended to read the material before hand)
- Final test

Logic gates

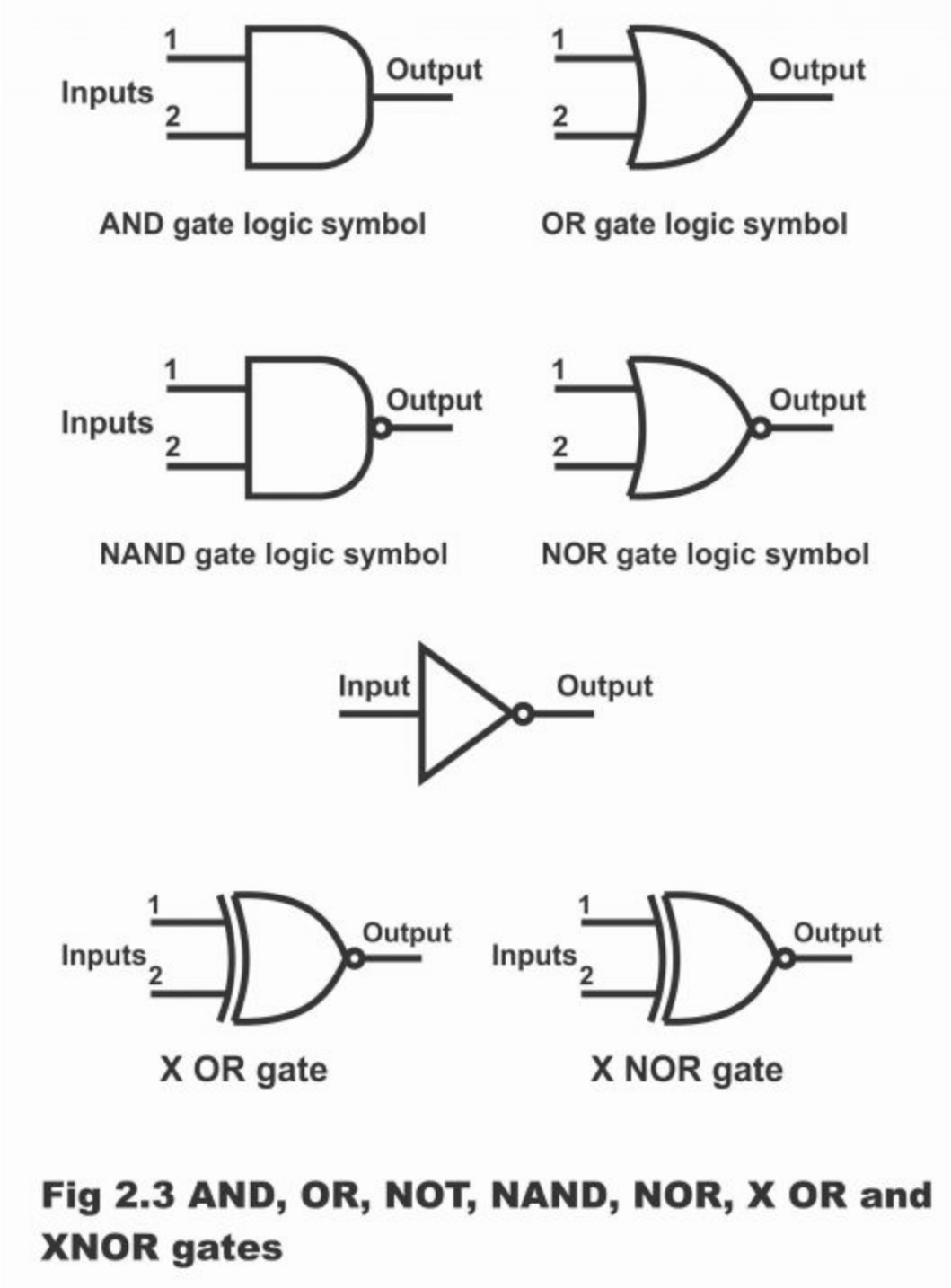
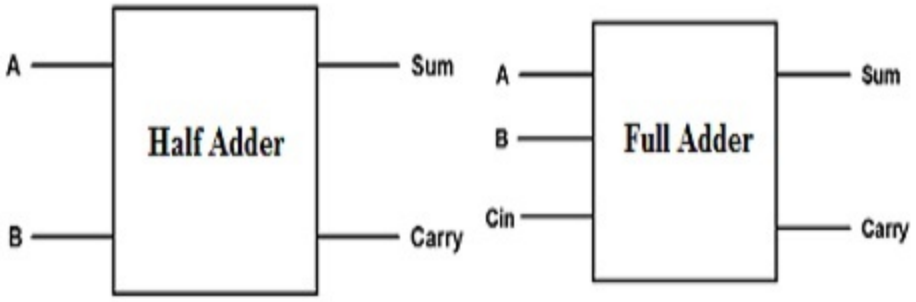


Fig 2.3 AND, OR, NOT, NAND, NOR, X OR and XNOR gates

Adder and half-adder

Half Adder is a combinational logic circuit that adds two 1-bit digits. The half adder produces a sum of the two inputs. A full adder is a combinational logic

circuit that performs an addition operation on three one-bit binary numbers. The full adder produces a sum of the three inputs and carry value.



Logic operations

Operation	Logic symbol	Boolean algebra notation
not	¬	$\bar{A}$
and	∧	AB
or	∨	A + B
xor	⊕	None

De Morgan’s Laws

In propositional logic and Boolean algebra, **De Morgan's laws**, also known as **De Morgan's theorem**, are a pair of transformation rules that are both valid rules of inference. They are named after Augustus De Morgan, a 19th-century British

mathematician. The rules allow the expression
of conjunctions and disjunctions purely in terms of each other via negation.

The rules can be expressed in English as:

- The negation of a disjunction is the conjunction of the negations
- The negation of a conjunction is the disjunction of the negations

or

- not (A or B) = (not A) and (not B)
- not (A and B) = (not A) or (not B)

DNF - Disjunctive normal form

In boolean logic, a **disjunctive normal form** (**DNF**) is a canonical normal form of a logical formula consisting of a disjunction of conjunctions; it can also be described as an **OR of ANDs**, a sum of products, or (in philosophical logic) a *cluster concept*. As a normal form, it is useful in automated theorem proving.

> For example, all of the following formulas are in DNF:
>
> - $(A \wedge \neg B \wedge \neg C) \vee (\neg D \wedge E \wedge F)$
> - $(A \wedge B) \vee (C)$
> - $(A \wedge B)$
> - (A)
>
> However, the following formulas are **not** in DNF:
>
> - $\neg(A \vee B)$, since an OR is nested within a NOT
> - $\neg(A \wedge B) \vee C$, since an AND is nested within a NOT
> - $A \vee (B \wedge (C \vee D))$, since an OR is nested within an AND

All logical formulas can be converted into an equivalent disjunctive normal form. However, in some cases conversion to DNF can lead to an exponential explosion of the formula.

Converting a truth table to a DNF

Given a truth table:

A	B	C	F
1	1	1	1
1	1	0	0
1	0	1	1
1	0	0	0
0	1	1	0
0	1	0	1
0	0	1	0
0	0	0	1

Do the following:

- For every A,B,C that output F = 1, take all the inputs that are 0 add write a negation symbol before them than simply add all the values. So for example in the first row F = 1, so we write $A \wedge B \wedge C$, and for the third row we write $A \wedge \neg B \wedge C$, because B was 0.
- Then take all those components and or them together

For the truth table above, the DNF would be:

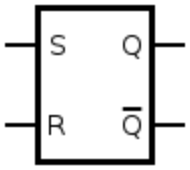
$$(A \wedge B \wedge C) \vee (A \wedge \neg B \wedge C) \vee (\neg A \wedge B \wedge \neg C) \vee (\neg A \wedge \neg B \wedge \neg C)$$

But this is not the optimal way of doing it, there are better algorithms, that generate shorter answers that require less logic gates.

Universal gates

NAND and NOR are universal.

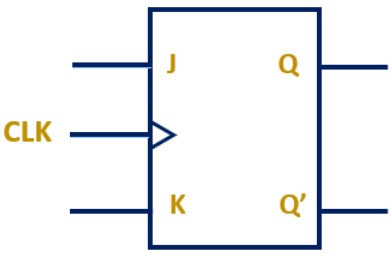
SR LATCH



When a high input is applied to the *Set* line of an SR latch, the *Q* output goes high (and not Q low). The feedback mechanism, however, means that the *Q* output will remain high, even when the *S* input goes low again. This is how the latch serves as a memory device. Conversely, a high input on the *Reset* line will drive the *Q* output low (and not *Q* high), effectively resetting the latch's "memory". When both inputs are low, the latch "latches" – it remains in its previously set or reset state.

JK flip flop

Symbol



Truth Table

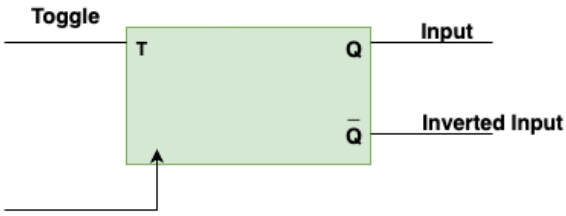
CLK	J	K	Q_{n+1}
↑	0	0	Q_n
↑	0	1	0
↑	1	0	1
↑	1	1	Q_n'

The J-K flip-flop is the most versatile of the basic flip-flops. It has the input-following character of the clocked D flip-flop but has two inputs, traditionally labeled J and K. If J and K are different then the output Q takes the value of J at

the next clock edge. The inputs are labeled J and K in honour of the inventor of the device, Jack Kilby.

If J and K are both low then no change occurs. If J and K are both high at the clock edge then the output will toggle from one state to the other. It can perform the functions of the set/reset flip-flop and has the advantage that there are no ambiguous states. It can also act as a T flip-flop to accomplish toggling action if J and K are tied together. This toggle application finds extensive use in binary counters.

T flip flop



T	Q	Q(t+1)
0	0	1
1	0	1
0	1	1
1	1	0

- Here, T is the Toggle input, Q is present state input, Qt+1 is the next state output.
- From here we can see that, whenever Toggle (T) is 0, next state output (Qt+1) is same as current state input (Q).
- Whenever Toggle (T) is 1, next state output (Qt+1) will be complement of current state input (Q) which means it gets toggled.

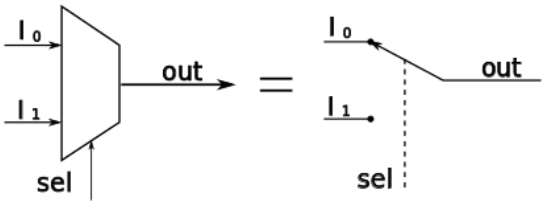
D flip flop

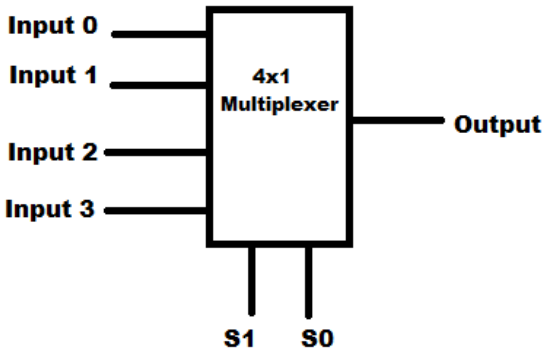
- When the clock signal is low, the flip flop holds its current state and ignores the D input.
- When the clock signal is high, the flip flop samples and stores D input.
- The value that was previously fed into the D input is reflected at the flip flop's Q output.

Register

A register is a storage location for data and instructions in digital electronics. **Register in digital electronics** are the basic element of a computer, and can hold arbitrary numbers of bits. This can range from one bit to millions of bits. Typically, it stores information for small amounts of time until it is sent on to another device or location.

Multiplexer





In electronics, a **multiplexer** (or **mux**; spelled sometimes as **multiplexor**), also known as a **data selector**, is a device that selects between several analog or digital input signals and forwards the selected input to a single output line.

The select lines represent a binary number which is the number of the input line that will be selected and forwarded to the output line.

Demultiplexer

The demultiplexer simply takes one input line and outputs it to one of the output line, depending on the state of the select lines. It is an inverse of the multiplexer.