

Exercise walk-through

15.2.1

**The full adder, flip-flops drawn
and traced, and four-variable
Karnaugh maps**

A-Level Computer Science

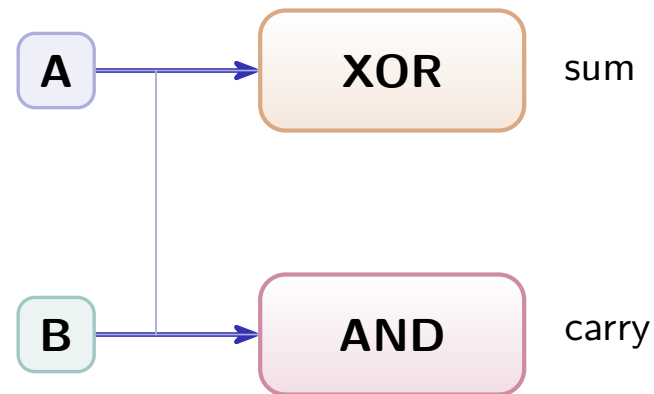
Half adder – two bits, no carry-in

A **half adder** adds two bits A and B :

$$S = A \text{ XOR } B, \quad C = A \text{ AND } B$$

It ignores any carry-in – hence “half”.

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

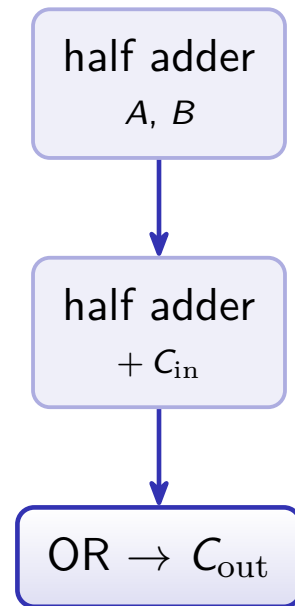


Full adder – three bits, including carry-in

A **full adder** adds A , B and a carry-in:

$$S = A \text{ XOR } B \text{ XOR } C_{\text{in}}$$

Built from **two half adders plus an OR gate**. Chain them (each carry-out feeds the next carry-in) to make a multi-bit *ripple-carry* adder.



Flip-flops – one bit of memory

A **flip-flop** is a **bistable** circuit: two stable states, so it **remembers** one bit. Building block of registers and SRAM.

SR

SR

S sets $Q = 1$; R resets $Q = 0$

$S = 0, R = 0$ holds

$S = 1, R = 1$ is **invalid**

two cross-coupled NOR gates

JK

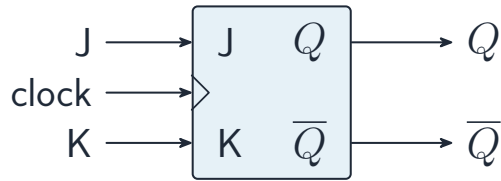
JK

$J = 1, K = 1$ is a **toggle**

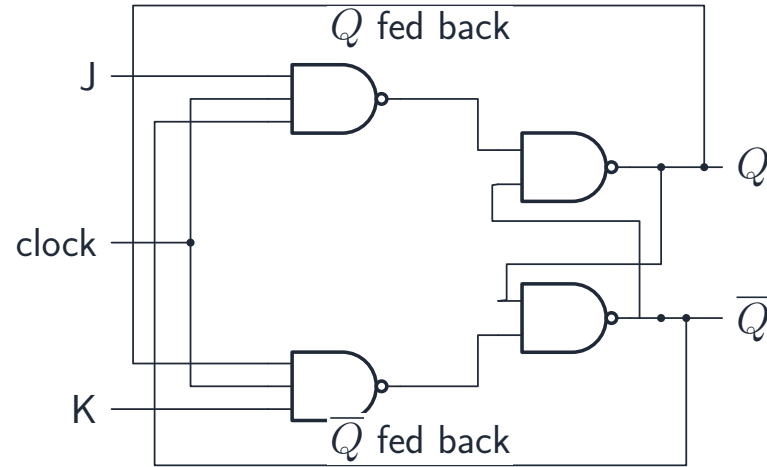
ideal for **counters**

usually *clocked* – inputs act on a clock edge

Flip-flops – one bit of memory, in detail



block symbol



build from NAND gates

A JK flip-flop: its symbol and a build from NAND gates

Pipelining



A CPU heat-sink and fan carry heat away from the processor

Exercises 2.1-2.2

2.1 State the **two** components a full adder has that a half adder does not, in terms of its inputs and what it is built from. [2]

2.2 Give the Boolean expression for the **Sum** output and for the **carry out** of a full adder. [2]

Exercises 2.1·2.2

2.1 State the **two** components a full adder has that a half adder does not, in terms of its inputs and what it is built from. [2]

2.1

- it has a **third input**, the carry in C_{in} , so it can add a column that receives a carry from the column to its right
- it is built from **two half adders and an OR gate**

2.2 Give the Boolean expression for the **Sum** output and for the **carry out** of a full adder. [2]

2.2

- $\text{Sum} = A \oplus B \oplus C_{\text{in}}$
- $C_{\text{out}} = A \cdot B + C_{\text{in}} \cdot (A \oplus B)$ – any equivalent form is accepted

Exercises 2.3-2.4

2.3 Use De Morgan's laws to simplify $\overline{\overline{A} + B}$. [2]

2.4 State what happens to the output of a **JK** flip-flop when $J = 1$ and $K = 1$. [1]

Exercises 2.3·2.4

2.3 Use De Morgan's laws to simplify $\overline{\overline{A} + B}$. [2]

2.3

- De Morgan on the whole bar: $\overline{\overline{A} + B}$
- double negation gives $A \cdot \overline{B}$

2.4 State what happens to the output of a **JK** flip-flop when $J = 1$ and $K = 1$. [1]

2.4

- the output **toggles**: it changes to the opposite of its previous value

Exercise 2.5

2.5 State why the rows and columns of a Karnaugh map are written in **Gray code** rather than in binary counting order. [2]

Exercise 2.5

2.5 State why the rows and columns of a Karnaugh map are written in **Gray code** rather than in binary counting order. [2]

2.5

- in Gray code, **adjacent cells differ in exactly one variable**
- so a group of adjacent 1s has that one variable taking both values, which lets it be cancelled; in binary counting order adjacent cells could differ in two variables and grouping them would be invalid

Exercise 3.1

3.1 Complete the truth table for a **full adder** with inputs A , B and C_{in} .

[4]

A	B	C_{in}	Sum	C_{out}
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Exercise 3.1

3.1 Complete the truth table for a **full adder** with inputs A , B and C_{in} .

[4]

A	B	C_{in}	Sum	C_{out}
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

3.1

- Sum column correct: 0, 1, 1, 0, 1, 0, 0, 1
- C_{out} column correct: 0, 0, 0, 1, 0, 1, 1, 1

Full solution on next slide

Full solution 3.1

3.1

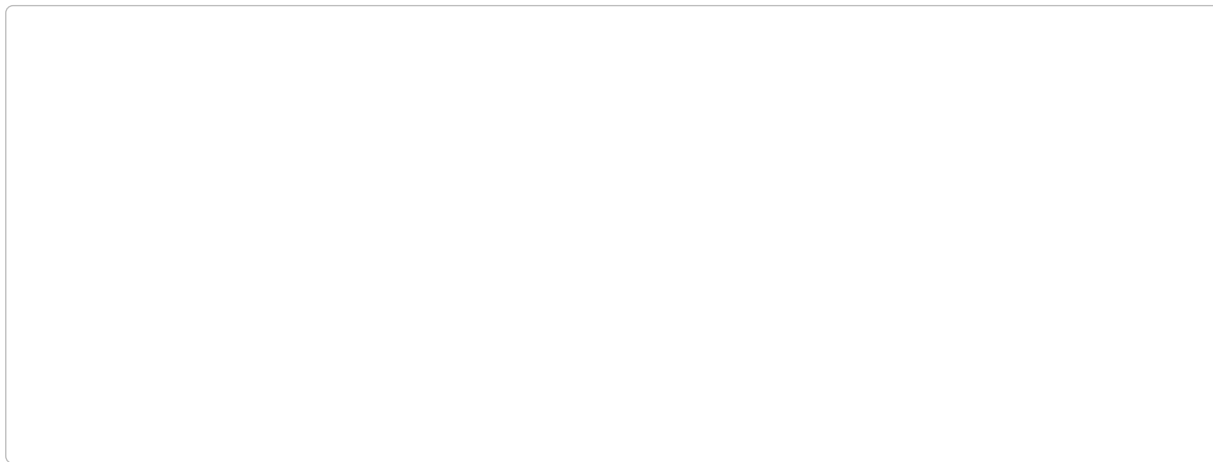
- Sum column correct: 0, 1, 1, 0, 1, 0, 0, 1
- C_{out} column correct: 0, 0, 0, 1, 0, 1, 1, 1

A	B	C_{in}	Sum	C_{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

- the Sum is 1 on the rows with an **odd** number of 1s; the carry out is 1 whenever at least two inputs are 1

Exercise 3.2

3.2 (a) Draw the logic circuit for an **SR flip-flop**, labelling the inputs S and R and the outputs Q and $\overline{Q}$. [3]



(b) Complete the truth table for the flip-flop, and state which row is **invalid** and why. [4]

Exercise 3.2

3.2 (a) Draw the logic circuit for an **SR flip-flop**, labelling the inputs S and R and the outputs Q and $\overline{Q}$. [3]



(b) Complete the truth table for the flip-flop, and state which row is **invalid** and why. [4]

3.2

(a) two **NOR** gates

Full solution 3.2

3.2

(a) two **NOR** gates

- **cross-coupled**: the output of each is an input to the other
- S and R labelled as the remaining input of one gate each, and the outputs labelled Q and $\overline{Q}$

(b) $S = 0, R = 0$: Q **holds** its previous value

- $S = 0, R = 1$: $Q = 0$, reset; $S = 1, R = 0$: $Q = 1$, set
- $S = 1, R = 1$ is the **invalid** row
- because both outputs would be forced to 0, so Q and $\overline{Q}$ would no longer be complements of each other, and what happens next depends on which input changes first

Exercise 3.3

3.3 Explain why a flip-flop can act as a **data storage element**.

[2]

Exercise 3.3

3.3 Explain why a flip-flop can act as a **data storage element**.

[2]

3.3

- the outputs are **fed back** as inputs, so with both inputs at 0 the circuit holds whatever state it was last put into
- that state persists until it is deliberately set or reset, so one flip-flop stores **one bit**;
a register is a row of them

Exercise 3.4 (a)

3.4 A logic circuit with four inputs A , B , C and D gives an output of 1 only when $ABCD$ is 0000, 0010, 1000 or 1010.

(a) Use a Karnaugh map to obtain the simplified Boolean expression for the output. [3]

Exercise 3.4 (a)

3.4 A logic circuit with four inputs A , B , C and D gives an output of 1 only when $ABCD$ is 0000, 0010, 1000 or 1010.

(a) Use a Karnaugh map to obtain the simplified Boolean expression for the output. [3]

(a) the four cells are the **corners** of the map, and a K-map **wraps round**, so they form one valid group of four

- across all four, $B = 0$ and $D = 0$; A and C each take both values and cancel
- $\mathbf{F = \overline{B} \cdot \overline{D}}$
- four separate groups of one, giving four ANDed terms, earns the first mark only: the question is about grouping

Exercise 3.4 (b)

3.4 A logic circuit with four inputs A , B , C and D gives an output of 1 only when $ABCD$ is 0000, 0010, 1000 or 1010.

(b) State **two** benefits of using a Karnaugh map rather than Boolean algebra. [2]

Exercise 3.4 (b)

3.4 A logic circuit with four inputs A , B , C and D gives an output of 1 only when $ABCD$ is 0000, 0010, 1000 or 1010.

- (b) State **two** benefits of using a Karnaugh map rather than Boolean algebra. [2]
- (b) any two: it gives the simplified expression **directly**, without a chain of algebra
- the simplified circuit needs **fewer gates**, so it is cheaper, smaller and faster; and grouping is visual, so it is less error-prone than algebraic manipulation

Exercise 3.5

3.5 Simplify $Z = A + \overline{A}B$, showing each step and naming the law used.

[3]

Exercise 3.5

3.5 Simplify $Z = A + \overline{A}B$, showing each step and naming the law used. [3]

3.5

- $A + \overline{A}B$: apply the **distributive** law, $A + \overline{A}B = (A + \overline{A})(A + B)$
- $A + \overline{A} = 1$ – the **complement** law
- $1 \cdot (A + B) = \mathbf{A + B}$ – the identity law
- quoting the absorption result $A + \overline{A}B = A + B$ directly earns the answer mark but not the working marks, and the question says to show each step