Combinational Circuits II

Aims

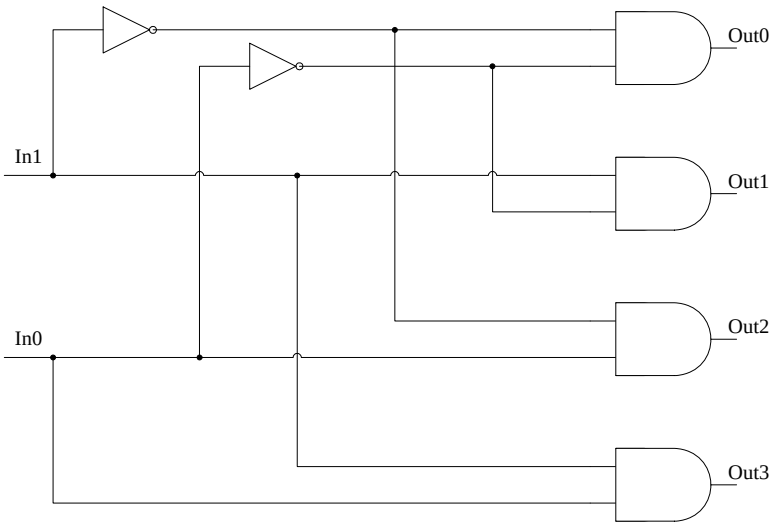
- to show how logic gates can be combined to produce useful circuits such as decoders and adders
- to show how individual combinational circuits can be combined to produce a simple ALU
- to introduce a simple memory storage device called a register

3.1 Decoders

A ***decoder*** is a combinational circuit that converts binary information from ***n*** inputs to a maximum of 2^n unique outputs.

- A decoder takes an ***n***-bit number as input and uses it to select (set to 1) exactly one of the 2^n outputs.

Below is a **2-to-4 decoder**



In0	In1	$\overline{In0}$	$\overline{In1}$	Out0	Out1	Out2	Out3
0	0						
0	1						
1	0						
1	1						

SAQ 3.1

1. How many outputs does a 2-input decoder have?
2. How many outputs does a 3-input decoder have?
3. How many outputs does a 4-input decoder have?
4. What is the purpose of a decoder?

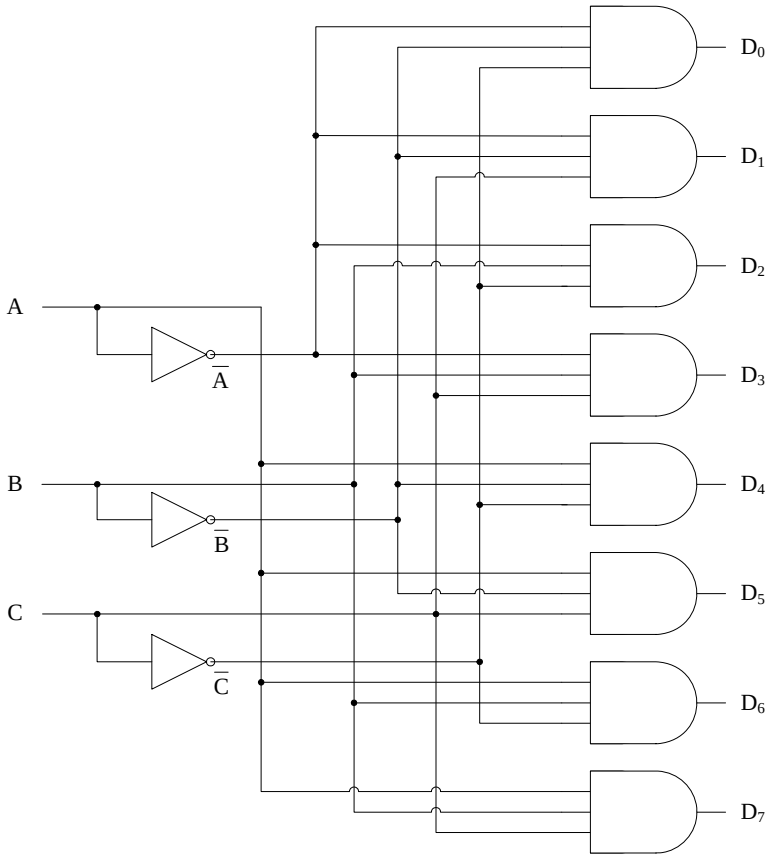
answer

3.1.1 Line Decoder

The circuit below is a ***3-to-8 decoder*** and its operation is straightforward.

- Each **AND** gate has three inputs (the first is either **A** or $\overline{A}$, the second is either **B** or $\overline{B}$, and the third is either **C** or $\overline{C}$
- Each gate is enabled by a different combination of inputs :

D_0 by $\overline{A}\,\overline{B}\,\overline{C}$, **$D_1$** by $\overline{A}\,\overline{B}\,$**C** etc.



The operation of the 3-to-8 (line) decoder can be clarified from the truth table (right).

A	B	C	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	**1**	0	0	0	0	0	0	0
0	0	1	0	**1**	0	0	0	0	0	0
0	1	0	0	0	**1**	0	0	0	0	0
0	1	1	0	0	0	**1**	0	0	0	0
1	0	0	0	0	0	0	**1**	0	0	0
1	0	1	0	0	0	0	0	**1**	0	0
1	1	0	0	0	0	0	0	0	**1**	0
1	1	1	0	0	0	0	0	0	0	**1**

For each possible input combination there are seven outputs that are equal to **0** and only one that is equal to **1**.

The output equal to **1** represents the equivalent of the binary number that is applied to the inputs.

An application of a decoder is as a ***device selector***.

- If, for example, a system has eight devices and only one can be in use at any instant, the inputs to the decoder can be used to select which device is used.

3.2 Adders

A computer that cannot ***add integers*** is unthinkable.

- Consequently, a circuit for performing addition is an essential part of every **CPU**.

3.2.1 Half Adder

A ***half adder*** is an arithmetic circuit that performs the ***addition of two bits***.

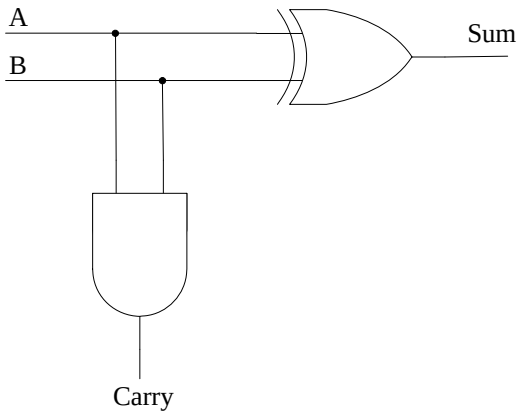
The circuit has two inputs (**A**, **B**) and two outputs (**Sum**, **Carry**).

- Two outputs are necessary because the **sum** of two binary digits ranges from 0 to 2, and the binary equivalent of 2 requires two digits.

The sum output represents the ***sum*** of the two inputs while the carry output represents the ***carry*** to the next leftward position.

The simple addition performed by a **half adder** can consist of four possible operations:

0+0	=	0
0+1	=	1
1+0	=	1
1+1	=	10



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

- The carry output is **1** only when both inputs are **1**.
- The sum output is **1** only when either of the inputs are **1** (the sum output represents the least significant bit of the sum).

The boolean algebra representation of the two outputs is:

$$\textbf{Sum} = A \oplus B$$
$$\textbf{Carry} = A \cdot B$$

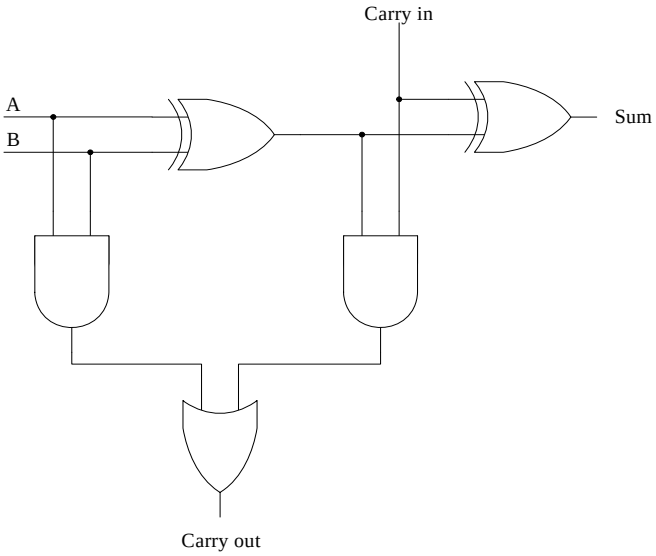
3.2.2 Full Adder

A **full adder** is a circuit that performs the ***addition of three bits***.

- The name of the circuit stems from the fact that two half adders are joined together to create a full adder.

The circuit has three inputs (**A**, **B**, **Carry in**) and two outputs (**Sum**, **Carry out**).

- Two outputs are necessary because the sum of three binary digits ranges from 0 to 3, and the binary equivalent of 2 or 3 requires two digits.



A	B	Carry in	Carry out	Sum
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

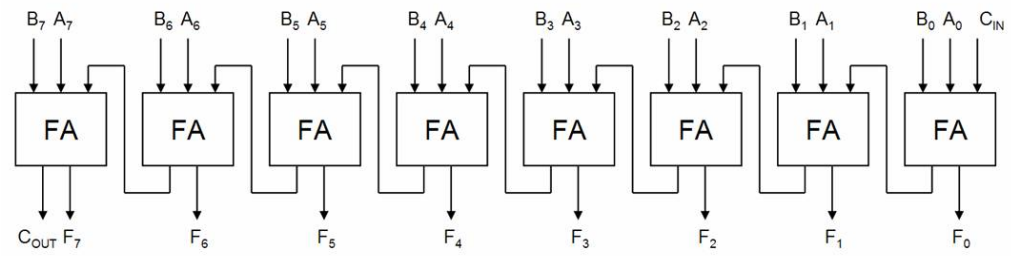
- The sum output is **1** when only one input is equal to **1**, or when all three inputs are equal to **1**.
- The carry output is **1** when either two or three of the inputs are equal to **1**.

The boolean algebra representation of the two outputs is:

$$\textbf{Sum} = (A \oplus B) \oplus \textbf{Carry in}$$

$$\textbf{Carry out} = (A \oplus B) \cdot \textbf{Carry in} + (A.B)$$

To build an adder for **two 8-bit numbers** the **full adder is replicated eight times**.



- The carry out of one full adder is used as the carry in into its left neighbour
- The carry into the rightmost bit is set to **0**
- The carry out from the Most Significant Bit sets the OVERFLOW flag in the CPU Status Register

This type of adder is called a ***ripple carry adder***

… because the addition cannot complete until the carry has rippled all the way across the adders.

SAQ 3.2

1. How many bits does a half adder add?
2. How many bits does a full adder add?
3. Why does a half adder require two outputs?
4. Why does a full adder require two outputs?
5. How would you build an adder to add two 8-bit numbers?

answer

4.1 ALU

At the heart of every computer is a processor, or ***CPU*** **(Central Processing Unit)**

CPUs consist of two main components –

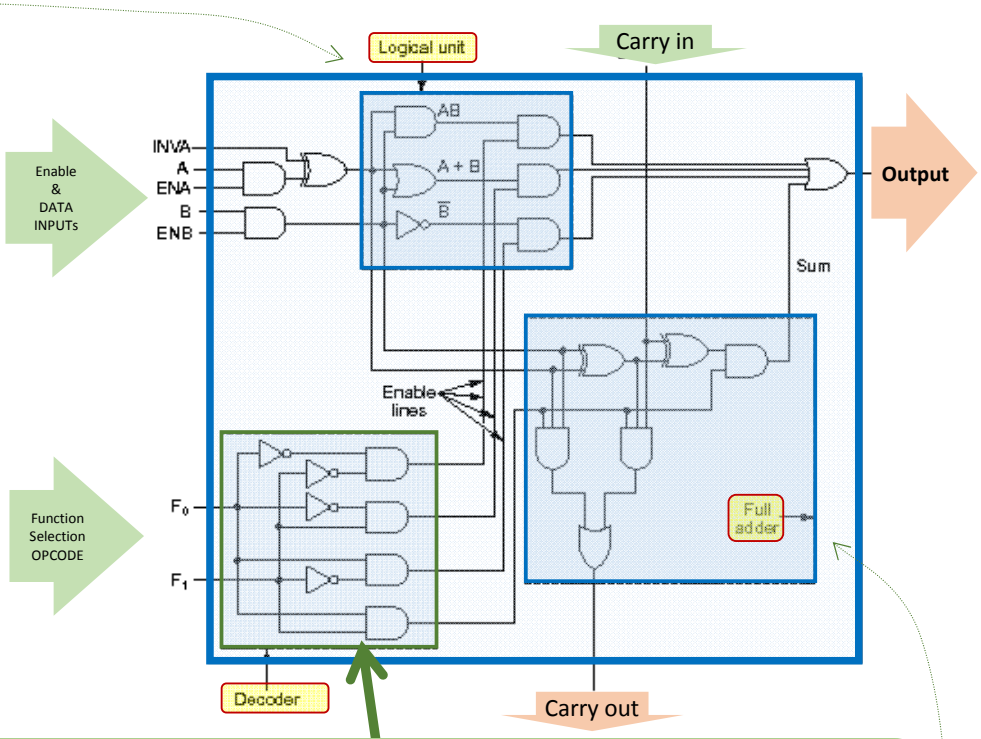
- an ***ALU*** **(Arithmetic and Logic Unit)**
- ***Control Unit***

We will return to the **control unit** later in the course

The ***ALU*** is a circuit, used in the **CPU** for performing **A**rithmetic and (Boolean) **L**ogical operations

- The simple 1-bit ALU shown here can compute any of four functions:

	Operation	F_0	F_1
1	**A AND B**	0	0
2	**A OR B**	0	1
3	**NOT B**	1	0
4	**SUM of A and B**	1	1



The **lower left corner** of the **ALU** contains a **2-to-4 Decoder** to generate ***enable signals*** for the four operations, based on the 2-bit function control (OPCODE selection) inputs F_0 and F_1

- Depending on the values of F_0 and F_1 one of the outputs of the four functions is passed through to the final **OR** gate for output.

The **upper left corner** of the **ALU** contains the **logic** to compute ***A AND B***, ***A OR B***, and ***NOT B***, but only one of these results is passed onto the final **OR** gate, depending on the enable signal produced by the decoder

- Only one decoder output will be **1**, and so only one of the four **AND** gates that are inputs to the final **OR** gate will output a **1**; the other three will output a **0**.

As well as being able to use **A** and **B** as inputs for logical and arithmetic operations,

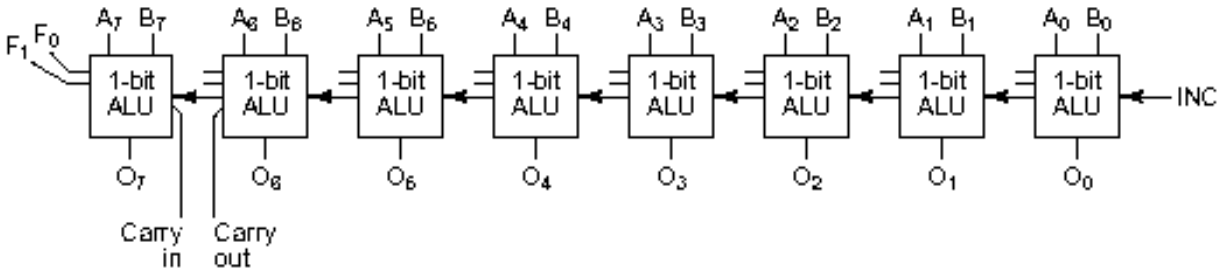
- it is possible to force either one to **0** by negating ***ENA*** or ***ENB***.
- It is also possible to produce $\overline{A}$ by setting ***INVA***.
- Under normal circumstances ***ENA*** and ***ENB*** are both **1** to enable both inputs and ***INVA*** is **0**. Under such conditions **A** and **B** are fed into the logic unit unmodified.

The **lower right corner** of the **ALU** contains a full adder for computing the sum of **A** and **B**.

- The full adder also handles the **carries**, because it is possible that several of these circuits will be connected together to perform addition of larger numbers.

Such **1-bit ALUs** allow computer designers to build an **ALU** of any desired size.

- Below is an **8-bit ALU** built of eight 1-bit ALUs
    - The ***INC*** signal is used for addition operations
        - When present it increments (adds 1) to the result, allowing the computation of sums such as **A+1**



SAQ 4.1

1. What is the purpose of an ALU?
2. What are the three main components of a simple 1-bit ALU?

answer

4.2 Memory

Combinational circuits have the property that ***their outputs are wholly dependent on their inputs***.

There is another class of circuit that is ***sequential*** in nature.

- ***Sequential circuits*** have the property that their ***outputs depend not only on present input values, but also on past input values***.

4.2.1 Flip-Flops

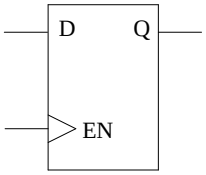
A ***flip-flop*** is a sequential circuit that forms the basic building block of memory.
A flip-flop stores a single bit by producing an output of a **0** or **1** until changed by conditions on the input(s).

- The term flip-flop comes from the operation of the flip-flop – the output ***flips*** from a **0** to a **1** or ***flops*** from 1 to a 0.
- The **output of a flip-flop** is given the identification letter ***Q***.

There are several types of flip-flop, but only the ***D-type flip-flop*** will be discussed here.

4.2.2 D-type Flip-Flops

Flip-flops are fundamental components of logic systems and so have a standard symbol to denote them. The symbol for a D-type flip-flop is shown below.

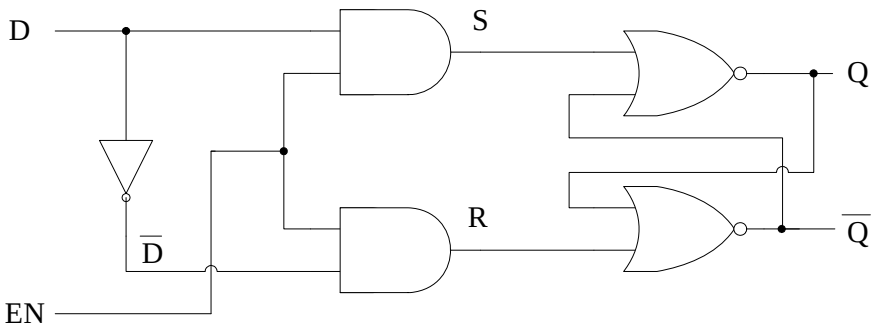


A D-type flip-flop has two inputs, **D** and **EN**.

- The **D** input is referred to as the ***data*** input and is the data that the flip-flop is to store.
- The **EN** input is referred to as the ***enable*** input.
- In a flip-flop the output ***Q*** only changes on the activating transition of the **EN** signal from one specified logic level to the other logic level.
- Without an activating transition of the EN signal the output will not change even if the input D does. : In this situation the output remains constant until the next activating transition.
    - This is referred to as ***edge triggering***.
    - In ***positive edge triggering***, the activating transition is from a **0** to a **1**.
    - In ***negative edge triggering***, the activating transition is from a **1** to a **0**.

Both forms are common in flip-flops.
For simplicities sake we will assume that positive edge triggering is being used.

D Type flip-flop constructed from logic gates



The operation of the ***D-type flip-flop*** can be clarified by a simplified truth table.

EN	D	Q
0	0	Q
0	1	Q
1	0	0
1	1	1

- When **EN** is **0** the input to the two **AND** gates is **0** and so the **S** and **R** inputs to the two **NOR** gates are also set to **0**.
    - This means that the value of **Q** cannot change and remains constant.
- When **EN** is **1** the **S** input is connected to **D** and the **R** input to $\overline{D}$.
    - This means that the value on the **D** input is passed through to the output **Q**.
    - When **EN** returns to **0** the output **Q** is held constant at the last value of input **D** until the next activating transition of the **EN** signal.

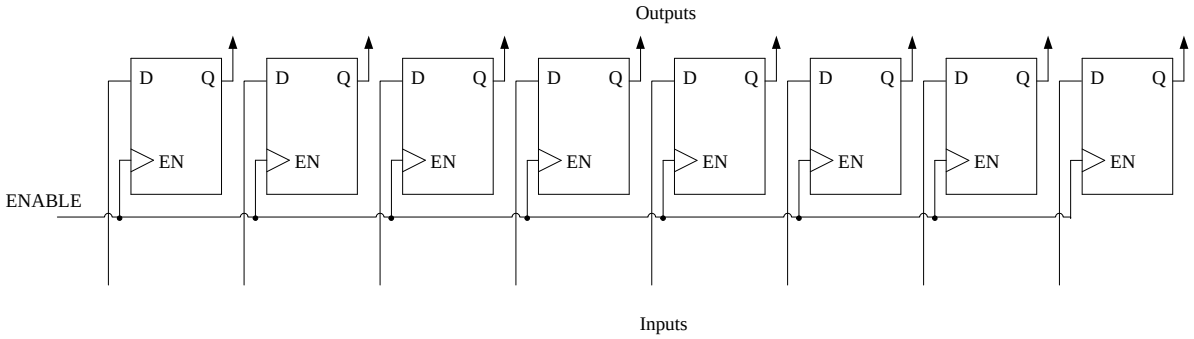
4.2.3 Registers

As well as an ***ALU***, a CPU also contains a ***small, high-speed memory used to store temporary results and control information***.

This memory is made up of a number of ***registers***, with each register being able to hold one number, up to a maximum size determined by the size of the register.

A flip-flop can store one binary digit. To store eight bits, eight flip-flops must be combined.

The diagram below shows eight flip-flops combined into an 8-bit register.



- When **EN** is **0** the flip-flops do not record the data being input to them and their **Q** outputs remain unchanged.
- When a device wishes to transfer its data to the flip-flops it sends it's data to the inputs and then creates an activating transition of the **EN** signal (**EN** becomes **1**).
  - The flip-flops then store the data on their inputs.
- When **EN** returns to **0** the data remains frozen in the flip-flops.
  - Registers will be returned to later in the course.

SAQ 4.2

1. What is the difference between a combinational circuit and a sequential circuit?
2. What is the purpose of a flip-flop?
3. When does the output of a flip-flop change?
4. What is a register?

answer

References

http://ece-research.unm.edu/pollard/classes/338/lademo/LookAheadDemo.htm