Pearson BTEC Level 4 Higher Nationals in Engineering (RQF)

Unit 20: Digital Principles

Unit Workbook 1 of 1

CONTENTS

1 INTRODUCTION

This Workbook guides you through the learning outcomes related to:

LO1. Explain and analyse simple combinational logic circuits.

Concepts of combinational logic:

Simple logic circuits implemented with electro-mechanical switches and transistors. Circuits built from AND, OR, NAND, NOR, XOR gates to achieve logic functions, e.g. majority voting, simple logical controls, adders.

Number systems, and binary arithmetic:

Binary, Decimal, Hexadecimal number representation, converting between, applications and relative advantages. Addition and subtraction in binary, range of n-bit numbers.

Analysis of logic circuits:

Truth Tables, Boolean Algebra, de Morgan's theorem, Karnaugh Maps.

Simplification and optimisation of circuits using these techniques.

LO2. Explain and analyse simple sequential logic circuits.

Sequential logic elements and circuits:

SR latch built from NAND or NOR gates.

Clocked and edge-triggered bistables, D and JK types.

Simple sequential circuits, including shift registers and counters.

Timing Diagrams.

Memory technologies:

Memory terminology, overview of memory technologies including Static RAM, Dynamic RAM and Flash memory cells.

Relative advantages in terms of density, volatility and power consumption.

Typical applications, e.g. in memory stick, mobile phone, laptop.

LO3. Describe and evaluate the technologies used to implement digital electronic circuits.

Logic values represented by voltages:

Sequential logic elements and circuits:

The benefit of digital representation of information.

The concept of logic input and output values and thresholds.

Digital technologies:

Introduction to discrete logic families, CMOS and TTL, relative advantages in terms of speed, power consumption, density.

Programmable logic, FPGAs, relative advantages and applications.

LO4. Describe and analyse a range of digital subsystems, hence establishing the building blocks for larger systems.

User interface:

Examples to include switches, light emitting diodes and simple displays.

Digital subsystems:

Examples to be drawn from adders (half, full, n-bit), multiplexers and demultiplexers, coders and decoders, counters applied as timers, shift registers applied to serial data transmission, elements of the ALU (Arithmetic Logic Unit). Emphasis on how these can be applied, and how they might fit into a larger system.

IMPORTANT NOTE:

It is recommended that in conjunction with this Workbook, that you also read "round your subject". This means accessing and reading the references and other material given in this workbook as well undertaking your own research.

It is also recommended that you consult the following, which are related units:
Unit 19: Electrical and Electronic Principles
Unit 22: Electronic Circuits and Devices
Unit 52: Further Electrical, Electronic and Digital Principles

Definitions: Commonly used words

Explain: Make (an idea or situation) clear to someone by describing it in more detail or revealing relevant facts.

Identify: Indicate the main features or purpose of something by recognising it and/or being able to discern and understand facts or qualities.

Describe: Give a detailed account in words of.

Analyse: Examine (something) methodically and in detail, typically to explain and interpret it.

Critically Analyse: A critical analysis (sometimes called a critique, critical summary, or book review) is a systematic analysis of an idea, text, or piece of **literature** that discusses its **validity** and evaluates its worth.

Apply: Be applicable or relevant.

Explore: Skills and/or knowledge involving practical research or testing.

Determine: Ascertain or establish exactly by research or calculation.

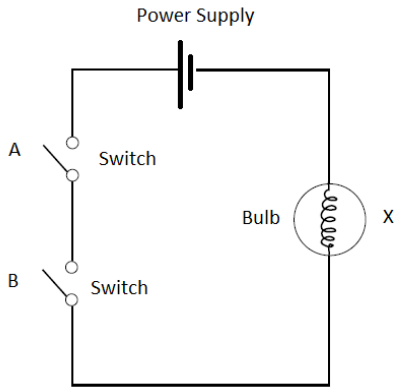
Scope: The extent of the area or subject matter that something deals with or to which it is relevant.

See Moodle 'useful course resources' for an in-depth definition of words/terms or consult the Pearson BTEC Higher Nationals in Engineering Specification – Issue 5 - September 2017.

2 CONCEPTS OF COMBINATIONAL LOGIC

2.1 Simple Logic Circuits.

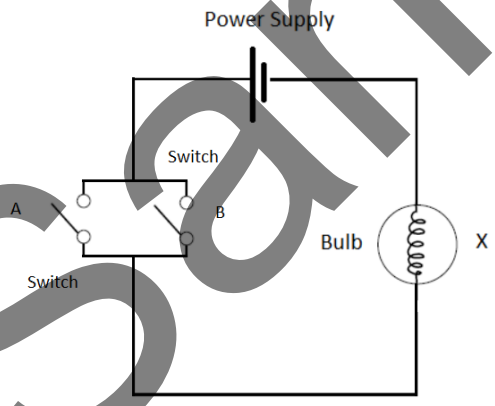
Consider a simple electrical circuit consisting of a battery, a bulb, and two switches connected in series. The switches can be considered as inputs to the circuit and the bulb as the output. A truth table is a convenient method of describing the operation of the circuit (See Figure 1).



A	B	X
Open	Open	Off
Open	Closed	Off
Closed	Open	Off
Closed	Closed	On

Figure 1 Switch representation of a 2-input AND function and its associated Truth Table

As the bulb is only lit (ON) when *both* the A and B switches are Closed (ON), then this circuit can be said to perform a 2-input AND function. In fact, the results depend on the way in which the switches are connected; consider another circuit in which the two switches are connected in parallel (Figure 2).



A	B	X
Open	Open	Off
Open	Closed	On
Closed	Open	On
Closed	Closed	On

Figure 2 Switch representation of a 2-input OR function and its associated Truth Table

In this case, as the light is ON when either A or B are Closed (ON), this circuit can be said to provide a 2-input OR function.

These circuits are performing simple logical functions, which can also be implemented in many other ways (transistor, pneumatic, hydraulic, and magnetic devices for example, where ON and OFF are represented by different mechanisms)

Thus, it is possible to specify a particular logic function without necessarily specifying its final physical realisation. To facilitate this, special symbols are employed to represent logic functions, and the associated

truth table assignments are specified using more abstract terms, specifically; TRUE and FALSE. The representation of the logic function is usually called a GATE.

2.2 NOT Gate

The simplest of all Logic Gates is the NOT Gate. It's Symbol, Truth Table, and Timing diagram representation is shown in Figure 3. The small circle on the output of the gate represents an inverting function.

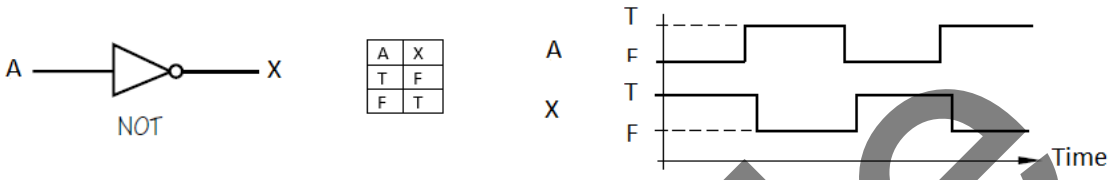


Figure 3 Logic NOT representation:
Symbol, Truth Table and Timing diagram representation

As a reminder that these abstract functions will eventually have physical realisations, the timing diagram waveforms shows a delay between transitions on the inputs and corresponding responses at the outputs. The actual values of these delays depend on the technology used to implement the functions, but it is important to note that in any physical implementation there will always be some element of delay.

2.3 AND, OR and XOR Gates

Three slightly more complex functions are known as AND, OR, and XOR. These are shown in Figures xxxx, below.

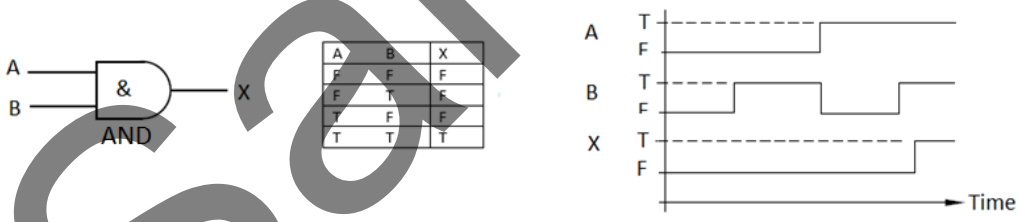


Figure 4 Logic AND representation:
Symbol, Truth Table and Timing diagram representation

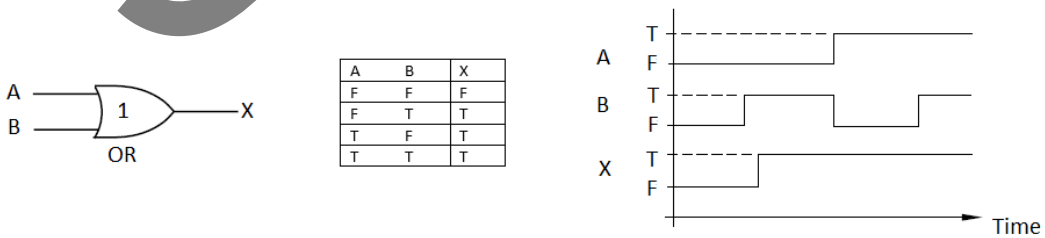


Figure 5 Logic OR representation:
Symbol, Truth Table and Timing diagram representation

3 NUMBER SYSTEMS

3.1 Decimal and Binary number systems

Almost certainly because humans have ten fingers, the decimal, or base-10 numbering system has been adopted universally. The value of particular digit in a decimal number depends on the **value** of the digit and its **position** within the number. The decimal system is, therefore, a **place-value** system.

Many other numbering systems are used for varying purposes, but it is not surprising that the logic systems that we introduced in the Chapter 4 infer a Binary, or Base-2 numbering system, the relevant values of the digits used being 0 (representing False, for example) and 1 (representing True, for example).

Each column in a binary number has a weight derived from the base, and each digit is combined with its column's weight to determine the final value of the number. This is shown in Figure 7.

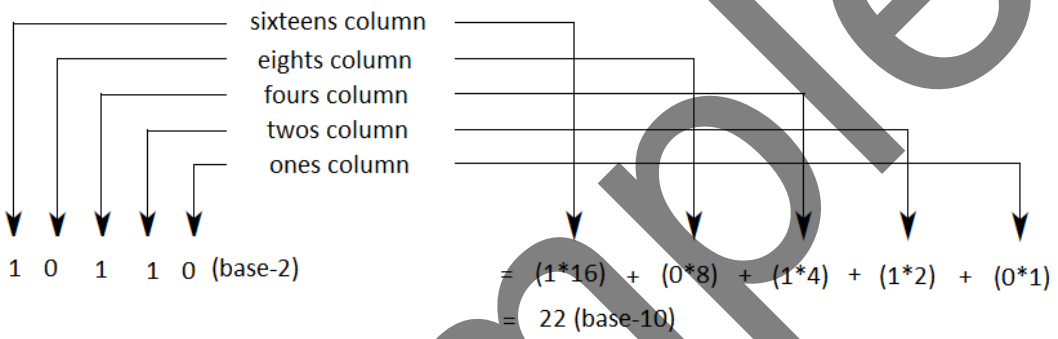


Figure 7 Combining digits with column weights in binary

The 1 or 0 in a binary number is a binary digit, which neatly contracts to become **bit**. The binary value 10110_2 is said to be 5 bits wide.

A group of 4 bits is known as a nybble (nibble) and a group of 8 bits is known as a byte.

Did you know that there are only ten types of people in the world; those that understand binary and those that don't! (Think about it!)

3.2 Octal (Base-8) and Hexadecimal (Base-16)

Any number system having a base that is a power of two (2, 4, 8, 16, 32, etc.) can be easily mapped into its binary equivalent and vice versa. For this reason, electronics engineers typically make use of either the octal (base-8) or hexadecimal (base-16) systems.

As a base-8 system, octal requires eight individual symbols to represent all of its digits. This isn't a problem because we can simply use the symbols 0 to 7. However, as a base-16 system, hexadecimal requires sixteen individual symbols to represent all of its digits; (0 to 9) and (A to F).

The rules for counting in octal and hexadecimal are the same as for any other place-value system – when all the digits in a column are exhausted, the next count sets that column to zero and increments the column to the left (see Figure 8).

Decimal	Binary	Octal	Hexadecimal
0	00000000	000	00
1	00000001	001	01
2	00000010	002	02
3	00000011	003	03
4	00000100	004	04
5	00000101	005	05
6	00000110	006	06
7	00000111	007	07
8	00001000	010	08
9	00001001	011	09
10	00001010	012	0A
11	00001011	013	0B
12	00001100	014	0C
13	00001101	015	0D
13	00001110	016	0E
15	00001111	017	0F
16	00010000	020	10
17	00010001	021	11
18	00010010	022	12
:	:	:	:
Etc.	Etc.	Etc.	Etc.

Figure 8 Counting in Octal and Hexadecimal

Although not strictly necessary, binary, octal and hexadecimal numbers are often prefixed by leading zeros to pad them to whatever width is required. This padding is usually performed to give some indication as to the physical number of bits used to represent that value within a computer. Each octal digit can be directly mapped onto three binary digits, and each hexadecimal digit can be directly mapped onto four binary digits (Figure 9).

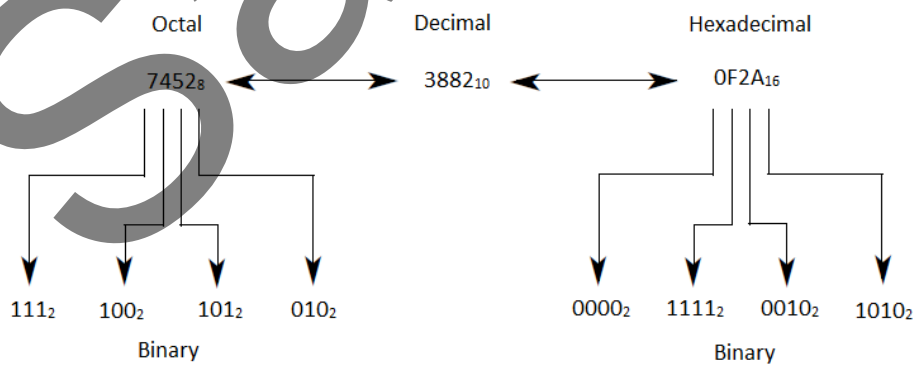


Figure 9 Mapping Octal and Hexadecimal to Binary

In the early digital computers, data paths were often 9 bits, 12 bits, 18 bits, or 24 bits wide, which explains the original popularity of octal notation. Due to the fact that each octal digit maps directly to three binary bits, these data-path values were easily represented in octal. More recently, digital computers have standardized on data-path widths that are integer multiples of 8 bits; for example, 8 bits, 16 bits, 32 bits,