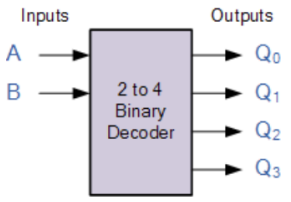


Truth Table

A	B	Q_0	Q_1	Q_2	Q_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

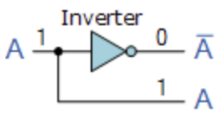
Binary Decoder

Binary Decoder is another combinational logic circuit constructed from individual logic gates and is the exact opposite to that of an Encoder

The term "Decoder" means to translate or decode coded information from one format into another, so a binary decoder transforms "n" binary input signals into an equivalent code using 2^n outputs.

Binary Decoders are another type of digital logic device that has inputs of 2-bit, 3-bit or 4-bit codes depending upon the number of data input lines, so a decoder that has a set of two or more bits will be defined as having an *n*-bit code, and therefore it will be possible to represent 2^n possible values. Thus, a decoder generally decodes a binary value into a non-binary one by setting exactly one of its *n* outputs to logic "1".

If a binary decoder receives n inputs (usually grouped as a single Binary or Boolean number) it activates one and only one of its 2^n outputs based on that input with all other outputs deactivated.



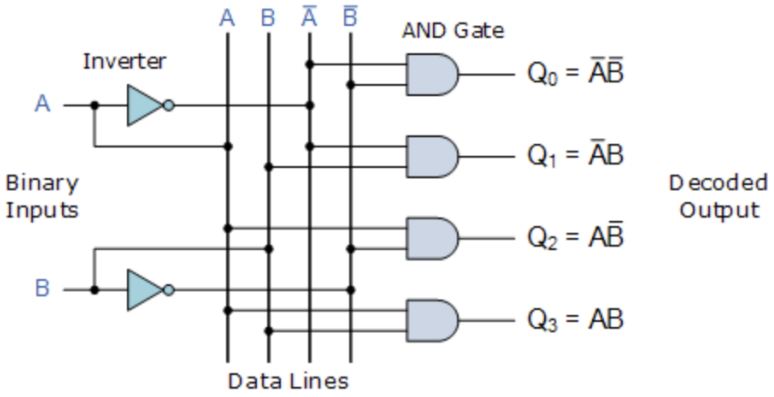
So for example, an inverter (*NOT-gate*) can be classed as a 1-to-2 binary decoder as 1-input and 2-outputs (2^1) is possible because with an input A it can produce two outputs A and $\overline{A}$ (not-A) as shown.

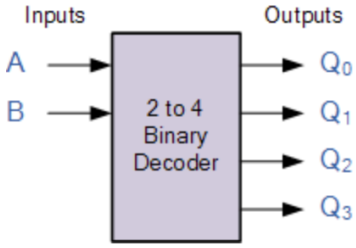
Then we can say that a standard combinational logic decoder is an *n-to-m* decoder, where $m \le 2^n$, and whose output, Q is dependent only on its present input states. In other words, a binary decoder looks at its current inputs, determines which binary code or binary number is present at its inputs and selects the appropriate output that corresponds to that binary input.

A *Binary Decoder* converts coded inputs into coded outputs, where the input and output codes are different and decoders are available to "decode" either a Binary or BCD (8421 code) input pattern to typically a Decimal output code. Commonly available BCD-to-Decimal decoders include the TTL 7442 or the CMOS 4028. Generally a decoders output code normally has more bits than its input code and practical "binary decoder" circuits include, 2-to-4, 3-to-8 and 4-to-16 line configurations.

An example of a 2-to-4 line decoder along with its truth table is given as:

A 2-to-4 Binary Decoders





Truth Table

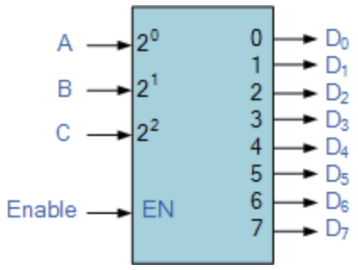
A	B	Q_0	Q_1	Q_2	Q_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

This simple example above of a 2-to-4 line binary decoder consists of an array of four AND gates. The 2 binary inputs labelled A and B are decoded into one of 4 outputs, hence the description of 2-to-4 binary decoder. Each output represents one of the miniterms of the 2 input variables, (each output = a miniterm).

The binary inputs A and B determine which output line from Q0 to Q3 is “HIGH” at logic level “1” while the remaining outputs are held “LOW” at logic “0” so only one output can be active (HIGH) at any one time. Therefore, whichever output line is “HIGH” identifies the binary code present at the input, in other words it “de-codes” the binary input.

Some binary decoders have an additional input pin labelled “Enable” that controls the outputs from the device. This extra input allows the decoders outputs to be turned “ON” or “OFF” as required. These types of binary decoders are commonly used as “memory address decoders” in microprocessor memory applications.

We can say that a binary decoder is a demultiplexer with an additional data line that is used to enable the decoder. An alternative way of looking at the decoder circuit is to regard inputs A, B and C as address signals. Each combination of A, B or C defines a unique memory address.



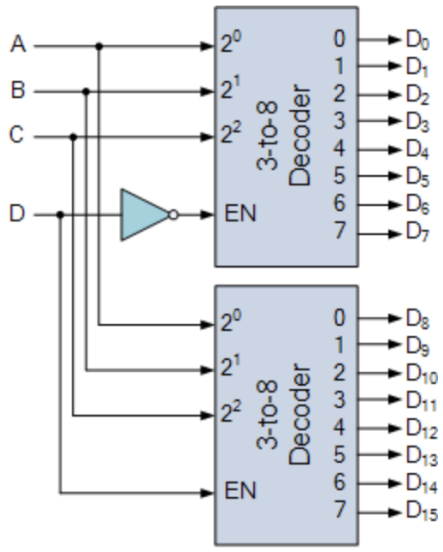
74LS138 Binary Decoder

We have seen that a 2-to-4 line binary decoder (TTL 74155) can be used for decoding any 2-bit binary code to provide four outputs, one for each possible input combination. However, sometimes it is required to have a **Binary Decoder** with a number of outputs greater than is available, so by adding more inputs, the decoder can potentially provide 2^n more outputs.

So for example, a decoder with 3 binary inputs (n = 3), would produce a 3-to-8 line decoder (TTL 74138) and 4 inputs (n = 4) would produce a 4-to-16 line decoder (TTL 74154) and so on. But a decoder can also have less than 2^n outputs such as the BCD to seven-segment decoder (TTL 7447) which has 4 inputs and only 7 active outputs to drive a display rather than the full 16 (2^4) outputs as you would expect.

Here a much larger 4 (3 data plus 1 enable) to 16 line binary decoder has been implemented using two smaller 3-to-8 decoders.

A 4-to-16 Binary Decoder Configuration



4-to-16 Line Decoder Implemented with two 3-to-8 Decoders

Inputs A, B, C are used to select which output on either decoder will be at logic "1" (HIGH) and input D is used with the enable input to select which encoder either the first or second will output the "1".

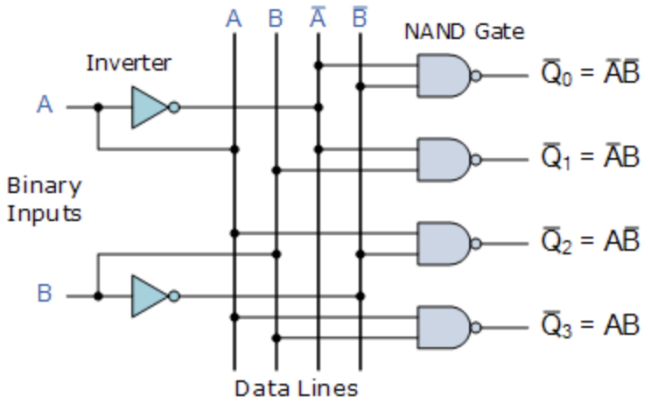
However, there is a limit to the number of inputs that can be used for one particular decoder, because as *n* increases, the number of AND gates required to produce an output also becomes larger resulting in the fan-out of the gates used to drive them becoming large.

This type of active-"HIGH" decoder can be implemented using just Inverters, (NOT Gates) and AND gates. It is convenient to use an AND gate as the basic decoding element for the output because it produces a "HIGH" or logic "1" output only when all of its inputs are logic "1".

But some binary decoders are constructed using NAND gates instead of AND gates for their decoded output, since NAND gates are cheaper to produce than AND's as they require fewer transistors to implement within their design.

The use of NAND gates as the decoding element, results in an active-"LOW" output while the rest will be "HIGH". As a NAND gate produces the AND operation with an inverted output, the NAND decoder looks like this with its inverted truth table.

2-to-4 Line NAND Binary Decoder



Truth Table

A	B	Q_0	Q_1	Q_2	Q_3
0	0	0	1	1	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0

Then for the NAND decoder, only one output can be LOW and equal to logic "0" at any given time, with all the other outputs being HIGH at logic "1".

Decoders are also available with an additional "Enable" input pin which allows the decoded output to be turned "ON" or "OFF" by applying a logic "1" or logic "0" respectively to it. So for example, when the enable input is at logic level "0", (EN = 0) all outputs are "OFF" at logic "0" (for AND gates) regardless of the state of the inputs A and B.

Generally to implement this enabling function the 2-input AND or NAND gates are replaced with 3-input AND or NAND gates. The additional input pin represents the enable function.

Memory Address Decoder

Binary Decoders are most often used in more complex digital systems to access a particular memory location based on an "address" produced by a computing device. In modern microprocessor systems the amount of memory required can be quite high and is generally more than one single memory chip alone.

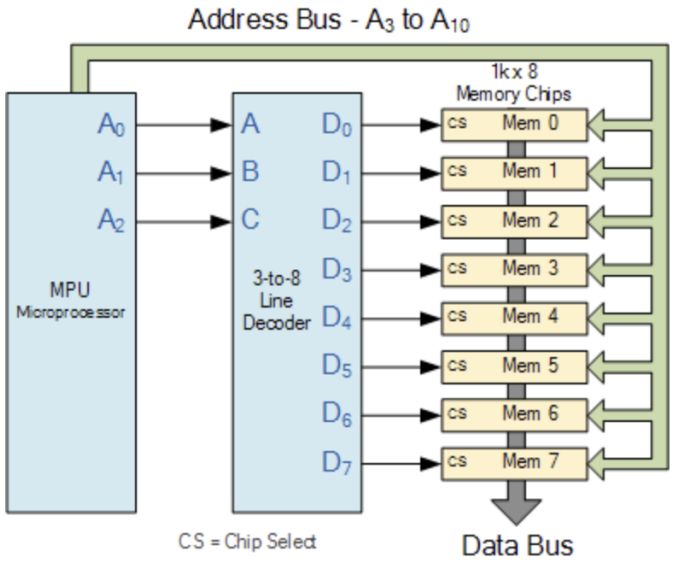
One method of overcoming this problem is to connect lots of individual memory chips together and to read the data on a common “Data Bus”. In order to prevent the data being “read” from each memory chip at the same time, each memory chip is selected individually one at time and this process is known as **Address Decoding**.

In this type of application, the address represents the coded data input, and the outputs are the particular memory element select signals. Each memory chip has an input called **Chip Select** or **CS** which is used by the MPU (micro-processor unit) to select the appropriate memory chip when required. Generally a logic “1” on the chip select (CS) input selects the memory device while a logic “0” on the input de-selects it.

So by selecting or de-selecting each chip one at a time, allows us to select the correct memory address device for a particular address location. The advantage of address decoding is that when we specify a particular memory address, the corresponding memory location exists ONLY in one of the chips.

For example, Lets assume we have a very simple microprocessor system with only 1Kb (one thousand bytes) of RAM memory and 10 memory address lines available. The memory consists of 128×8-bit (128×8 = 1024 bits) devices and for 1Kb we would need 8 individual memory chips but in order to select the correct memory chip we would also require a 3-to-8 line binary decoder as shown below.

Memory Address Decoding



The binary decoder requires only 3 address lines, (A_0 to A_2) to select each one of the 8 chips (the lower part of the address), while the remaining 8 address lines (A_3 to A_{10}) select the correct memory location on that chip (the upper part of the address).

Having selected a memory location using the address bus, the information at the particular internal memory location is sent to a common “Data Bus” for use by the microprocessor. This is of course a simple example but the principals remain the same for all types of memory chips or modules.