# Introduction to Digital Systems

This course is concerned with the architectures, circuits and components that make up modern digital electronic systems. In particular we will focus on circuits used in user programmable, microprocessor-based computers. We will use the same design tools and environment currently used by the digital system industry.

We will begin with an examination of the difference between digital and analog systems and the general characteristics of digital systems that are fundamental in the understanding of digital electronic circuits.

**What is a *Digital System* ?**

*Analog*
: of or relating to calculation by quantities which vary *continuously*.

*Digital*
: of or relating to calculation by quantities which vary by *discrete* units (units which vary from one another in well-defined *steps*).

*System*
a group of devices forming a network for distributing something or serving a common purpose. A simple system performs a single fixed function. A complex system can perform a variety of functions as selected by *CONTROL* inputs. It may perform *CONDITIONAL* execution of functions according to *SENSOR* (conditional) inputs:
IF <*condition*> THEN <*function*>

# Characteristics of Digital Systems

**Topics of Discussion**

- Data Representation
- Dynamic Range (accuracy)
- Errors: s*usceptibility, detection, correction*
- Modularity
- Suitability for *control* applications.
- Sequencing Events

## Data Representation

*Data is represented in a digital system as a vector of binary variables.*

In a digital system, i.e. a system in which data variables take on a finite number of discrete values, it is always possible to represent or code these values as a vector of *binary* variables. A binary variable corresponds to a binary digit and is called a *BIT*. A binary variable has only two possible values. It is standard practice to designate these two values with the symbols *0* and *1*. Other common symbols for binary variables are:

*0 == False == Low*
*1 == True == High*

**Electrical Representation of digital data**

Virtually all digital logic components in existance today are designed to generate *binary* signals only. Each binary state is represented by a range of voltages. The specific definition of logic levels in terms of voltage ranges can vary with different integrated circuit technologies, but for most circuits, can be defined as follows:

*Logic 0*
    0 - 0.8 volts
*Logic 1*
    2.0 - 5.0 volts

Any voltages between 0.8 volts (the upper limit of a logic 0) and 2.0 volts (the lower limit of a logic 1) are undefined. Digital components will not generate voltages in this undefined zone (unless they are faulty). Thus there can be no confusion between a logic 0 level and a logic 1 level because they are separated by this "no-fly zone".

In an electronic digital system, each *bit* is represented by a unique electrical signal. One of the implications of this is that in order to perform a function on a multi-bit variable, multiple circuits are needed. For example, to add two 16 bit numbers, 16 adder circuits are required - one for each pair of bits.

In an *analog* electronic system, data is represented by the magnitude of the signal voltage. For example, a value of 4 may be represented by a voltage of 4 volts, and a value of 7 by a voltage of 7 volts, etc. To perform a function only a single circuit is required. A single adder, for instance could add a 4 volt signal to a 7 volt signal and generate an 11 volt signal as a result. Where variables have a wide range of values, the circuits used must be able to handle a wide range of voltages. Accuracy may be compromised in order to facilitate a practical range of signal voltages. Analog circuits capable of performing functions on signal voltages greater than just a few tens of volts can be impractical. At the same time, very low signal voltages (less than a few millivolts) can be obscured or distorted by low level electrical noise that is always present in an electronic circuit.

**Digital Representation of Data**

For variables which must take on a range of values beyond that of a single bit, multiple binary variables can be *weighted* and grouped together.

Example: The variable COUNT whose range of values is to be 0 through 7, can be made up of the binary variables C2, C1 and C0 and weighted as 2**2, 2**1, and 2**0 respectively.

| COUNT | C2 | C1 | C0 |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 2 | 0 | 1 | 0 |
| 3 | 0 | 1 | 1 |
| 4 | 1 | 0 | 0 |
| 5 | 1 | 0 | 1 |
| 6 | 1 | 1 | 0 |
| 7 | 1 | 1 | 1 |

When, in an electronic digital system, groups of signals are used to represent multi-bit variables, they are often referred to collectively as a *bus*. The signals C2, C1, and C0 depicted above, could be referred to as the COUNT bus.

## Dynamic Range

*Digital Systems can provide accuracy (dynamic range) limited only by the number of bits used to represent a variable*

There are many practical examples of improved dynamic range of digital systems over analog systems. Compare for example, the improved dynamic range of compact disc systems (digital) over cassette magnetic tape (analog). With magnetic tape there is a restricted range over which the tape can be magnetized. Outside of this range, the magnetic particles of the tape will not be altered. The result of this is that a range in volume between quiet and loud passages is significantly restricted. With compact discs, digital data is recorded as 16 bit samples. This gives a range of 2**16 or >65,000 discrete values. For this reason, CDs can sound much "crisper" than magnetic tapes.

## Errors in Digital Systems

### Susceptibility

*Digital systems are less prone to error than analog systems.*

Since binary variables in a digital system are represented by a range of electrical voltages, binary variables can still be distinguished even when altered slightly by electrical noise, or differences in electrical characteristics of components due to changes in temperature or tolerances during the manufacturing process.

In an analog system, these effects directly interfere with the integrity of the data. For example, consider the presence of media induced "tape hiss" present in a magnetic cassette tape recording. In contrast, there is no such extraneous noise present in a CD recording.

### Detection and Correction

*Data representation in a digital system is suitable for error detection and correction.*

The numerical representation of data in a digital system allows for a number of error detection and correction schemes to be implemented. Most of these techniques involve increasing the size of the data word to include some sort of "validity" bits.

Parity schemes involve adding a single bit to the data word to indicate whether the total number of bits in the word is odd or even. Parity schemes are capable of detecting single bit errors only and correction is not possible because it is not possible to know which bit of data is in error. Double bit errors go undetected.

ECC (Error Correction Code) schemes involve adding multiple parity bits to the data word. These parity bits are sometimes called "hamming" bits. The distribution of data bits used to generate each hamming bit make it possible to both detect and correct single bit errors and to detect but not correct double bit errors.

In a digital system, there is no electrical distinction between the most significant bit (msb) and the least significant (lsb) of a multi-bit variable. Therefore, any error that does occur, perhaps because of some electrical disturbance, may cause a significant problem. For this reason, error detection and correction capabilities may be necessary if data integrity is critical.

In an analog system, error detection is next to impossible. Only errors that cause the data to fall outside a predetermined range of values can be detected. In any event, errors cannot be corrected, although their effects can be minimized. In an analog system, the magnitude of an error is generally proportional to the error cause. For example, a small crease in a magnetic tape will only cause a slight "pop" in the recorded sound.

## Modularity

*Digital systems are designed in a hierarchical manner using re-useable modules.*

Complex functions can generally be broken down in to simpler subtasks which can be executed in sequence and/or concurrently. There is an analogy here with the way software programs are constructed. Complex tasks are broken down into subroutines which themselves can often be broken down into simpler subtasks. By passing different parameters to the subroutines, they can be re-used (called) by different parts of the program.

Digital circuits are designed in a similar fashion. Circuits are formed with networks of compatible standard modules which perform a variety of simpler subtasks. Compatibility between modules exists because of the standard electrical representation of binary data. Commonly used complex functions have been implemented as off-the-shelf integrated circuits (ICs) so that the circuit does not have to be designed using the simplest digital functions listed below.

The simplest logical functions are:

*AND, OR,* and *NOT*

These simple functions are often called "gates".

Any digital function - simple or complex - can be implemented using only these three simple functions. In fact, AND can be implemented using a network of NOTs and NORs; OR can be implemented by a network of NOTs and ANDs. We will look at sample implementations of arithmetic circuits, data shifting circuits, data routing circuits and others.

These three functions have been implemented as standard modules in the form of "integrated circuits". More complex functions have also been implemented as ICs although internally, they are also composed of NOT/AND/OR networks.

## Microprocessor versus circuit based implementation.

Implementing digital functions of even moderate complexity may require that a choice be made between a custom digital circuit or the use of a microprocessor. A very wide range of microprocessor chips are available that include memory and input/output facilities. These complete computers are called microcontrollers, and a device that uses one to implement a digital function is called an "embedded system". These microcontrollers are packaged in integrated circuits having from 8 to 40 pins and many cost just a couple of dollars. Using a microcontroller in a design has many benefits, including short design time, flexibility, and size.

In cases where a microcontroller cannot provide the required performance, a mixture of a microcontroller and some custom circuitry can be an effective solution.

In this course, we will have the opportunity to complete a design that includes an embedded processor and custom circuitry packaged in a single integrated circuit called a "Field Programmable Gate Array" or FPGA. We will use a state-of-the-art computer based design environment that will allow us to enter the design in a variety of forms, run simulations, and test the final product.