LECTURE NOTES OF MANAGEMENT AND ANALYSIS OF PHYSICS DATASET

Comment.

Edited By

Author

The University of Padua

Contents

Introduction

Lecture 1.
Wednesday 9^{th} October, 2019.
Compiled: Sunday 16^{th} February, 2020.
Prof. Collazuol

Lecture 2.
Tuesday 15^{th} October, 2019.
Compiled: Sunday 16^{th} February, 2020.
Prof. Collazuol

Lecture 3.
Wednesday 16^{th} October, 2019.
Compiled: Sunday 16^{th} February, 2020.
Prof. Collazuol

Chapter 1

Introduction to Trigger and Data Acquisition

Lecture 1.
Tuesday 22nd October, 2019.
Compiled: Sunday 16th February, 2020.
Prof. Meschi Alice

1.1 Sensors, detectors and experiments

1.1.1 Sensors

A **transducer** is a device that converts energy from one form to another. Transducers can be categorized by which direction information passes through them; for instance a **sensor** is a transducer that receives and responds to a signal or stimulus from a physical system.
In modern parlance, the sensors is "a device, module, machine, or subsystem whose purpose is to detect events or changes in its environment and send the information to other electronics". A sensor is always used with other electronics.
In practice, any device that detects or measures a physical event or quantity and transforms this event or quantity into another that is "easier" to perceive and/or measure, *sensors* and *transducers* can be often exchanged/confused.
In most cases today, the final quantity is an electrical signal (either steady or transient). In what follows, we will be dealing mostly with transducers that produce an electrical signal, in most cases a *pulse*.
A sensor's sensitivity indicates how much the sensor's output changes when the input quantity being measured changes. They are usually designed to have a small effect on what is measured; making the sensor smaller often improves this and may introduce other advantages.

1.1.2 Detectors

In real life, we often deal with a complex of (one or more) sensors or transducers, not necessarily homogeneous. We refer to this complex as a **detector** and this often includes the electronics used to read out the information about the physical quantity or event. In nuclear physics and high energy physics, when we say "detector", we almost always mean "ionizing particle detector", as the ion chamber in Figure 1.1.
An *ionization chamber* measures the charge from the number of ion pairs created within a gas caused by incident radiation. It consists of a gas-filled chamber with two electrodes; known as anode and cathode. A voltage potential is applied between the electrodes to create an electric field in the fill gas. When gas between the electrodes is ionized by incident ionizing radiation, ion-pairs are created and the resultant positive ions and dissociated electrons move to the electrodes of the opposite polarity under the influence of the electric field. This generates an ionization current which is measured by an electrometer circuit.
Sometimes for "detector" we mean a whole experiment ("the CMS detector").

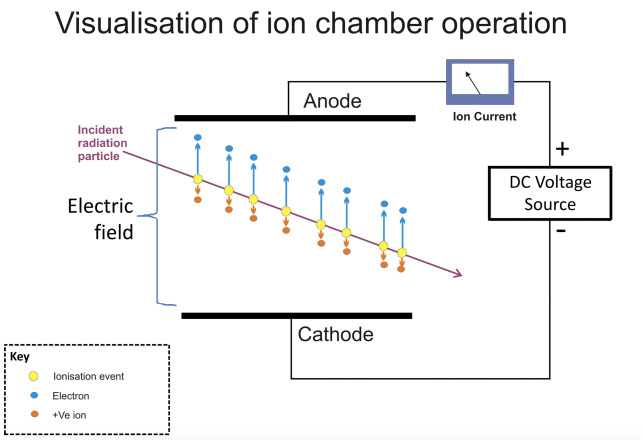


Figure 1.1: Schematic diagram of parallel plate ion chamber, showing creation of ion pairs, and drift of ions due to electric field. Electrons typically drift 1000 times faster than positive ions due to their smaller mass.

Detectors of the past often used analog (or sensorial) means to measure/register a phenomenon. For example, the *visual or aural observation*, that often involved counting the occurences of some phenomenon. Also *photography* was often used for more complex observations (emulsion experiments, bubble chambers).
A computer does better counting and recording of information, once the information is in digital form.
Detectors in other areas exploit different physics, however the principles remain the same.

1.1.3 Example: Rutherford scattering

Rutherford scattering is the elastic scattering of charged particles by the Coulomb interaction. Alpha particles from a radioactive source, as Americium 241, constituted a beam that strike a thin gold foil. They passed through that gold foil, which is about 1.5 μm in thickness. Then, they are detected by one of the two detectors set up behind the foil. One of the detectors counts particles going straight ahead which is the majority, and we count a rate between one and two thousand per second. The second counter we can move to measure the scattering rate as a function of angle.
The experiment Rutherford and his colleagues carried out was very laborious than the one described. In fact, the detectors were tiny fluorescent screen (see Figure 1.2). Alpha particles produce a tiny, but visible flash of light when they strike a fluorescent screen, therefore they were detected by looking just by eye for flashes of light.
Surprisingly, alpha particles were found at large deflection angles and some were even found to be back-scattered. This experiment showed that the positive matter in atoms was concentrated in an incredibly small volume and gave birth to the idea of the nuclear atom.

1.2 Modern data acquisition: the digital way

1.2.1 Role of Trigger and DAQ

The **trigger** process the signals generated in a detector and save (only) the interesting information on a permanent storage medium. It uses criteria to rapidly decide which events in detector to keep when only a small fraction of the total is recorded. For instance, in particle physics, since experiments are typically searching for "interesting" events (such as decays of rare particles) that occur at a relatively low rate, trigger

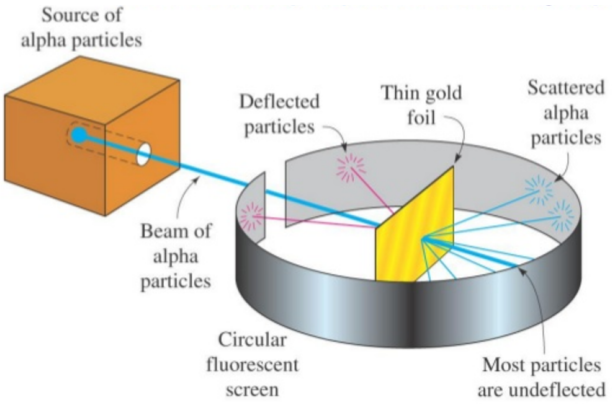


Figure 1.2: Graphic representation of the Rutherford scattering.

systems are used to identify the events that should be recorded for later analysis.
Data Acquisition is the process of sampling signals that measure real world physical conditions and converting the resulting samples into digital numeric values that can be manipulated by a computer. Data acquisition systems, abbreviated by the acronyms DAQ, typically convert analog waveforms into digital values for processing. They are usually controlled by software programs.
Modern DAQ is all about digital information. However, physics is not digital, in fact sensors produce analog signals that must be treated and interpreted. There is a lot of physics (and math, and technology) that you only learn in DAQ and trigger.

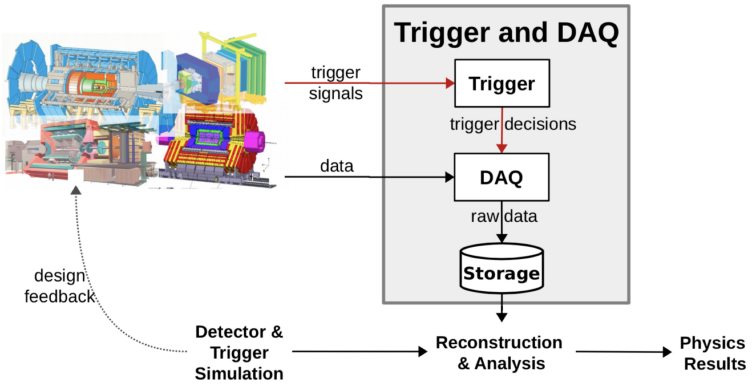


Figure 1.3: General scheme of data acquisition.

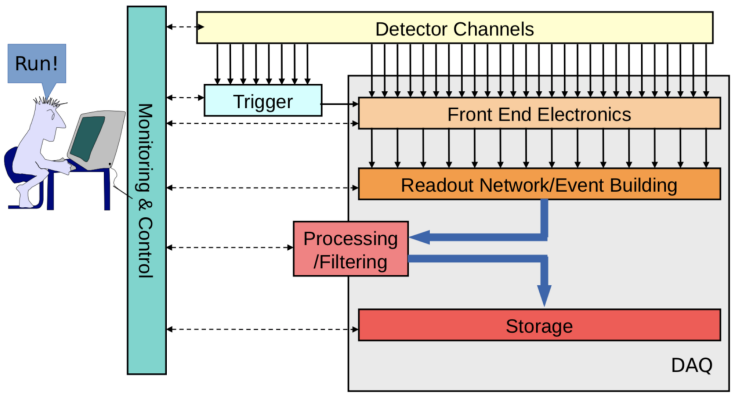


Figure 1.4: Example of a modern collider Trigger/DAQ.

Now, we consider the example of measuring temperature at a fixed frequency. In this case, the ADC (Analog to digital converter) performs analog to digital conversion, that is the digitization (our front-end electronics), while the CPU does readout and processing. The system is clearly limited by the time τ to process an event (ADC conversion + CPU processing + Storage). Therefore, the DAQ maximum sustainable rate is simply the inverse of τ: if $\tau = 1\,\mu$s, we have $R = 1/\tau = 1\,$kHz. If the events are asynchronous and unpredicatable (beta decays studies), a physics trigger is needed: a discriminator generates an output signal only if amplitude of input pulse is greater than a given threshold. There is a delay introduced to compensate for the trigger latency.

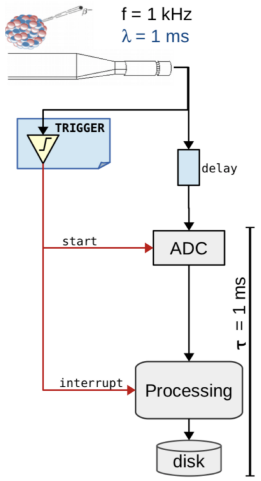


Figure 1.5

Let us say for the moment an ADC is a device that converts some analog quantity (e.g. a voltage) into a binary-encoded number with n bits.

1.2.2 Microprocessor architecture

Computer systems generally consist of three main parts: the central processing unit (CPU) that processes data, memory that holds the programs and data to be processed, and I/O (input/output) devices as peripherals that communicate with the outside world.

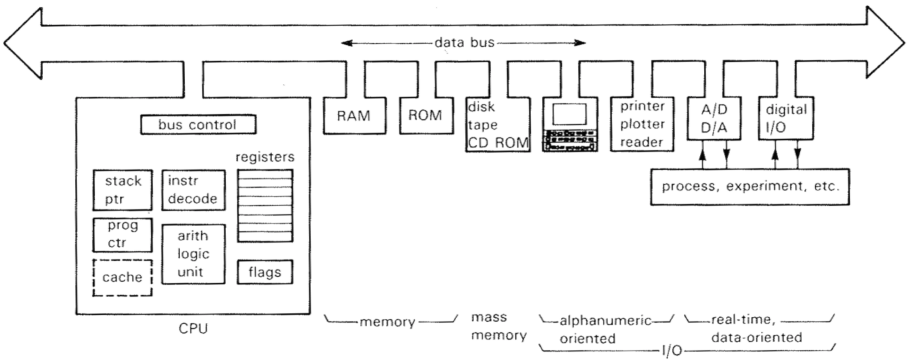


Figure 1.6: Scheme of a microprocessor architecture.

In most traditional computer architectures, the CPU and main memory tend to be tightly coupled. A **microprocessor** conventionally is a single chip which has a number of electrical connections on its pins that can be used to select an "address" in the main memory and another set of pins to read and write the data stored at that location. In most cases, the CPU and memory share signalling characteristics and operate in synchrony. The bus connecting the CPU and memory is one of the defining characteristics of the system, and often referred to simply as the system bus. The microprocessor is a multipurpose, clock driven, register based, digital integrated circuit that accepts binary data as input, processes it according to instructions stored in its memory and provides results as output. Microprocessors contain both combinational logic and sequential digital logic. Microprocessors operate on numbers and symbols represented in the binary number system.
The integration of a whole CPU onto a single or a few integrated circuits greatly reduced the cost of processing power.

In general:

- **CPU**: does computation on words.

  It fetches instructions from memory (the "program"). Instructions are bit codes corresponding to an operation (part of an "instruction set"). They are first decoded and then fed to an Arithmetic and Logical Unit (ALU) that performs the operation on data contained in registers (e.g. add, complement, compare, shift, move...). A program counter keeps track of the current location in the program being executed.

  Data (as instructions) are fetched (usually) from memory over a bus. A bus controller handles the communication with memory and other I/O peripherals (such as a DAQ board, for example).

  Modern CPUs have a more or less large "cache" – fast memory that contains recently or frequently accessed data for quick retrieval (not requiring access to the memory bus).

  The stack is a portion of memory that works like a LIFO: there are two fundamental instructions PUSH and POP to move data to and from the stack, and a stack pointer always pointing at the top.

  This simplified view is common (give or take few parts) to many different architectures, including those specific to DAQ (the CPU could be just an "intelligent bus master").

- **Memory access**.

  The data transfer can be controlled by the *programmed data transfer*. A software routine residing in memory requests the peripheral device for data transfer. Programmed data transfers are generally used when a small amount of data is transferred with relatively slow I/O devices.

  Another possibility is *direct Memory Access (DMA) or "block" transfer*. In this mode, the data transfer is controlled by the peripheral device. DMA transfer is used when a large block of data is to be transferred.

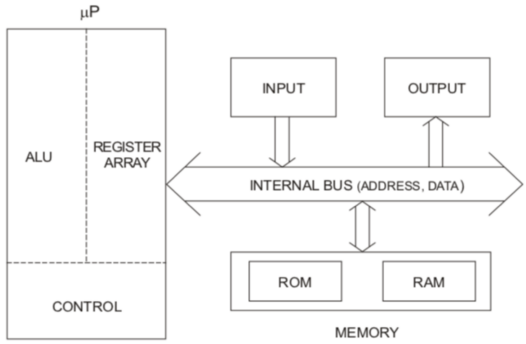


Figure 1.7: Scheme of a memory in a microprocessor.

- **Addressing memory**. The address space is the total number of unique addresses available to the CPU. Different addressable devices are "mapped" to different unique address ranges for access over the bus.

  Example 1 (Mapping address). The processor can usually address a memory space that is much larger than the memory space covered by an individual device or memory chip. In order to splice a device into the address space of the processor, decoding is necessary. For example, the Intel 8088 had 20-bit

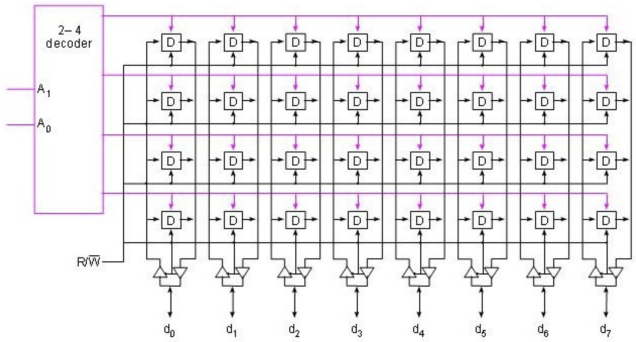


Figure 1.8: Scheme of an address space.

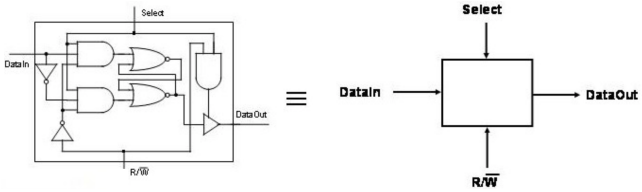


Figure 1.9: Address.

addresses for a total of 1MB of memory address space. However, the BIOS on a 2716 EPROM had only 2KB of memory and 11 address pins. A decoder was used to decode the additional 9 address pins and allow the EPROM to be placed in any 2KB section of the 1MB address space.

- **Bus**. Buses can be parallel buses, which carry data words in parallel on multiple wires, or serial buses, which carry data in bit-serial form. An example of parallel bus is the VME bus. Besides modular electronics, it was the base of Sun-2 computer arch. in the 80s. In the example shown in Figure 1.10, we illustrate the write bus cycle, that is a sequence of operations in memory write.

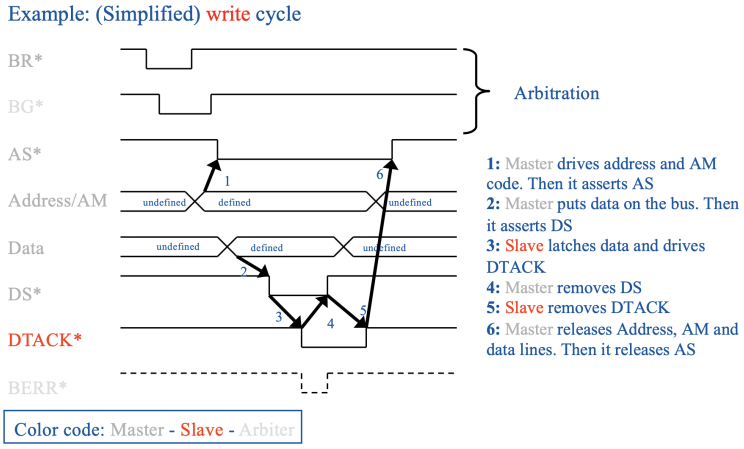


Figure 1.10: Example of write cycle.

1.2.3 Back to DAQ

What does it mean that the CPU "processes" data ? The readout of one sensor is encoded in n bits. If multiple channels, "atom" of information is m bits for channel id + n bits for e.g. ADC counts (for example representing energy deposited in one sensor). We need to "**decode**" the information packed to feed into e.g. an algorithm (usually written in a high-level language working with standard data types). Channel

id usually corresponds to a *position in space*, while ADC counts must be translated into a physical quantity, usually by applying a *calibration*.
In complex experiments, many channels are received by multiple electronic boards interconnected by a data bus. A computer may not be the most convenient form factor for this, we use modular electronics. In this case, in an architecture similar to the one discussed before, at least one particular element on the BUS is a CPU. It is common to use the bus (e.g. VME) to collect data from multiple boards in a single portion of memory.
In modern experiments, different boards from portions of the detector are read out by CPUs. We want to combine all the portions corresponding to the same "event" in the memory of a single CPU for processing. For doing that, we need a mechanism to associate all data corresponding to the same event (e.g. a bunch crossing in a collider, a gamma ray shower in a telescope array...). This process is called **event building** and it is usually performed by a distributed program through a **switched network**.

Decoding

In DAQ and trigger it is very useful to be able to read hex "at a glance". The bit-wise operations are the most basic ingredient of "unpacking" data. Electronics engineers tend to pack data a lot, often using odd conventions. For example, the most common way to encode negative numbers is 2's complement. An example of decoding raw packets is shown below:

```
>>> data = 0x00610002
>>> bin(data)
'0b11000010000000000000010'
>>> module_mask = 0x0000ffff
>>> det_mask = 0xffff0000
>>> detector = (data & det_mask) >> 16 >>> detector
97
>>> hex(detector)
'0x61'
>>> module = (data & module_mask) >>> module
2
```

1.3 Back to the sensor: the analog way

1.3.1 Photomultiplier Tube

Photomultiplier tubes (PMTs for short) are extremely sensitive detectors of light in the ultraviolet, visible, and near-infrared ranges of the electromagnetic spectrum. They are constituted by a photocathode, several dynodes, and an anode.

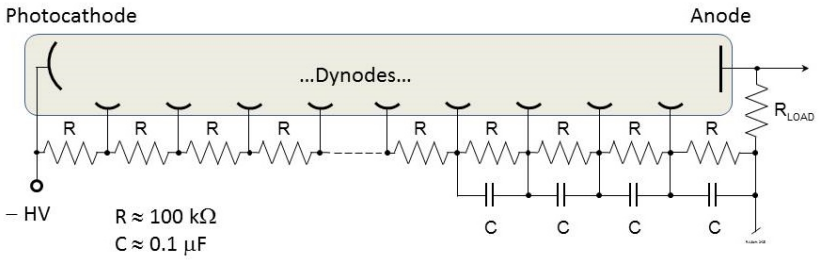


Figure 1.11: Photomultiplier representation.

Incident photons strike the photocathode material. Electrons are ejected from the surface as a consequence of the photoelectric effect. These electrons are directed by

the focusing electrode toward the electron multiplier, where electrons are multiplied by the process of secondary emission. The electron multiplier consists of a number of electrodes called dynodes. Each dynode is held at a more positive potential, by $\sim 100\,\mathrm{V}$, than the preceding one. A primary electron leaves the photocathode with the energy of the incoming photon, then a small group of primary electrons is created by the arrival of a group of initial photons. The primary electrons move toward the first dynode because they are accelerated by the electric field. Upon striking the first dynode, more low energy electrons are emitted, and these electrons are in turn accelerated toward the second dynode. The geometry of the dynode chain is such that a cascade occurs with an exponentially-increasing number of electrons being produced at each stage. The last stage is called the anode. This large number of electrons reaching the anode results in a sharp current pulse that is easily detectable, for example on an oscilloscope. The typical overall gain is $\sim 10^6$. The dark current is thecurrent flowing in the PMT without light, that represents the noise of the system.

1.3.2 Pulse signals

Since most sensor produce an electrical pulse in response to the phenomenon they measure, it is important to define and understand a certain number of concepts concerning a pulse signal.
A **pulse** in signal processing is a rapid, transient change in the amplitude of a signal from a baseline value to a higher or lower value, followed by a rapid return to the baseline value.

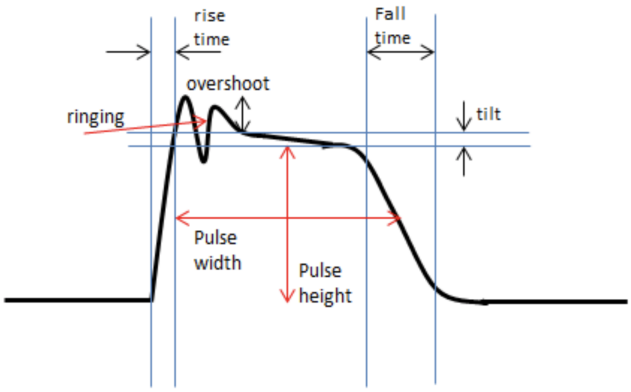


Figure 1.12: Pulse signal.

1.3.3 Measuring with a scope

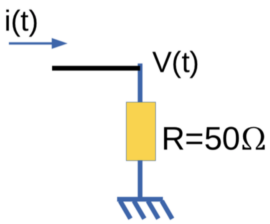
Let us do an approximate Q measurement using an oscilloscope. Consider a simple circuit as in Figure 1.13a. The charge is therefore:

$$Q = \int i(t)\,\mathrm{d}t = \frac{1}{R}\int V(t)\,\mathrm{d}t \tag{1.1}$$

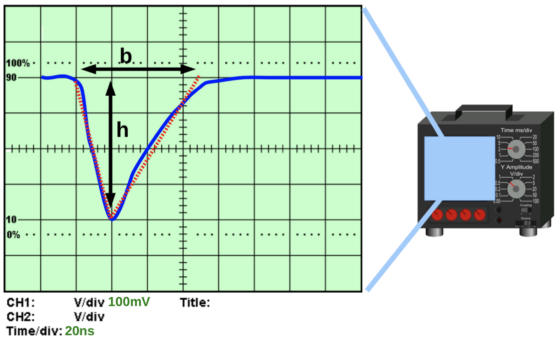
For calculating this quantity, we do a linear approximation of the exponential decay of the signal, namely:

$$Q \approx \frac{1}{R}\frac{bh}{2} \tag{1.2}$$

It would be much more convenient to have a direct electronic measurement: a data acquisition system. However, the oscilloscope method is still fundamental. It allows for the validation of your DAQ yes, you should never trust it a priori! Always remember the oscilloscope also has limitations.



(a) Circuit.



(b) Signal in the oscilloscope.

1.3.4 Discriminator

How does a "discriminator" work? A **discriminator** is an electronic circuit that has an output voltage only when the amplitude of the input pulses exceeds a predetermined value. For instance, in the context of light detection,if we are using a photodiode, the discriminators could be attempting to activate a circuit element if the light they are detecting is above a certain threshold. Combining signals from multiple discriminators may require compensating for the different length of the signal paths.
For instance, *constant fraction discriminator* (CFD) is an electronic signal processing device, designed to mimic the mathematical operation of finding a maximum of a pulse by finding the zero of its slope. Typical input signals for CFDs are pulses from plastic scintillation counters, such as those used for lifetime measurement in positron annihilation experiments. The scintillator pulses have identical rise times that are much longer than the desired temporal resolution. This forbids simple threshold triggering, which causes a dependence of the trigger time on the signal's peak height, an effect called *time walk* (see Figure 1.13). Identical rise times and peak shapes permit triggering not on a fixed threshold but on a constant fraction of the total peak height, yielding trigger times independent from peak heights.

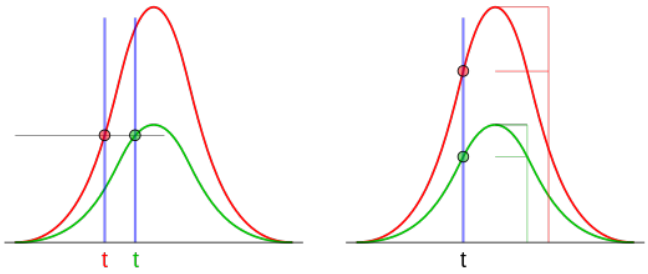


Figure 1.13: Comparison of threshold triggering (left) and constant fraction triggering (right).

1.3.5 SCA principle

A single channel analyzer (SCA) produces an output logic pulse on the condition that the peak amplitude of its input signal falls within the pulse-height window that is established with two preset threshold levels.

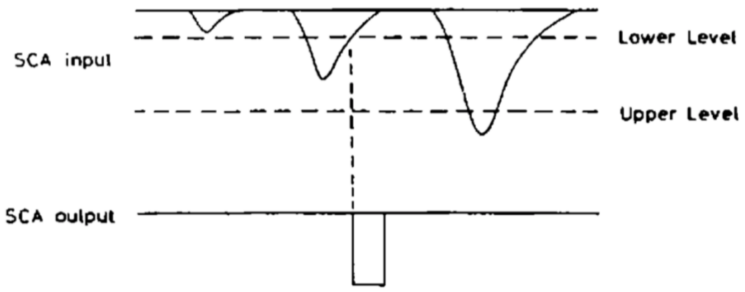


Figure 1.14: Basic operation of a SCA: only signal with amplitudes fall within the window defined by the upper and lower level threshold trigger a signal.

1.4 More of analog circuit

We will no go back to the simple experiment of some kind of particle relavated by a scintillator. We first introduce two contents about analog signal. The total charge is proportional to the number of photons that is proportional to the charge. When we discuss analog pulses we always consider the pulse.
The discriminator generates a pulse, but if you want to integrate the signal, you need more than a discriminator, otherwise you miss parts of the charge. Thanks to the delay, the pulse arrive before the integrating of the pulse. Fastest way to digitize: see the unit on the screen of the oscilloscope, in order to digitize the voltage compare the voltage with that unit.
The Flash ADC is the fasten and simple analog digital convertor. Aim: measure current. Therefore it is not sufficient to digitize the pulse and do a sampling of the pulse, but first it is necessary to integrate the pulse and then integrate the voltage. We use an analog integrator circuit.
Using analog signal, we may pay attention to the response.

1.4.1 Summary

We can write the response function as an attenuation of a function of frequencies.

1.5 Analog to Digital Converter

The flash ADC is divided in three parts. A voltage input, a differential comparator that will give a high if there is a voltage, a low if there not. See the results in the violet table. This is why is called thermometer code (?). One-hot encoding: we convert the binary code into one-hot by removing all the 1 if there is a sum 1 with 1. Clearly, you would like to have a lot of bits. The number we want depend on the speeds. Bipolar transistor manage more current. For a flash adc it is important the resolution. Because your unit is a bit, the maximum value can you commit it is half of the value of the bit. Flash ADC has a constant resolution on the valid inpout range. There is also a power scale with a large dynamic range. Normally, nobody makes discrete components ADC.

Lecture 4. *Wednesday 30^{th} October, 2019. Compiled: Sunday 16^{th} February, 2020. Prof. Meschi*
Lecture 4. *Tuesday 5^{th} November, 2019. Compiled: Sunday 16^{th} February, 2020. Prof. Collazuol Marco*

Chapter 2

Digital circuits

We will see a division on the operations of digital circuits, done by the type of operations that they permit:

- **Asynchronous operations**: they serve for diverse specific processing that do not require using a clock, but their problem is that the mismatch between delays can impair the results;
- **Synchronous operations**: the timing is controlled by a clock, and the period must be such as to accommodate the most time consuming processing section. Often the circuit includes sections where processing is performed in multiple clock periods;
- **Sequential operations**: a sequential system relies on a sequence of actions, required to occur in the **right order**. It produce output on the basis of both input and output. They are needed when memory of the past actions is required;
- **Combinational operations**: the output depends only on input. They can be always contructed with gates.

We can see in Figure 2.1 an example of the two types of circuits: a combinatorial and a sequential one.

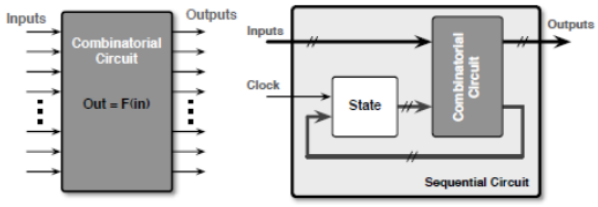


Figure 2.1: On the left an example of combinatorial circuit, where the output depends only on the inputs. On the right a memory circuity, that is essentially a sequential circuit because the output is used with the input to determine the output.

2.1 Ripple counter

A ripple counter is also knonw as an asynchronous sequential circuit. It is asynchronous because the output signals are NOT synchronized to a single clock signal, since they are many clock signals, and sequential because its current output value (or state) depends on previous output values in sequence.
We will now use the names defined by the Figure 2.2. $Q0$ value is first inverted (triangle) and then used as D input on the next clock cycle. The FF is triggered by

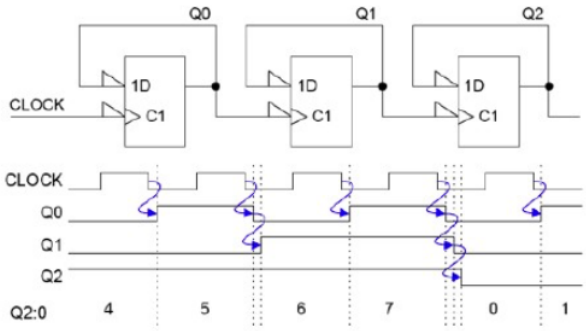


Figure 2.2: On the higher part of the figure we can see the circuit of the ripple counter: the triangles are inverters. On the lower part we can see the time evolution of the states: we can note the delays. In table 2.1 we can see explicitly the counting.

the falling edge of the clock, so the output changes on the falling edge of the clock. $Q0$ is therefore changing at half the rate of CLOCK, hence this flipflop acts as a divide-by-2 circuit. The $Q0$ signal is used as input for the next FF, and so $Q1$ is toggling (being activated) at half the frequency of $Q0$. The circuit is effectively a binary counter.
To analyze it in a more general way we can state that the output of each flip flop is given to the data input (D-FF).

State	$Q2, Q1, Q0$	Number
4	100	4
5	101	5
6	110	6
7	111	7
0	000	0
1	001	1

Table 2.1: Table of counting connected to the Figure 2.2. A 1 correspond to an high line and a 0 to a low line.

This is a simple finite state machine because it has $2^3 = 8$ states which cycles through in a sequence. Adding more flip flops in the same configuration will increase our counting power with a 2^n possible states, with n the number of FF.
Every pulse is providing a change in the circuit. Every time we have a falling edge the state changes, with some delays. We can estimate the delay as $1ns$ each time, so the delay between the pulse of the clock and the change of Q2 is $3ns$.

2.2 Synchronus counter

A far better approach is to use the FF toghether as a group, in parallel, and clock them with the SAME clock as shown in Figure 2.3. The logic block is a combinatorial circuit which computes the next D value from the current Q. The relationship between the two is simple: $D = Q + 1$.
The good news are that this circuit is faster than the ripple and the delay is constant instead of dependent on the size of the counter. It is $1ns$ for the Q and $2ns$ for the D. There are no delays between the updates of the three Q or between the three D.

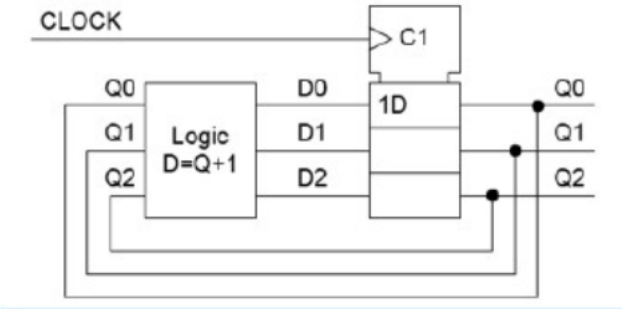


Figure 2.3: Synchronous counter. One single clock is used to control all the three FF. The output is given to the logic block and then used as input.

2.3 Combinatorial logic functions

Logical funciton can be written in more than one way, but there is one way that optimizes the number of gates used. Minimizing the number of gates required is not always a priority: speed or logical clarity almost always have higher priority. In FPGA logical function are implemented in the well defined canonical form.

2.3.1 Miniterms and canonical form

Let us consider an arbitrary logical function of n variables $F(A_1, ..., A_n)$, and let us further consider n-term products of the form $\prod_i A_i$ in which each variable is represented once, either negated or not Such products of this form are called a **miniterms**. For example in the case of two variables (A,B) the 4 miniterms are:AB,$A\overline{B}$,$\overline{A}B$,$\overline{AB}$. The number of miniterms is 2^n and the function F is defined on the set of its miniterms.
We can further divide the miniterms of F in two types of terms:

- T-terms, corresponding to the terms for which $F(term) = 1 = TRUE$
- F-terms, corresponding to the terms for which $F(term) = 0 = FALSE$

We have that $n_T + n_F = n$. It can be demonstrated that F **is the sum of its T-terms and this representation is called canonical form**, or the negated sum of F-terms.
You can also do it with maxiterms, defined by the sum of the variable negated or not as above: you take the product of the true terms.
NOTE: Given a basic set of operation (for example {AND, OR, FANOUT, EXCHANGE}) then you can build any function of m output on n input $f : \{0,1\}^n \rightarrow \{0,1\}^m$

2.4 Practical examples of combinatorial logic circuits

2.4.1 Sum between bits

To add two binary words we need, at each position:

- to know whether there is a carry from the next-lowest position (carry-in);
- to generate the carry for the next-highest position (carry-out).

We so define: **A**,**B** the bits of the word to be added, C the carry-in, Σ the sum and K the carry-out.

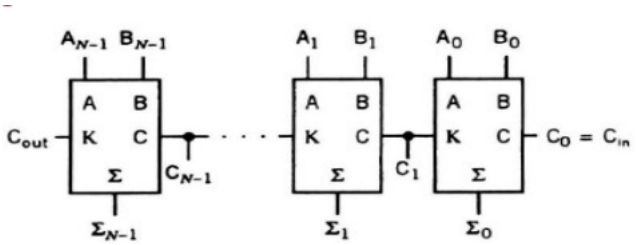


Figure 2.4: Adder of N-bit words with N 1-bit summer. The A_i,B_i represent the inputs, the C the carry-in and the K the carry out.

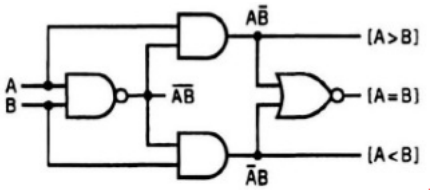


Figure 2.5: Circuit of the 1-bit magnitude comparator. The logical function are listed in table 2.2.

Σ is the low-order bit of the sum $A + B + C$ and it is 1 if and only if or only one addend is 1 or all three are 1. We can so interpret the sum as a logical variable:

$$\Sigma = A\bar{B}\bar{C} + A\bar{B}\bar{C} + \bar{A}\bar{B}C + ABC$$

We can interpret similarly **K**: it is 1 only if any pair of addends is 1. Writing it by a sum of logical variables we get:

$$K = AB + BC + AC$$

Note that the term $ABC = 1$ is already implied by the other three terms.
If we want to add two N-bit words we can chain N 1-bit adders together, as shown in Figure 2.4. The adder for the lowest-order bit accepts a carry-in Cin from a possible lower-order word, and the adder for the highest-order bit generates a carry-out Cout that can feed a possible higher-order word. It is important to note that each 1-bit counter create a delay, and so the complessive delay can be considered excessive.
In FPGA we will get access to **Aritmetic Logic Units** (ALU), a multipurpose integrated circuit capable of performing various arithmetic and logic operations.

2.4.2 Magnitude comparator for unsigned integers

First we construct the 1-bit magnitude comparator, that will have three outputs : =,<,>. The conditions are listed in Table 2.2 and the circuit is represented in Figure 2.5 A 2-bit comparator is then constructed as follows. We use the example of the

Condition	Logical operation
$A > B$	$A\bar{B}$
$A = B$	$AB + \bar{A}\bar{B}$
$A < B$	$\bar{A}B$

Table 2.2: Logical representation of the conditions for the 1-bit magnitude comparator. The circuit is represented in Figure 2.5

greater, so we want to see if $A_1A_0 > B_1B_0$. We note that a sufficient condition is $A_1 > B_1$, but if that is not satisfied we need $A_1 = B_1$ and $A_0 > B_0$ simultaneously. We list in Table 2.3 the various decompositions.

Condition	Decomposition in 1-bit cond	Logical operation
$A_1A_0 > B_1B_0$	$A_1 > B_1 + (A_1 = B_1)(A_0 > B_0)$	$A_1\bar{B_1} + (A_1B_1 + A_1\bar{B_1})\bar{A}B$
$A_1A_0 = B_1B_0$	$(A_1 = B_1)(A_0 = B_0)$	$(A_1B_1 + A_1\bar{B_1})(A_0B_0 + A_0\bar{B_0})$
$A_1A_0 < B_1B_0$	$A_1 < B_1 + (A_1 = B_1)(A_0 < B_0)$	$\bar{A_1}B_1 + (A_1B_1 + A_1\bar{B_1})A\bar{B}$

Table 2.3: Table of truth for the 2-bit adder.

NOTE: there are many slides between these and the sequential circuits part, but I am quite sure we didn't do them.

2.5 Sequential circuits

We recall what we have said in the digital circuit section, but this time applied in particular to the sequential circuits:

- **Synchronous sequential logic**: the time at which the transition between states occurs is controlled by a common clock signal and the changes in all variables occur simultaneously;
- **Asynchronous sequential logic**: the state transitions occur independently of any clock, and normally depend on the time at which input variables change, but the outputs do not necessarily change simultaneously:
- **Clock**: a clock signal is a square wave of a fixed frequency and it is used to trigger state transitions at fixed times in synchronous circuits.

2.5.1 Flip-flop and latches

FF and latches[1] are the fundamental unit of sequential circuits. They have two stable states and are essentially 1-bit storage devices: the output can be set to 1 or 0 depending on the input and, even if the input are changed, the output remains the same. The two complementary output are usually identified by Q **and** $\bar{Q}$.
The standard distinction between latches and FF is the following:

- **Latches**: FF sensitive to levels (high or low).
- **Flip-flops**: FF that are sensitive to transitions (signal edges).

They are needed to execute sequential logic, since they keep track of the past actions, they are memory. We will focus on FF obtained by cross-coupling logic gates and we will assume that all gates presents the same delay delay t.
FF come in many flavor but we will use only the d-flipflop in the FPGA lab. They are defined by the way they change the output.

Lecture 5.
Wednesday 6^{th} November, 2019.
Compiled: Sunday 16^{th} February, 2020.
Prof. Collazuol

2.5.2 Asynchronous sequential circuits

Set-Reset FF

They are a type of asynchronous FF. We will consider only the RS NAND FF in Figure 2.6, but all the statements are equivalent for the NOR with the appropiate changes. We call the two different input SET and RESET and call them **active** if they are low (=0) and **inactive** if they are high (=1). The two outputs, function of

[1] For you, lazy italian that doesn't want to type that on Google translate, it means "serrature"

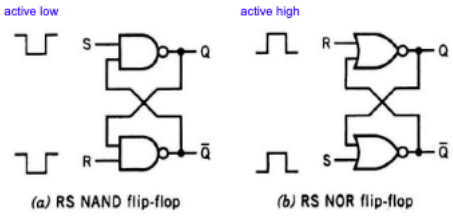


Figure 2.6: Set reset flip-flops in two different flavours: on the left the NAND and on the right the NOR. They differs from each other because where the NAND is active the NOR is inactive.

the inputs, are correlated: in fact we call them Q and $\bar{Q}$, because when $Q = 0$ $\bar{Q} = 1$ and viceversa. We so have three possible configurations:

- **Both inputs are inactive (S=R=1)**: the outputs can be in one of the two consistent states $(0, 1)$ or $(1, 0)$;
- **S is made active (S=0,R=1)**: Q becomes 1 after a time t and $\bar{Q}$ becomes 0 after a time $2t$. These states will stay the same even if S becomes inactive;
- **R is made active for a time longer than $2t$(R=0)**: Q will be set to 0.

For the NOR FF it is the same, but it is active high. A more compact version of their representation is visible in Figure 2.7 In Table 2.4 we can see the information

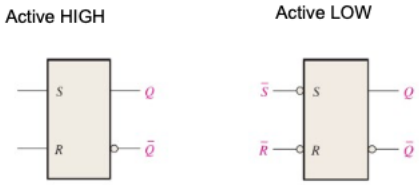


Figure 2.7: Compact circuit representation of the SR FF. On the left the NAND FF and on the right the NOR FF.

explained above in a more compact way. It is useful to understand the behaviour of

Input S	Input R	Output Q
0	0	Q_{old}
0	1	0
1	0	1
1	1	0

Table 2.4: Table of truth of the NAND SR FF

the system to watch its time evolution, shown in Figure 2.8. When a line is high it means that the value is 1, and when is low the value is 0. We can see how the output value is put to 1 if the set value is active (low). When the reset is active instead the output value becomes 0. Note that if the output is in 1 any consequential application of the set doesn't change the output, and analogously for the reset.
Some problems can arise from the asynchronous nature of the SR FF. A problem can arise if R and S become inactive at times that are separated by less than a one gate delay. In this configuration the FF is reduced to a pair of cross-coupled inverters. There is so a third non digital equilibrium state where both output voltages values are between 0 and 1!! (If the inverters are identical $V_1 = V_2 = V_M$. This state is

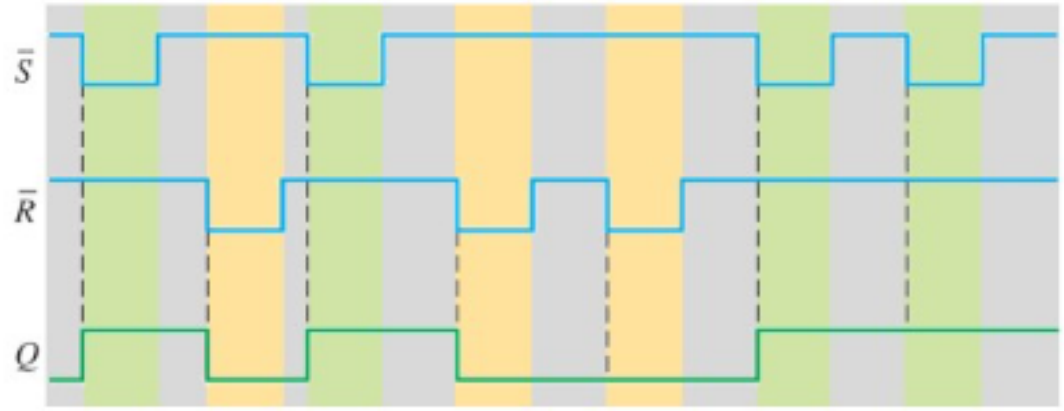


Figure 2.8: Time evolution of the NAND SR FF.

metastable: any fluctuations lends the FF to one of the two stables state[2]. We can see the inverter configuration and the metastable state in the middle row of Figure 2.9.
Let us imagine to slowly shorten the interval between the times when R and S become inactive:

1. If we now make R precedes S by less than one gate delay, nothing critical happens for some time because even though Q drops slightly, it remains above HIGH. Beyond a certain point, however, Q will drop below HIGH for longer times although it will eventually return to a value above HIGH
2. When the interval is reduced to zero, Q and Q are left in the metastable state.
3. If we now make S precede R by an interval shorter than one gate delay, Q will eventually drop below LOW, but it will take longer and longer to do so as the interval is shortened

The three configuration are enlisted, with the same numeration, in the first row of Figure 2.9. There exists so a metastability window τ_w such that if S and R are separated by less than τ_w the FF will become metastable. If we assume that R goes from 0 to 1 and back with an average frequency f the probability of becoming metastable becomes $f\tau_w$.We can also write the probability of being metastable after a time $T + \tau_w$ having been metastable for a time T:

$$P(T + \tau_w) = f\tau_w P(T)$$

and it is called **probability of synchronization failure** or **arbitration conflict**. $P(T)$ cannot be reduced to zero in an RS flip-flop, but it can be made arbitrarily small by waiting long enough before examining Q because in practice decreases exponentially for large T.

Gated Latch

A gated latch, represented in Figure 2.10 is a variation of the previous FF, modified by adding logic ports. The gated latch has an additional input, called enable (EN) that must be HIGH in order for the latch to respond to the S and R inputs (active LOW). You are reducing the time interval in which you are subjected to the metastable state. It is a latch because it is still sensitive to levels. We stress that:

[2]If it is not too clear to you think about a potential with two holes. If you put a particle on the maximum it stays there, since it is a equilibrium (first derivative=0) but with even a small perturbation it will fall in one of the two holes.

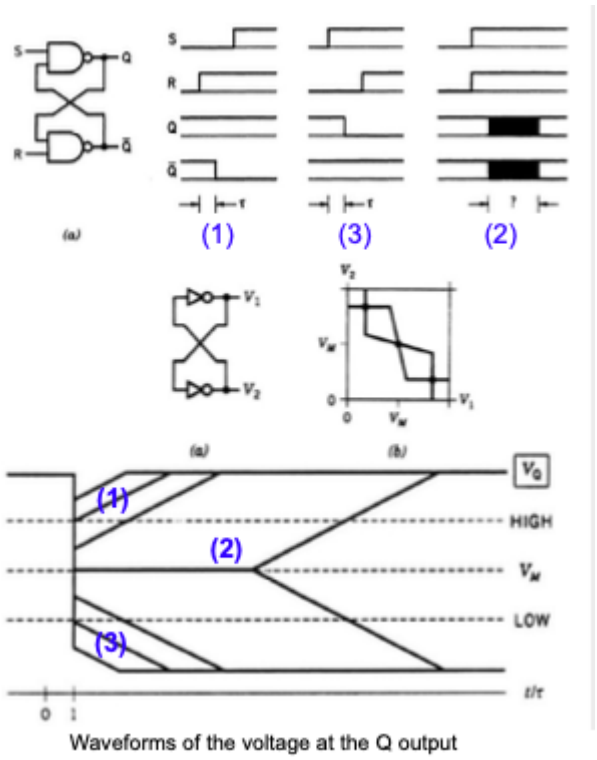


Figure 2.9: On the first row we have the possible states in the various configuration, on the second row the metastability and on the third a plot of how the metastable state goes to the stable ones in respect to t.

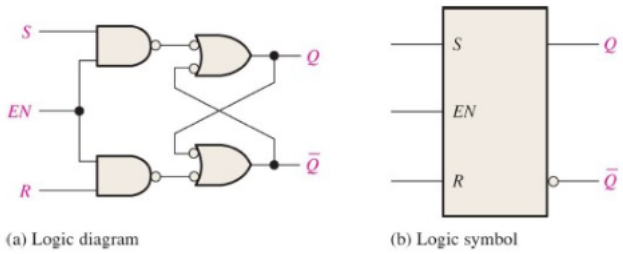


Figure 2.10: Gated latch representation

$$EN = 0 \Rightarrow \text{inputs} = (1, 1)$$
$$EN = 1 \Rightarrow \text{inputs} = (S, R)$$

D-Latch

What we will use on FPGA. It is a gated latch in which you anticorrelate the two input of the circuit with an inverter, as explained in Figure 2.11. There is so only one real input, called **Data input**. An important feature is that the inverter will introduce a delay that makes impossible the condition of simultaneous transitions. If you open the enable you are sensitive to the variation of the level, and so the output follows the input D.

2.5.3 Synchronuos Flip-flop

In a digital synchronous system all waveforms are synchronized by a clock. The clock itself doesn't carry informations, and it is a periodic waveform in which the period

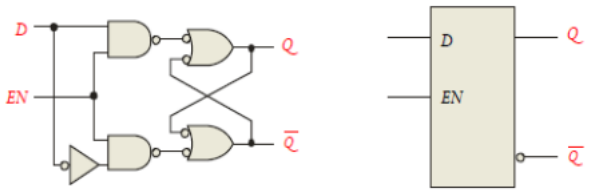


Figure 2.11: Gated D-Latch. On the left the logic configuration, with the inverter, can be seen. On the right there is the circuital representation.

(interval between pulses) equals the time for one bit. We choose the **rising edge** of the clock as activating, and it is really important to state if you are using the rising or the falling edge.
For example if we think of a clocked SR FF we understand that the output changes only on the rising edge of the clock.
In clocked FF we demand that inputs other than the clock are kept steady for a setup time t_s before a clock transition that might cause a metastable condition and also after a hold time t_H after the clock transition.

Edge-triggered FF

Edge-triggered flip-flops change state only at one of the clock signal transitions, and usually we choose the rising edge. In these devices the state of the output after the clock depends on the state of the output and the inputs just prior to the clock. There are some time costraints:

- The clock must remain active for a time t_w, the minimum clock-pulse width defined by the edge detection circuitry.
- Changes of state occur a propagation time t_p after the clock.

We need $t_w < t_p$, otherwise it would be impossible to build systems in which the response of a given FF depends on the state of the FF itself or other FF at the time of the clock transition, because the change will become "istantaneous" to our system.
An edge-triggered flip-flop is described by an equation (or a truth table) that gives the output during clock period n + 1 as a function of the values of the inputs and the output during clock period n:

$$Q_{n+1} = F(\text{inputs}_n)$$

4-bit shift register

It is an application of D flip-flop as we can see by Figure 2.12. It is a memory: a set of bit cells which can be loaded. Through the array of gates some input can be loaded in the four FF. The only control input is S/L (SHIFT/LOAD), and determines whether the D input of a given FF is the Q output of the FF of the next-highest order or whether it is the corresponding bit of the 4-bit word. So you can load the four bits and then you can read out from a single line the sequence of the four bits. With the enable clock you see the clock pulse and the bits are propagated: you shift the four bits such that the bits can be read out. Essentially you can, by pulsing with a clock the input, extract a stream of bits. With each pulse of the clock you get a bit. It is a very good way to read also a very long stream of bits.It is used in many cases also for programming FPGA itself.
On a more quantitative way:

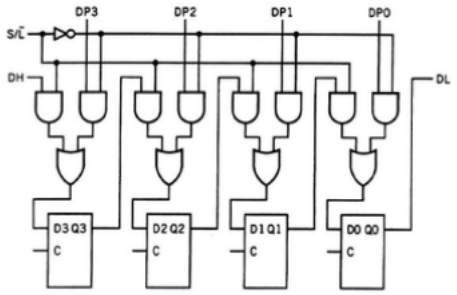


Figure 2.12: Representation of the 4-bit shift register. On the low we can see the four D-FF.

- **S/L=1**: the 4-bit word in the shift register is shifted one position to the right at the clock transition; DH is copied into Q3, and DL is our output bit.Q3 is copied into itself if it is tied to DH; we can thus obtain an aritmetic shift-right or divide-by-2 circuit.
- **S/L=0**: new parallel data are copied into the shift register (register load).

JK flipflop

Less common that the data FF but really useful, since they have two additional inputs J and K such that:

$$Q_{n+1} = J_n\bar{Q}_n + \bar{K}_nQ_n.$$

It is obteined by adding some logic gates to a data FF as shown in Figure 2.13. So a

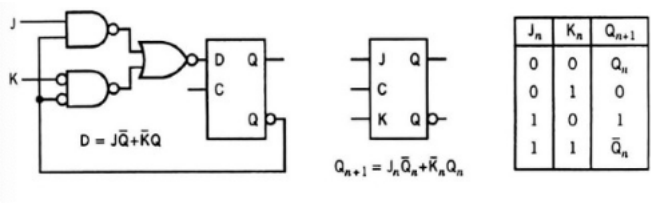


Figure 2.13: On the left there is the logic representation of a JK FF, in the center the circuital representation and on the right its truth table.

JK FF change state if J and K are both equal to 1 and remains the same if they are both equal to 0, as we can see from the truth table in Figure 2.13.
We stress that since it is a clocked FF its states changes only when there is a rising edge of the clock.

4-bit counter

We can now see an implementation of the JK FF: a 4-bit counter, represented in Figure 2.14. We have the following possibilities:

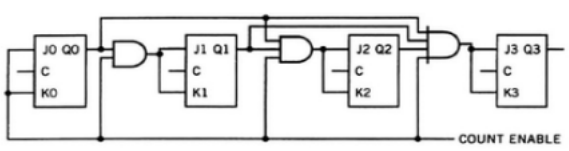


Figure 2.14: Four bit counter representation

- **COUNT ENABLE=0**: J_i and K_i inputs are all 0 and the counter stays wherever it is;
- **COUNT ENABLE=1**: the lowest order bit Q0 toggles at every clock, and in each succeeding position the AND gates allow the bit to toggle only if all preceding bits are equal to one, since the output of the AND of the i-th FF is the input of the (i+1)-th FF. By following the output rule if both input are one the output changes, and if both are 0 it stays the same.We remember that the count word is (Q3,Q2,Q1,Q0).

In this way the counter cycles through the sequence 0,1,...,15,0 and so on. We
With a more complicated gating scheme the counter can also count down and can be set to an arbitrary value.
We stress that loading a parallel word and then counting down to zero is useful when generating pulses of variable length because only one gate is needed to detect when the counter arrives at zero... in contrast, counting up to a given value requires a complete magnitude comparator.[3]

[3]Even if there are more slides in Lesson 5 the professor stopped here.

Conclusions

Bibliography