Imperial College London

E 2.1 Digital Electronics 2

Lecture 1
Introduction to Digital Circuits

Peter Cheung
Department of Electrical & Electronic Engineering
Imperial College London

Course webpage: www.ee.ic.ac.uk/pcheung/teaching/ee2_digital/
E-mail: p.cheung@imperial.ac.uk

Aims and Objectives

- The aims of the course are:
  - To learn about digital circuits and how they can be used with other off-the-shelf components
  - To learn how to implement reasonably complicated circuits on Field Programmable Gate Arrays (FPGAs) using commercially available CAD systems
  - To learn how to design with Hardware Description Language instead of schematics
  - To provide an understanding of how digital systems communicate with each other and with their external environment
- By the end of the course, you should be able to:
  - analyse the operation of synchronous digital systems
  - design a synchronous digital system to meet a specification
  - determine the worst-case propagation delay in a digital circuits
  - understand how A/D and D/A conversion circuits work
  - design finite state machines for control tasks
  - design some digital circuits on real hardware using FPGAs
  - design digital circuits through the use of Verilog HDL, and to write good Verilog codes

Welcome to my second year course on Digital Electronics. You will find that the slides are supported by notes embedded with the Powerpoint presentations. All my teaching materials are also available on the course webpage: www.ee.ic.ac.uk/pcheung/teaching/ee2_digital/. The QR code here provides a shortcut to the course webpage. I will be updating the notes, the laboratory instructions and tutorial problem sheets each week after the lectures. All my lectures will be recorded with Panopto. The recordings will also be upload as soon as possible after the lectures.

The course consists of about 16 hours lectures interleaved with 6 problem solving classes. These will be held on Monday 3pm to 4pm and Tuesday 4pm to 6pm starting from 8th of October 2019.

This course follows on from the first year Digital Electronics I course. Unlike the first year course where all gates and flip-flops are assumed to exhibit ideal behaviour, this course will teach you about real-life digital circuits.

Digital circuits are ubiquitous. For example, there are more electronic modules in petrol or diesel cars these days than mechanical systems, let alone electric cars! A mobile phone has many times more transistors than human alive on earth, and most of these transistor are digital, i.e. working as on-off switches. Therefore this second year digital electronics course is fundamental to any EEE or EIE education.

It is important for you to know at this stage what you are expected to learn (i.e. the learning outcomes) from this module. **Learning outcomes** specify WHAT you should be able to do as a result of taking this module. Let me go through the listed outcomes in some details:

1. Understand synchronous digital systems – if you are given a circuit with gates and flip-flops, you should be able to predict how it behaves. For example, you should be able to draw the timing diagrams for output signals given the input stimuli, or write down the **sequence of states** that the circuit must go through.

2. Design circuits to meet specification In real circuits, outputs response to changes in inputs after some delay. In order for a digital circuit to work as intended, such delay must be taken into account, and you as a design engineer must be sure that there are no **timing violations** (i.e. circuit delays causing the circuit to fail).

3. A/D and D/A conversions – the physical world is generally analogue in nature and is not just '1's and '0's. However, electronic systems are mostly digital. Analogue to Digital (ADC) and Digital to Analogue (DAC) conversion provides the link between the analogue physical world to the digital electronics world. You need to understand HOW analogue signals are converted to digital, and how to interpret the datasheet of such components.

4. Finite State Machines – Designing digital circuits involve understanding of various fields, and one of the field of study is known as Finite State Machine (FSM). This is a systematic ways of thinking how a digital system goes through different states, and as a result, control the operation of a digital sub-system.

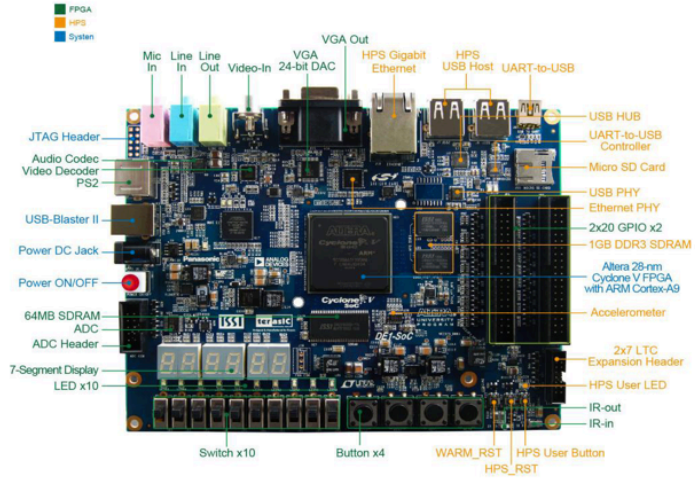
5. Field Programmable Gate Arrays – FPGAs is one of the primary technology for implementing digital circuits nowadays. This has replaced most of the implementations in "discrete logic" (such as 16-pin packaged TTL or CMOS gates). It has also replaced many Application Specific Integrated Circuits (ASICs) that the industry used to design. FPGAs prove to be much lower-risk and must easier to design as compared to other approaches. Therefore this course will be based around the use of FPGAs. (to continue .....)

About the course

- Assessment – 100% examination in June and one Lab Experiment
- Handouts in the form of PowerPoint slides with supplementary notes
- Text Books (Not essential)
  - Brown and Vranesic, "Fundamental of Digital Logic with Verilog Design" (2nd hand availabl, otherwise extremely expensive)
  - Dally and Harting, "Digital Design: a systems approach), around £40 (2nd hand ~£30)
  - Peter J Ashenden, "Digital Design (Verilog)", around £30 if you look hard!
- Exercises & Tutorial Problems:
  - Lectures are supported by tutorial problem sheets
  - A purpose-designed 2nd year Lab experiment (VERI) to support this course starting from Week 7 in the Autumn Term (starting from 11 Nov 2019)
  - You may install your own Quartus Prime design software on your own laptop
  - You may borrow the experiment board (DE1-SoC) from the stores to use at home. The loan period is one week, renewable if no one is waiting for it.

PYKC 8 Oct 2019 E2.1 Digital Electronics Lecture 1 Slide 3

DE1-SOC Board



PYKC 8 Oct 2019 E2.1 Digital Electronics Lecture 1 Slide 4

6. **Verilog HDL** – While you mostly use schematic diagrams to describe your digital designs in your first year, you will ABANDON this in favour of a computer language to specify and design your digital circuits. You may find this odd initially because diagrams are generally more intuitive than a computer language. However, using a **Hardware Description Language** (HDL as it is called), and in our case, using Verilog, is the way that most modern digital systems are specified and designed. No matter whether you like programming or not, as a electronic engineer, you will have no choice but to learn such a language.

This course will be assessed through an examination paper in June 2020. There will also be an associated E2 Laboratory Experiment – VERI.

The Laboratory Experiment is EXTREMELY IMPORTANT in helping you to learn this subject. It is intended to teach you how to design digital circuits using Verilog HDL targeting implementations on FPGAs. The Lab sessions will run for FOUR weeks starting on the 11th of November, and assessment for this experiment will take place in the last week of term (starting 9th of December). You may also borrow the experiment board (DE1-SOC) to use at home and at your leisure, one week at a time. There will be around one such board for every four students to share.

There are three recommended textbook "Fundamental of Digital Logic with Verilog Design 3/e" by Stephen Brown and Zvonko Vranesic. Unfortunately this book is in short supply and is extremely expensive to buy new. You may be able to pick up a second hand copy on the internet. Another possible book but less relevant is by Dally and Harting. While it is NOT necessary to own a textbook for my course because I do not follow a particular textbook in my lectures or in the lab, I would recommend you to get hold of a second hand copy or an eBook in digital as a reference. The third book, also not compulsory, is "Digital Design (Verilog)" and is very Verilog specific. It is a good book if you want to learn Verilog well. Again it is not compulsory.

The practical aspects of this course module is based around a FPGA board, the DE1-SoC. Here is an overall block diagram of the board. Don't worry about the details for now. I will be explaining to you the various bits on the board later when you are about to start the VERI experiment in the Lab in mid-November.

There should one DE1 board for every four students to borrow and use at home. The basic lending duration is one week. You can renew your borrowing period beyond one week if there are free ones in the Stores.

To borrow a board, bring your ID Card to Level 1 store, and you can check out a board to take away. But you must return it at the end of the loan period.
This board has everything you need to do the experiment and MORE. It consists of a Cyclone V FPGA (which I will explain in more details in a later lecture). It has various input and output devices. This is a very powerful board and it contains lots of additional hardware resources that go beyond the scope of this module and the experiment. You are encouraged to explore it, particularly if you are an EIE student.

Lecture Plan

- Introduction & Verilog
  1. Introduction to Digital Logic
  2. Introduction to FPGAs
  3. Verilog HDL – Part 1
  4. Verilog HDL – Part 2
- Control Logic Design and timing
  5. Counters/Timers & Shift Registers
  6. Digital signals and timing constraints
  7. Finite state machine – Part 1
  8. Finite state machine – Part 2
- Interfacing to analogue circuits
  9. D-to-A converters
  10. A-to-D converters
- Digital systems and interfaces
  11. Digital Systems and DE1-SoC board
  12. Serial-Parallel Interface
- Memory Circuits
  13. Memory interface and embedded memory
  14. External memory, SRAMs and SDRAMs
- Computation circuits
  15. Arithmetic circuits in FPGAs
  16. Processors on FPGAs

This is the current lecture plan for the course. Details may change as we progress through the term. There will be around 16 lectures (slightly higher than the nominal 15 lecture per module in the second year).

I will cover a number of topics that form the basic course in digital electronics. By the end of the course, you should be able to independently design digital circuits using FPGAs. There will also be a couple of lectures on how to interface digital systems with the analogue system via D-to-A and A-to-D converters.

I will be linking my lectures to the Lab Experiment wherever possible. To do well on this course, you really need to take the Lab seriously!

IEC and other notations

(see tutorial on course webpage)

- Logic Levels
  - A logic 1 (or high) is usually represented by the positive supply voltage (e.g. 3.3v)
  - A digital chip can have multiple positive supply voltages
    - For example, FPGA internal circuits may use 1.8v, but interface circuits use 3.3v
  - A logic 0 (or low) is usually GND or 0v
- Gates
  - The label indicates how many of the inputs must be high to make the output high:

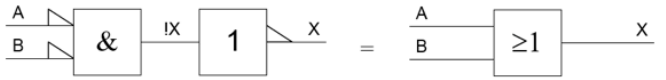
    & AND gate: all inputs high

    ≥1 OR gate: one or more inputs high

    =1 Exclusive-OR: exactly one input high

    2n Even Parity: even number of inputs high
- Inversion triangles (or bubbles)
  - We can invert signals on the way in or on the way out:

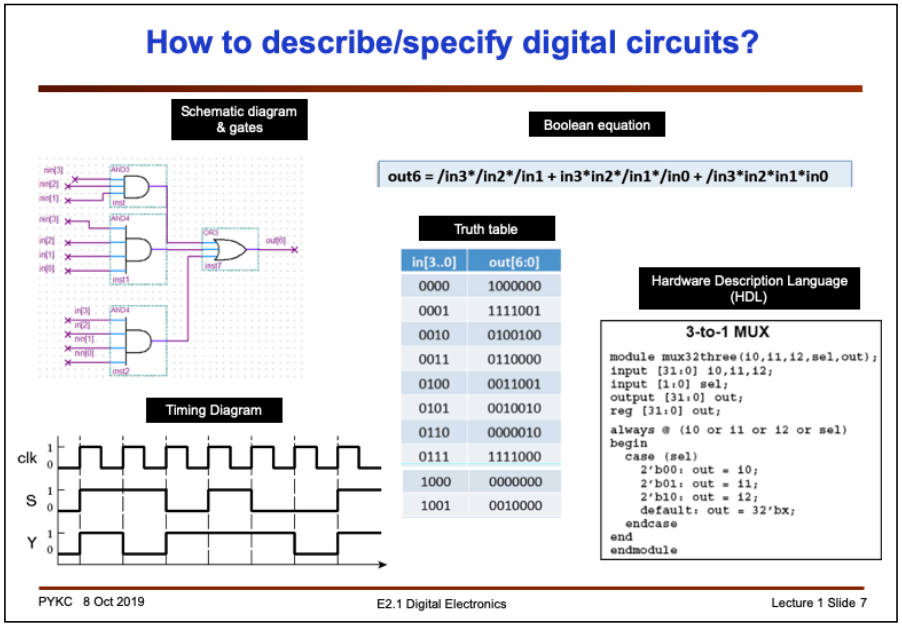
$\overline{X}$ or !X denotes the inverse of X.



This lecture is just partly a revision lecture, and I also want to introduce you to alternative notations used both in the notes and in some textbooks. This follows the IEEE standard for digital schematics.

Instead of using curves for gates, one could use rectangular blocks and a symbol to denote the logic function. Inversion could be on the input or output terminal. Instead of a circle, we could use a small triangle as shown here.

IEEE publishes the standards, and there is an excellent tutorial on this digital circuit notations published by Texas Instrument (see the course webpage). You don't really need to spend much effort on this – just need to learn the basics so that you can understand the meaning of the symbols and the labels used for signals.



In the first year, you learned about the different ways of describing or specifying a digital circuit.

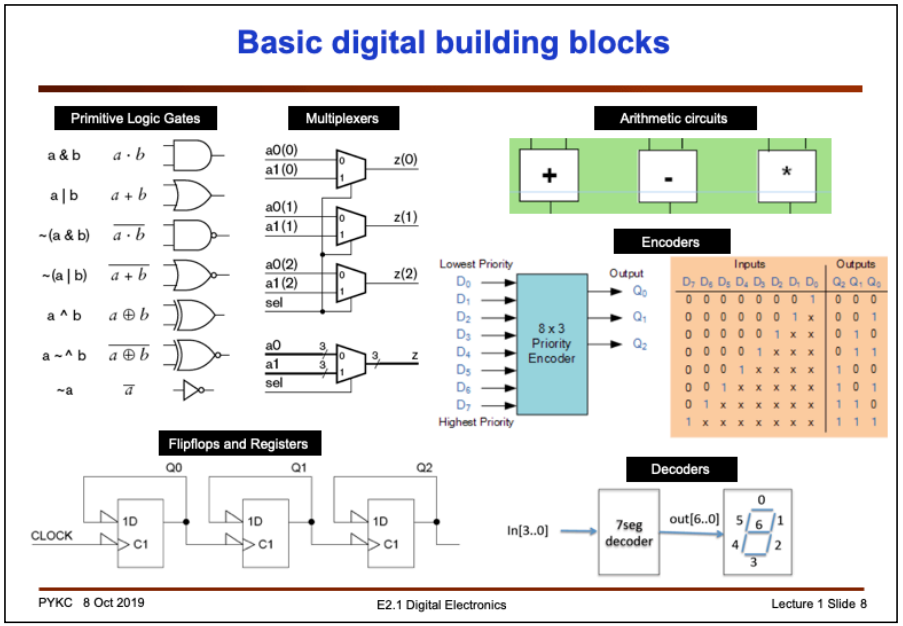
1. Schematic diagrams with gates – this method is the first thing you learned and it is easy to understand. However, as will be seen in Lecture 3, this is not necessarily the best way to specify a large digital system.

2. Boolean equations – this provides a formal way to express logical relationships between Boolean variables. Useful when designing on paper, but less useful in practice. In particular, we rarely use Boolean algebra to perform logic simplification in real-life!

3. Truth Tables – this is a universal way to describe the behaviour of a circuit and we continue to use this in datasheets or even in actual designs.

4. Timing diagrams – this is a useful way to explain behaviour of sequential circuits and is used in datasheets. However, not that useful as a method to specify a circuit in a CAD system.

5. Hardware Description Languages (HDLs) – this is a new method you learn this year (except EIE students who have already encountered this in their group project). This is what we will be using to specify and design digital hardware from now on. For this course, we will be using Verilog HDL, which is one that is very closed to the C language. It is also used extensively for designing integrated circuits such as ASICs and other type of chips. Another popular HDL is VHDL. I personally find VHDL too wordy (verbose). Finally, there are now emerging higher level languages such as OpenCL, which is attempting to make hardware design more like programming a computer. This topic is left to later years.



You have also learned about the various building blocks for digital electronics.

1. Primitive gates – We have the basic AND, OR, NAND, NOR, XOR and XNOR gates.

2. Multiplexers MUXs – These are really useful component. Shown here is a 2-to-1 MUX with two data inputs and one select input. The output is one or the other depending on the select input (sel). We often put a number of these together to provide multiplexing function to a mult-bit data word (as shown here with two 3-bit numbers).

3. Arithmetic circuits – Commonly found are adders and multipliers. Subtractor can be built from an adder if we use 2's complement representation of signed integers.

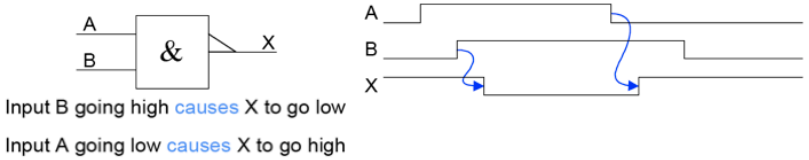
4. Encoders/Decoders – These two are related. **Encoding** is a logic module that reduces (encodes) a large number of bits and produces fewer output bits. **Decoders** are the opposite. Shown here is a 7-segment display decoder, where 4 input bits are decoded into 7 logic signals to drive the seven segments of the display. The **encoder** here is known as a **priority encoder**. It produces a 3-bit output showing where the first '1' is encounters from the most-significant bit D7 to the least significant bit D0.

5. Flipflops and Registers – These are the building blocks for all sequential circuits. As will be seen later, we really only use one type of flipflop – the D-FF.

These are all important components that all digital circuit designers need to be familiar with. However, nowadays, we rarely design large digital systems at such low levels. Instead we generally try to express these building blocks in a more abstract manner in a hardware description language (as we will see in later lectures).

In addition to these basic blocks, we also have memory devices and microprocessors. These are topics that we will cover towards the end of this module.

Cause & Effect



Input B going high causes X to go low

Input A going low causes X to go high

Propagation Delay:
The time delay between a cause (an input changing) and its effect (an output changing), assuming output load capacitance of 30pF.

Example: 74AC00: Advanced CMOS 2-input NAND gate

	min	typ	max	
A↑ to X↓ (t_{PHL})	1.5	4.5	6.5	ns
A↓ to X↑ (t_{PLH})	1.5	6.0	8.0	ns

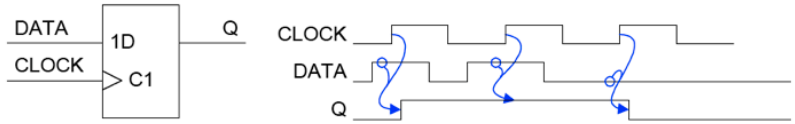
t_{PHL} and t_{PLH} refer to the direction that the output changes: high-to-low or low-to-high.

All digital circuits exhibit propagation delay. Here it shows the delay table for a "**discrete logic**" CMOS NAND gate. The delay could be in the region of nanoseconds. However, with the FPGA chips we use for this module, the internal "gate" propagation delay is approximately 100ps, which is much faster than discrete logic. As can be seen later, the "gate" inside the FPGA is also much more complex than a simple NAND gate.

Also note that propagation delay depends on the "**cause**" (input rising or falling, and on the slope of the edge) and the "**effect**" (output rising or falling). Delay also depends on what are connected to the output (i.e. the loading). As can be seen in the example here, the rising edge A to falling edge X delay is lower than that of A falling to X rising.

Note that I use an arrow to indicate the cause (the blunt end) and the effect (the pointed end) in a timing diagram.

D-Flipflop (1)



Notation:

>	input effect happens on the rising edge
C1	C ⇒ Clock input, 1 ⇒ This input *is* input number 1.
1D	D ⇒ Data input, 1 ⇒ This input *is controlled by* input number 1.

The meaning of a number depends on its position:

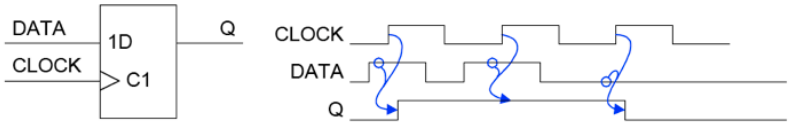
A number *after* a letter is used to identify a particular input.
A number *before* a letter means that this input is controlled by one of the other inputs.

You learned about various types of flipflops (FFs) in the first year. In fact, all you need is the D-FF. With a D-FF, you can construct circuits to behave like various types of flipflops: Toggle (T-FF), set-reset (SR-FF) or a JK-FF.

Therefore in this course, we will ONLY use D-FF for everything. This is in fact what happens in practical designs.

We use the IEEE standards for the symbol here. C mean clock input, the number 1 is a numerical label (as clock 1). D is for data input, and 1D means this input is controlled by input 1. Q is the flipflop output.

D-Flipflop (2)



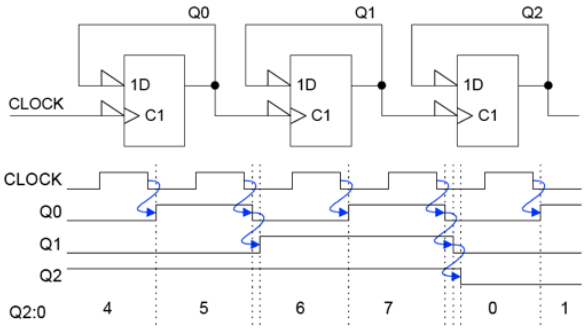
Cause and Effect:

- CLOCK↑ causes Q to change after a short delay. This is the *only* time Q ever changes.
- The value of D *just before* CLOCK↑ is the new Q.
- Propagation delay CLOCK↑ to Q is typically 1 ns.
- Propagation delay DATA to Q **does not make sense** since DATA changing does not cause Q to change.

Timing and delay parameters for flipflop is different from that with gates. Shown here is a D-FF that responses to a rising edge on the clock signal. A D-FF is like a camera, taking a "picture" from the scene (input is D). The clock input C1 is like the trigger on the camera – when pressed it samples the input and take a picture. The "cause" here is the rising edge of the CLOCK and the "effect" is the Q output sampling the D input, and keep the value until the next rising edge of the clock.

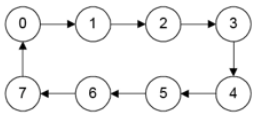
The delay here is from CLOCK rising edge to Q output changing. However, for the D-FF to work properly, there are two other timing parameters which are important: the **setup time** and the **hold time**. I will be talking about these in a later lecture.

Ripple Counter



- Notice inverters on the CLOCK and DATA inputs
- Least significant bit of a number is always labelled 0

State Diagram
(not including transient states):



Propagation Delay: CLOCK↓ to Q2 = 3 x 1ns = 3ns

Here is an example of a D-FF used in a ripple counter.

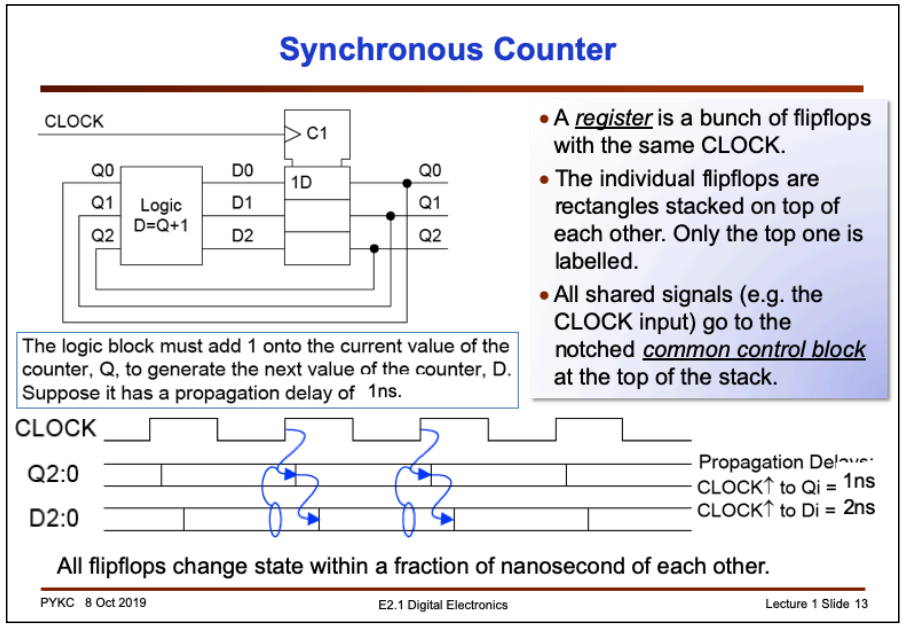
Q0 value is first inverted (represented by the triangle) and then used as D input on the next clock cycle. The flipflop is triggered on the FALLING edge of CLOCK. Therefore the Q output "TOGGLES" on each active edge of the clock (i.e. falling edge). Q0 is therefore changing at half the rate of CLOCK, hence this flipflop acts as a divide-by-2 circuit.

The Q0 signal is now used as clock input to the next D-FF. Hence Q1 is toggling at half the frequency of Q0. The circuit is effectively a binary counter.

This is a simple finite state machine (FSM) because it has 8 states which cycles through in a sequence. FSM will be covered in some later lectures in details and it is a very important topic in digital designs.

We then use the Q0 output as the clock input the next stage etc. Note that because the 2nd stage only starts to work once the first stage is completed, the propagation of effects "ripples" through the circuit – hence we call this a "ripple counter".

This counter is also known as an **asynchronous sequential circuit**. It is "**asynchronous**" because the output signals are NOT synchronised to a single clock signal (since there are many clock signals), and "**sequential**" because its current output value (or state) depends on previous output values in the sequence.



The ripple counter is potentially slow. The delay between the active edge of the clock and the counter output giving the correct value is dependent on the number of flipflops in the circuit and therefore the size of the counter (i.e. how many stages) .

A far better approach is to use the flipflops TOGETHER as a group, and clock them using THE SAME CLOCK signal as shown here. The Logic Block is a combinatorial circuit which computes the next D value D2:0 from the current Q value Q2:0. (D has three bits D0, D1 and D2. We use the notation D2:0 to represent this.) The relationship between D and Q is simple: D2:0 = Q2:0 + 1.

Since the three output bits Q2:0 change within a fraction of a nanosecond of each other, this circuit is: 1) faster than the ripple counter; 2) the “delay” is constant instead of dependent on the size of the counter.

This circuit is known as a **synchronous sequential circuit** because its function is synchronous to a single clock signal. If you regard the Q2:0 output value as a state value, it follows a finite number of states in a defined sequence. Therefore it is also a form of **Finite State Machine**.

Note the notation with the arrows.