Introduction to VHDL

COMP311
Tony McGregor
G.1.05
tonym@cs.waikato.ac.nz

What is VHDL?

- **V**HSIC **H**ardware **D**escription **L**anguage.
  (VHSIC = Very High Speed Integrated Circuit)
- Allows description of the structure and function of a digital hardware system.
- One of two widely used HDL’s.
  - Verilog is the other.
- Both are IEEE standards.
  - VHDL is IEEE Std. 1076 (2002 is the latest)
  - Verilog is IEEE Std. 1364 (2001 is the latest)

Why a Description Language?

A description language allows us to create a model of a system

VHDL models can be used to:

- Specify a system so that different people/groups can agree on what the system is.
- Use simulation to test and verify the design
- Formally verify correctness
- Automatic synthesis of circuits

VHDL advantages

- Allows specification of hardware using programming language-like structures.
- Self-documenting design.
- We can ‘synthesise’ a more detailed representation of a design from an abstract one.

Example VHDL

```
...

process (reset, clk)
begin
  if reset = ‘1’ then
    counter <= X”0000”;
  elsif rising_edge(clk) then
    if up = ‘1’ then
      counter <= counter + X”0001”;
    elsif down = ‘1’ then
      counter <= counter - X”0001”;
    end if;
  end if;
end process;

...
```

Recap – Digital Systems

- Represented by discrete voltage levels.
- We will deal mostly with binary digital systems.
  - Signals have two possible values
    - ‘1’ or ‘0’
    - TRUE or FALSE
    - ~5V or ~0V
- These systems are built on Boolean algebra.

Boolean Algebra

- Well defined operators for variables which only have two discrete values.

X	NOT X
0	1
1	0

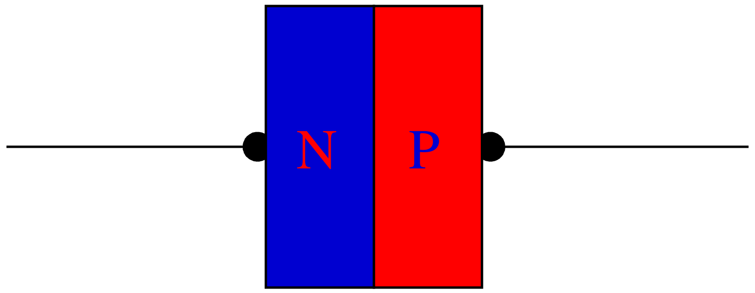
X	Y	X AND Y
0	0	0
0	1	0
1	0	0
1	1	1

X	Y	X OR Y
0	0	0
0	1	1
1	0	1
1	1	1

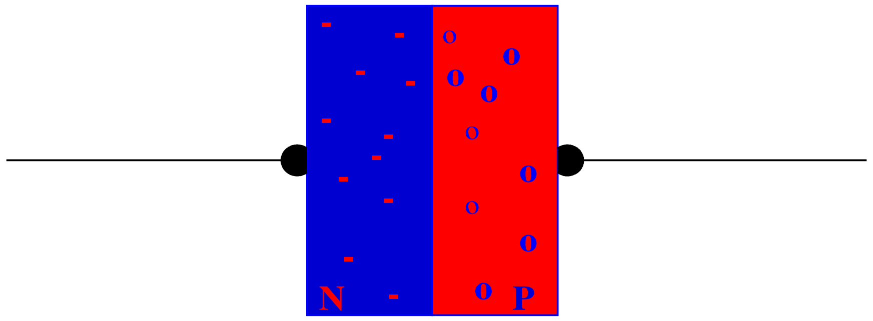
X	Y	X XOR Y
0	0	0
0	1	1
1	0	1
1	1	0

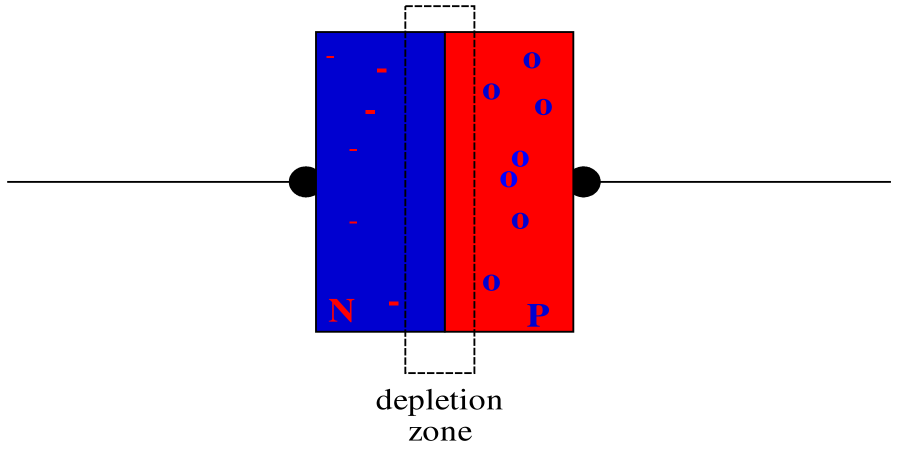
Transistors

- Build from silicon
  - a crystal => doesn't transmit electricity
- Can add 'impurities' that create some free electrons (n-type) or `holes' (p-type)
- The most basic semiconductor is a diode:

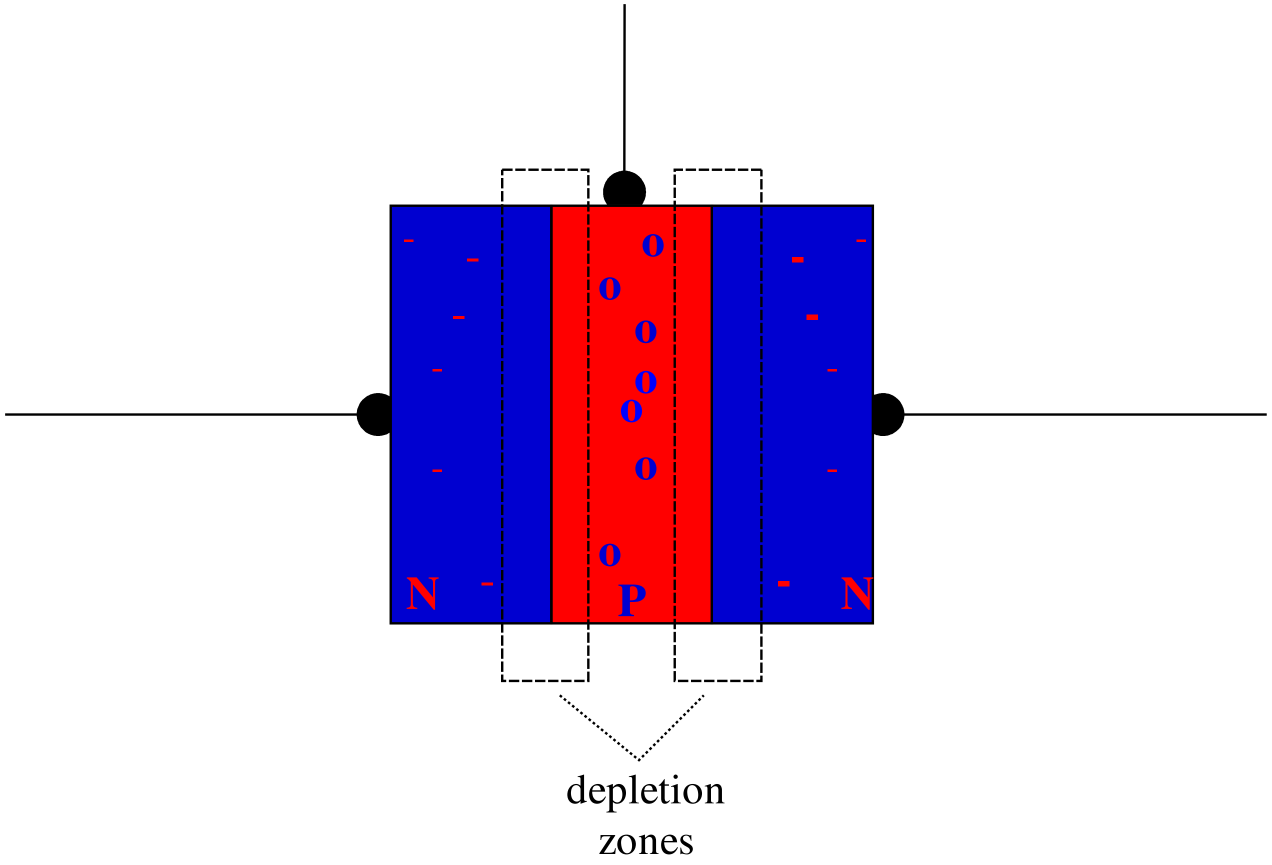


Diode

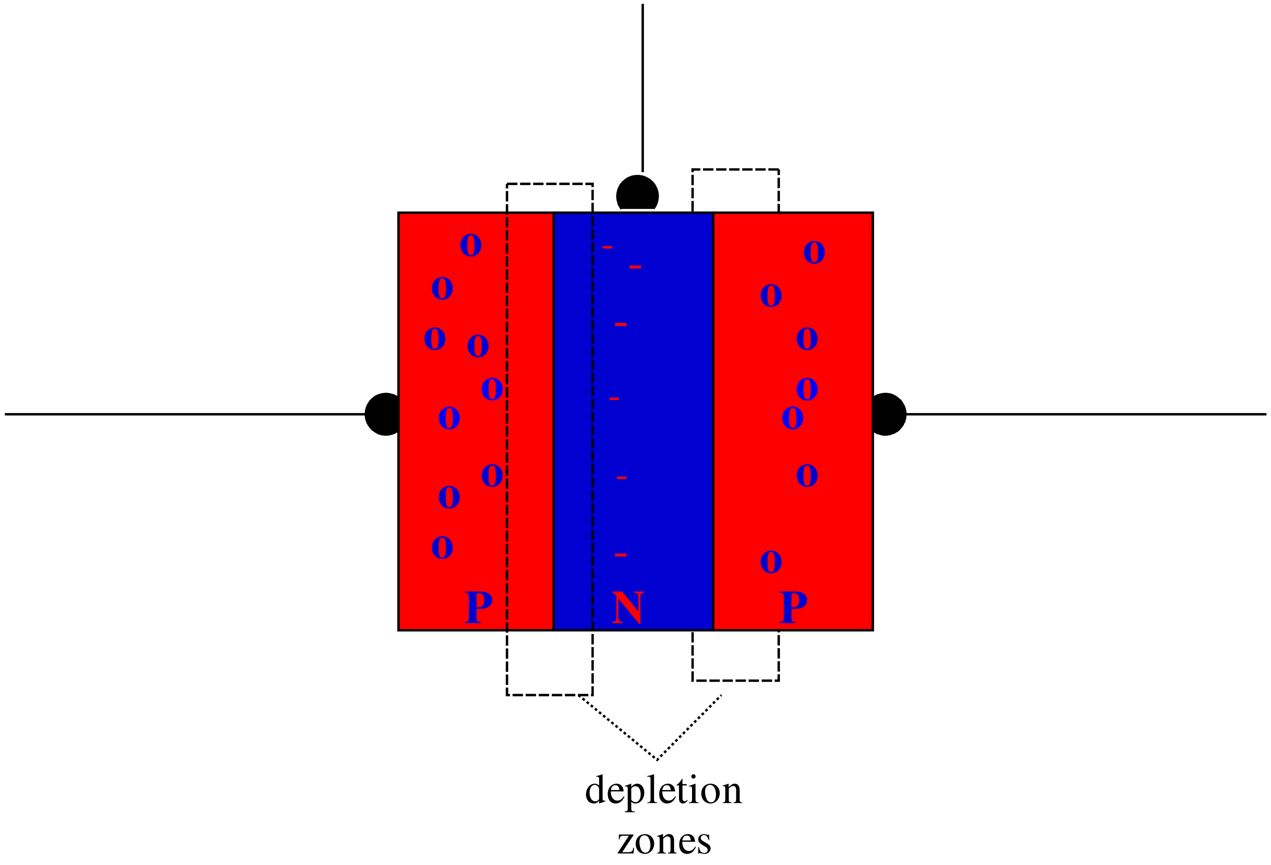




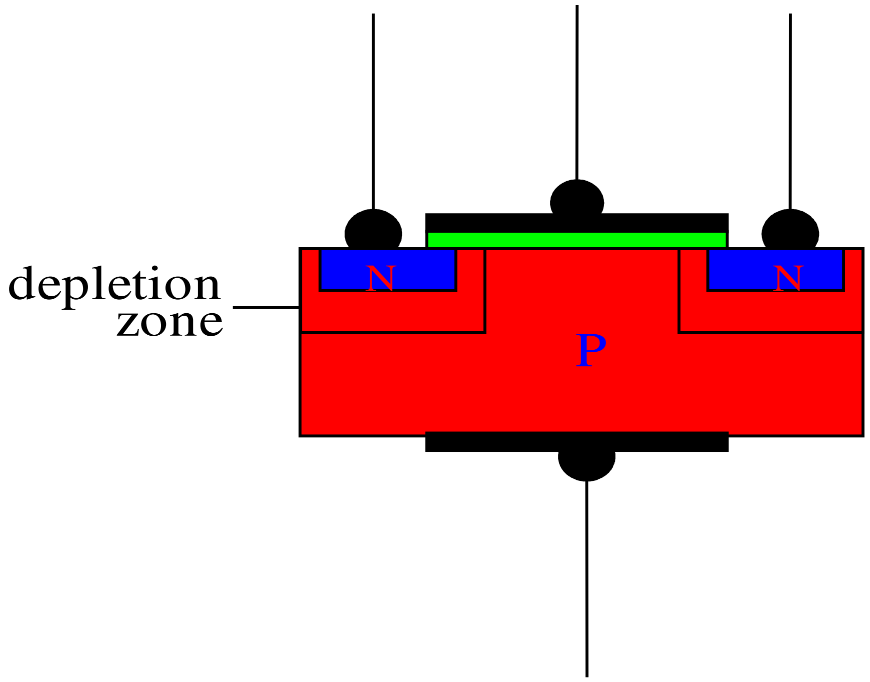
NPN Transistor



PNP Transistor

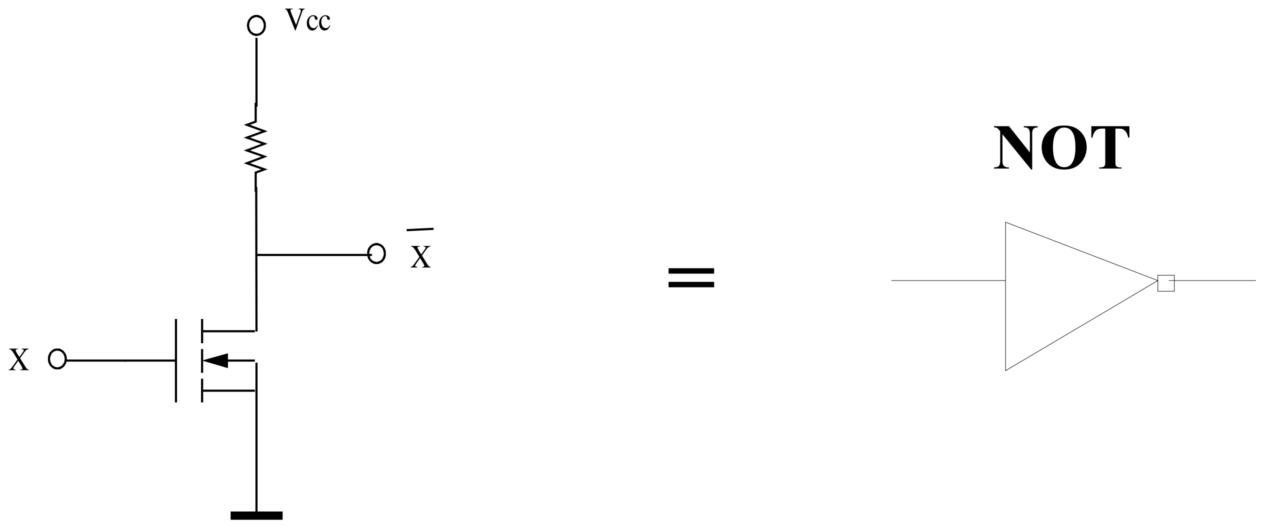


(Metal Oxide) Field Effect Transistor



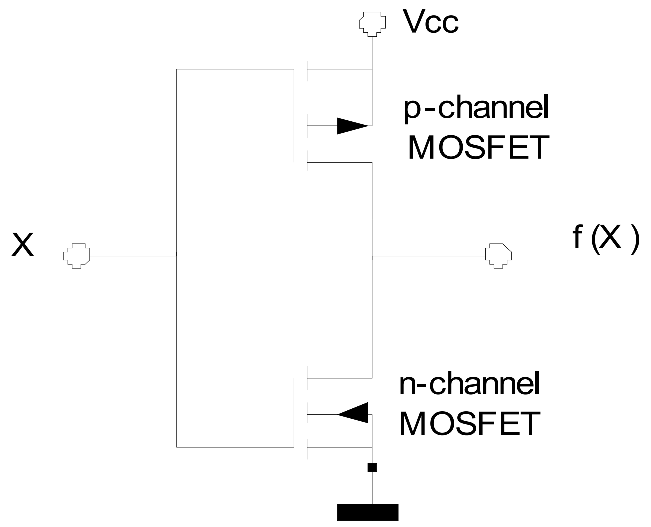
Gates

- Using transistors we can implement the Boolean operations in hardware.

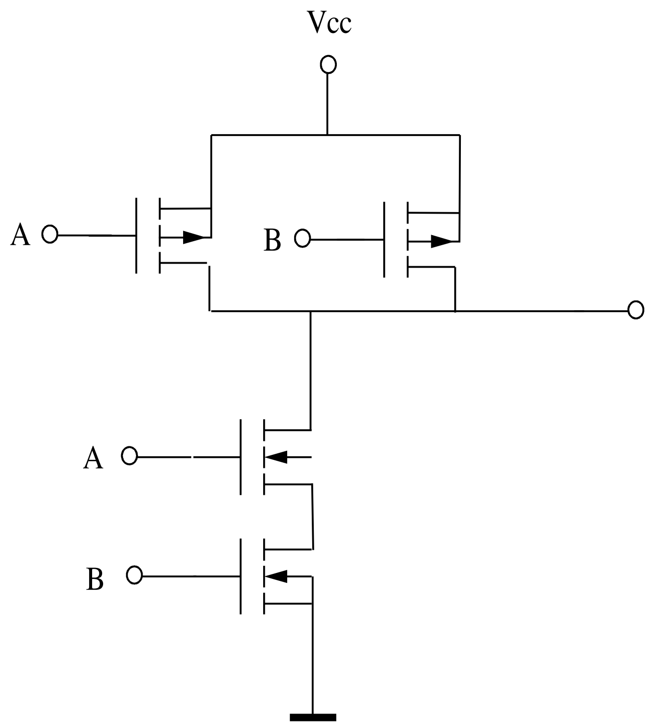


CMOS

- reduce power through the use of complementary MOS transistors

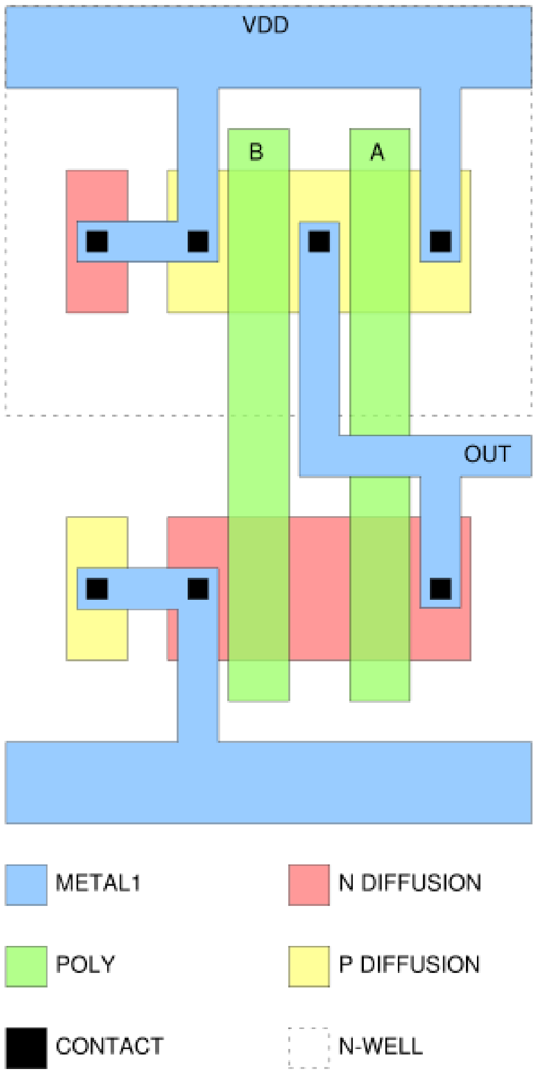


NAND Gate



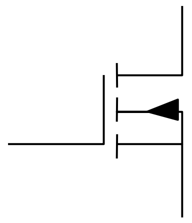
A	B	O
0	0	1
0	1	1
1	0	1
1	1	0

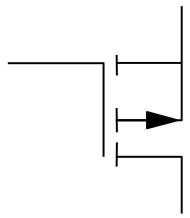
NAND VLSI Layout



2 Input OR Gate

?





input high = on

input low = off

input high = off

input low = on

for next year add nand gate diagram

Combinational Logic

- output is a function of the current input.
- system has no ‘memory’ of previous state.

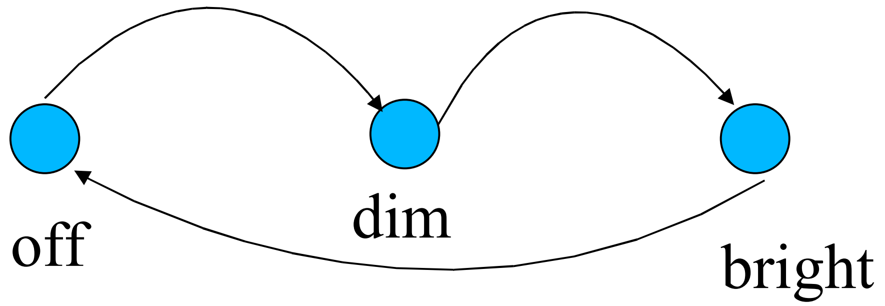
e.g. Consider a two switch hall light

SW1	SW2	Light
off	off	off
off	on	on
on	off	on
on	on	off

Sequential Logic

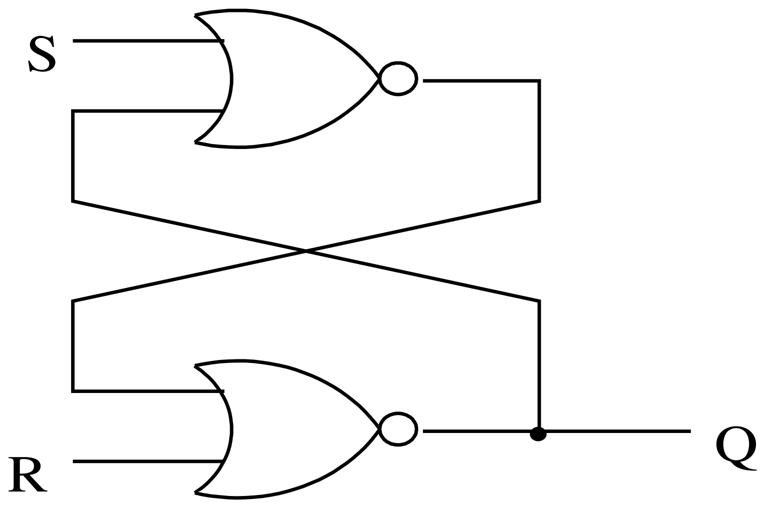
- output is a function of the input and the current state.
- circuit must have ‘memory’ to hold the state.

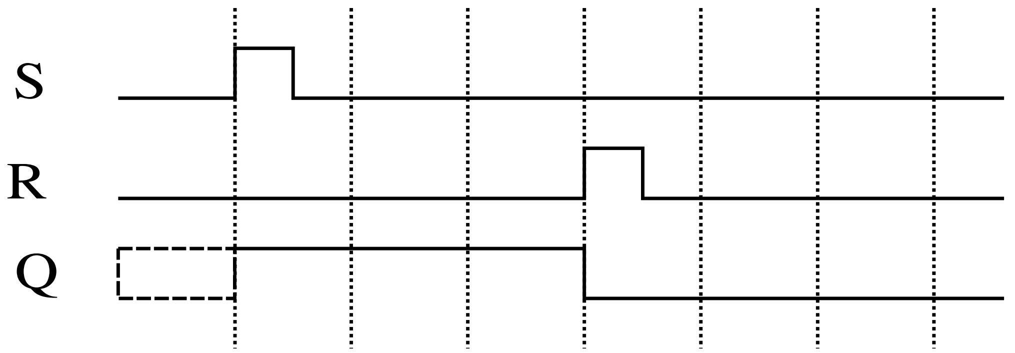
e.g. consider a lamp that cycles between off, dimly lit, and full brightness with the push of a button.



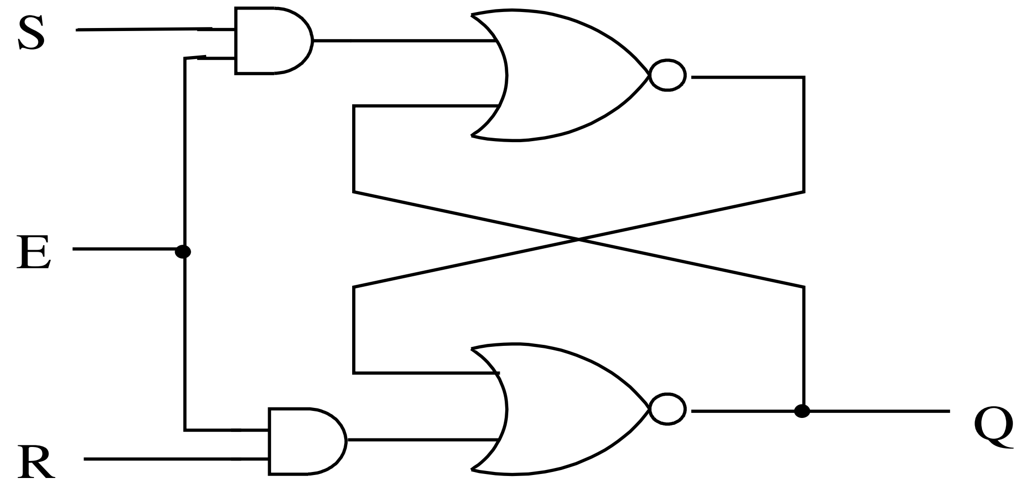
Basic Stroage Elements

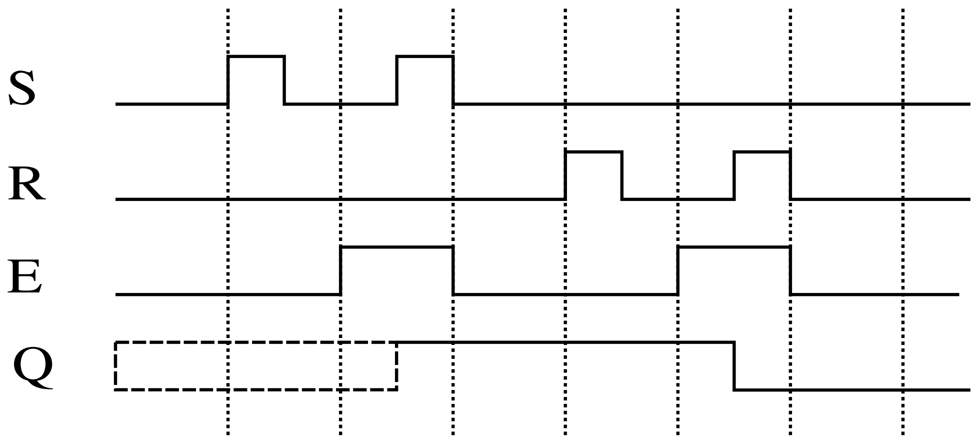
- RS latch



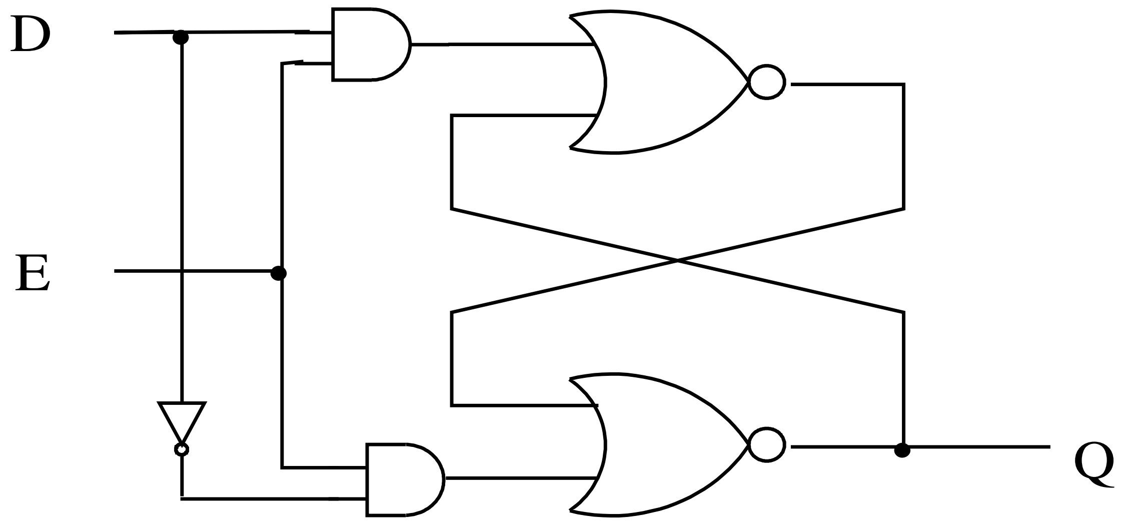


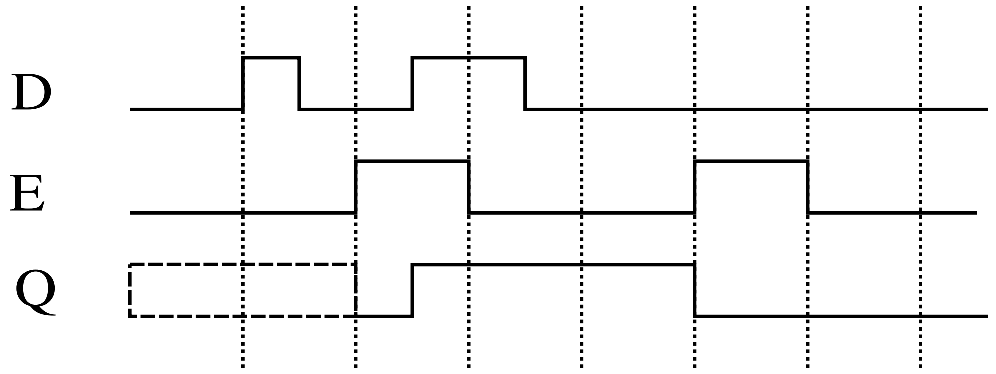
Enabled SR Flip flop



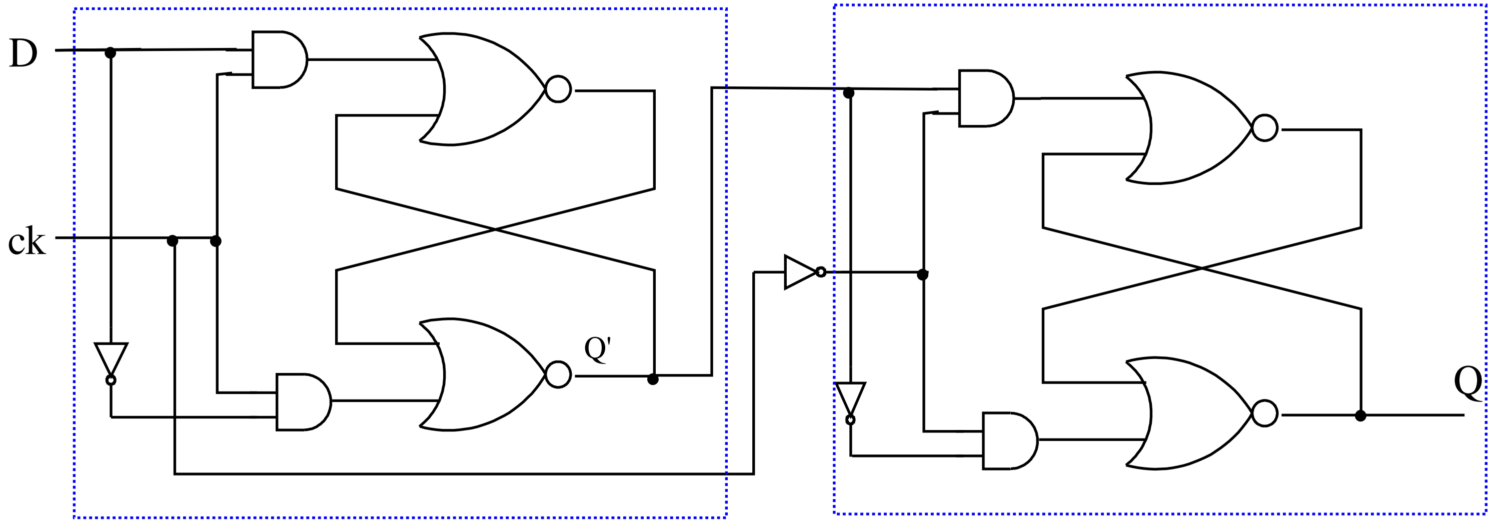


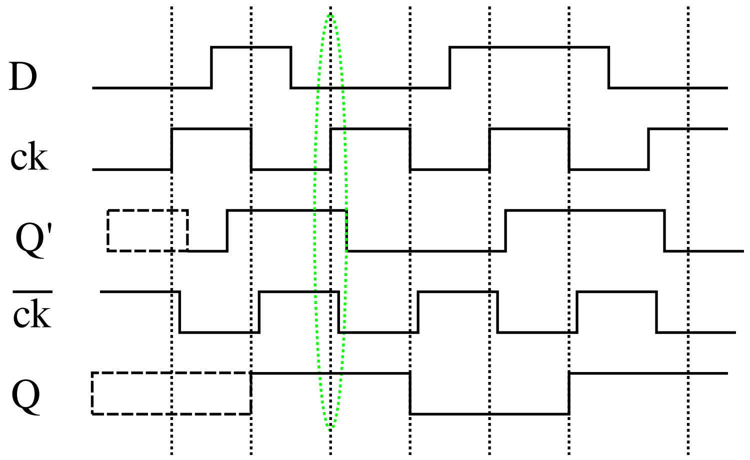
D Latch

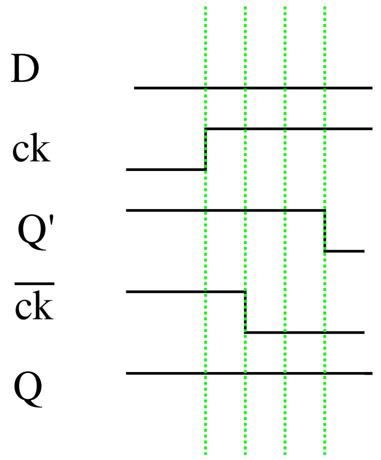




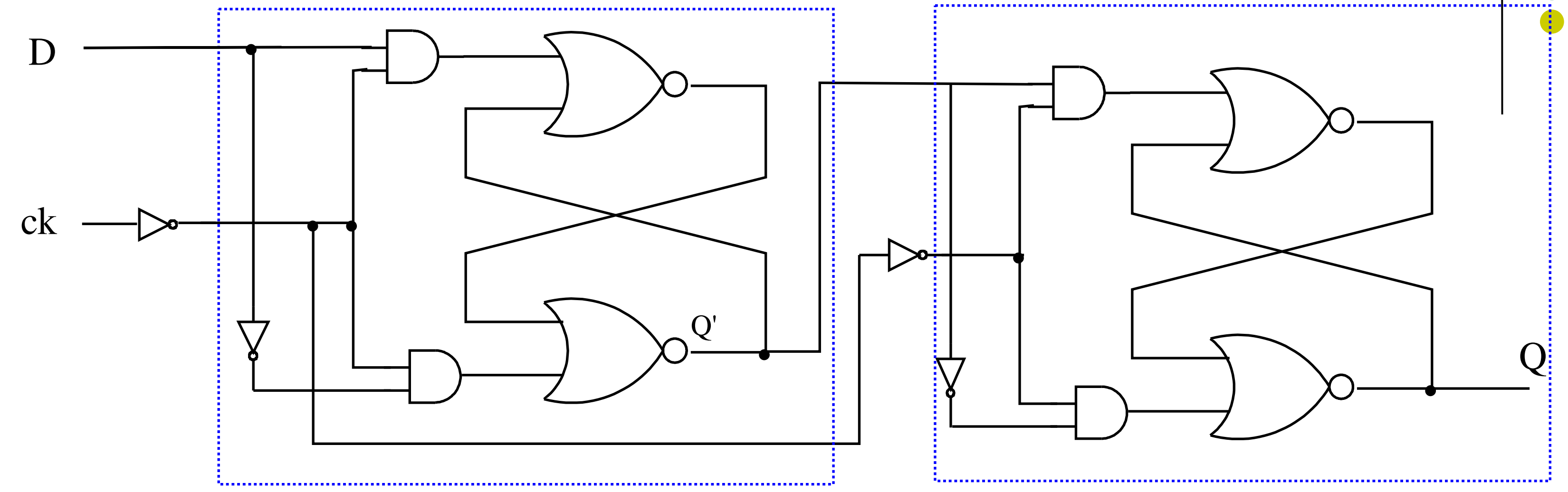
Falling Edge D Flip flop

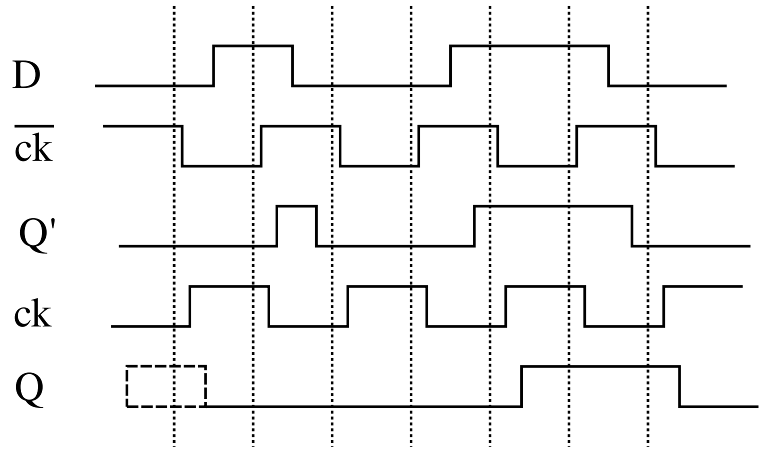


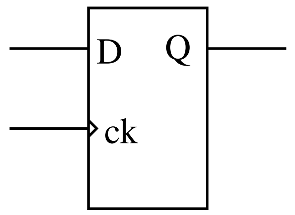




Rising Edge D Flip Flop

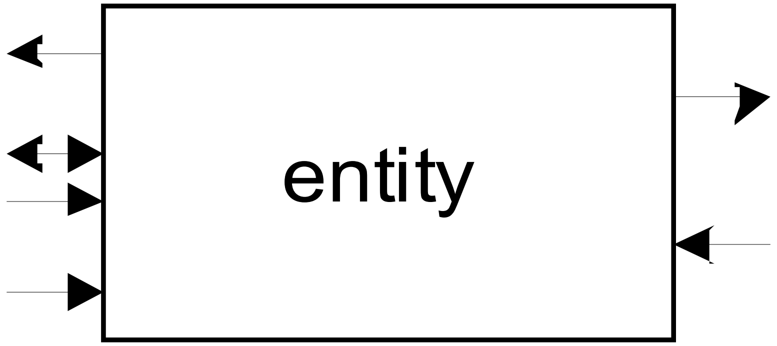






A VHDL Entity

- VHDL designs are divided into units known as ‘entities’.
- an entity consists of a series of signals which may be inputs, outputs or both.



Example entity declaration

```
entity and_gate is
  port (
     x : in std_logic;
     y : in std_logic;
     output : out std_logic
     );
end entity;
```

Architecture

```
entity and_gate is
  port (
    x : in std_logic;
    y : in std_logic;
    output : out std_logic
    );
end entity;

architecture behavioural of and_gate is
begin
  output <= x and y;
end behavioural;
```

Notes about VHDL

- As with all languages there are often many ways of saying (doing) the same thing.
- VHDL is **case-insensitive**.
- Synthesisable VHDL is a subset of VHDL.
  - Not all VHDL code or structures can be turned into hardware.

Entities

- The ‘entity’ is a part of the basic design unit in VHDL.
- The entity defines the inputs and outputs of the module, along with ‘generic’ parameters for the implementation.
- We will leave consideration of generics until later.

Entity Declaration Format

```
entity entity_name is
   generic (generic list);
   port (port definition list);
end entity_name;
```

Port declaration list

- This specifies the input and output ports for the entity.
- It is a **semicolon-separated** list of:

```
port_name : mode type
```

- ***`mode`*** is one of:
  - **`in`**
  - **`out`**
  - **`inout`**

- ***`type`*** specifies the datatype.

Example Entity Declaration

```
entity and_gate is
  port (
    x : in std_logic;
    y : in std_logic;
    output : out std_logic
    );
end and_gate;
```

Architecture

- Each entity has one or more architectures.
- The architecture defines an implementation of the entity.

Architecture Declaration

```
architecture arch_name of entity_name is
   …declarations…
begin
   …concurrent VHDL…
end arch_name;
```

Example Architecture

```
architecture structural of and_gate is
begin
  output <= x and y;
end structural;
```

Libraries

- In VHDL we compile entities, and packages into libraries.
- The working library is generally known as '**`work`**'.
- The standard VHDL types and operators are in a library known as '**`std`**'.
- We 'include' a particular library by using this line at the top of our file:

```
        library library_name;
```

- The '**`work`**' and '**`std`**' libraries are always available.

Use

- A library may contain various different entities and packages.
- The 'use' clause makes part or all of a particular package visible. e.g.

```
use library_name.package_name.item_name;
```

- We can make all of a package visible by using the specifier "**all**". eg.

```
use ieee.std_logic_1164.all;
```

Standard Types

- VHDL defines a set of standard types and operators.
- These are contained in the '`std`' library.
- The types include:
  - bit ('0','1')
  - boolean (TRUE, FALSE)
  - integer (-??? To +???)
  - natural (0 to integer'high)
  - positive (1 to integer'high)
  - character (ASCII characters e.g. 'D')
  - time (including units e.g. 10us, 15ps…)

The IEEE library

- The IEEE defines a standard library containing several packages with useful functions and types.
- To get to the IEEE library we use:

```
library ieee;
```

- To make one of the IEEE packages visible:

```
use ieee.std_logic_1164.all;
```

The std_logic_1164 package

- Provides more powerful types than bit
  - std_ulogic (‘U’, 'X', '1', '0', 'Z', 'W', 'H', 'L', '-‘)
    - 0 = forcing 0
    - 1 = forcing 1
    - Z = high impedance
    - L = weak 0 (pull down)
    - H = weak 1 (pull up)
    - U = uninitialised
    - W = weak unknown (strong will override)
    - -   = Don't care (no impact on a comparison)
    - X = forcing unknown (will be 1 or 0 but can't tell which)

The std_logic_1164 package

- std_logic
  - same as std_ulogic, but `resolved'
    - defined even when two outputs are connected together
- subtypes
  - X01 (subtype of std_ulogic: ‘X’, ‘0’, ‘1’)
  - X01Z …
  - UX01 …
  - UX01Z …

General notes

- VHDL source files generally have the filename extension '**`.vhd`**' or '**`.vhdl`**'
- Typically we use one file per entity or package.
- A comment can be placed on a line by preceding it with '**`--`**'

Putting it all together…

```
library ieee;                       -- Use the IEEE library
use ieee.std_logic_1164.all; -- std_logic_1164 package

entity and_gate is
  port (
    x : in std_logic;      -- These are the two input ports
    y : in std_logic;
    output : out std_logic  -- This is the output
    );
end and_gate;

architecture structural of and_gate is
begin
  -- Simple concurrent VHDL assignment
  output <= x and y;
end structural;
```