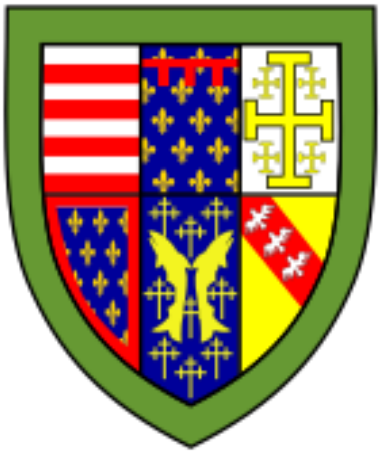
Queens' College Cambridge

Digital Electronics



Alistair O'Brien

Department of Computer Science

January 5, 2020

Contents

1 Combinational Logic

1.1 Boolean Algebra

A Boolean Algebra is a 3-tuple $(B, +, \cdot)$ such that B is a set of at least two elements and $+$ and $\cdot$ are binary operations, that is

$$+ : B \times B \mapsto B \text{ and } \cdot : B \times B \mapsto B.$$

and satisfies the following axioms:

1. **A1**: Commutative Law. For all $a, b \in B$

   $$a + b = b + a \text{ and } a \cdot b = b \cdot a.$$

2. **A2**: Associative Law. For all $a, b, c \in B$

   $$a + (b + c) = (a + b) + c \text{ and } (a \cdot b) \cdot c = a \cdot (b \cdot c).$$

3. **A3**: Distributive Law. For all $a, b, c \in B$

   $$a + (b \cdot c) = (a + b) \cdot (a + c) \text{ and } a \cdot (b + c) = (a \cdot b) + (a \cdot c).$$

4. **A4**: Existence of identity elements. The set B has two distinct identity elements, denote as 0 and 1, such that for all $a \in B$

   $$a + 0 = 0 + a = a.$$

   $$a \cdot 1 = 1 \cdot a = a.$$

5. **A5**: Existence of a complement. For every element $a \in B$ there exists an element $\overline{a} \in B$ such that

   $$a + \overline{a} = 1.$$

   $$a \cdot \overline{a} = 0.$$

1.1.1 Boolean Algebra Theorems

Theorem 1.1.1. (**Idempotent Law**) *For all* $a \in B$

$$a + a = a$$
$$a \cdot a = a$$

Theorem 1.1.2. (**Involution Law**) *For all* $a \in B$

$$\overline{\overline{a}} = a.$$

Theorem 1.1.3. (**Absorption Law**) *For all* $a, b \in B$

$$a + a \cdot b = a$$
$$a \cdot (a + b) = a$$

Theorem 1.1.4. (**DeMorgan's Law**) *For all* $a, b \in B$*, we have*

$$\overline{a + b} = \overline{a} \cdot \overline{b}$$
$$\overline{a \cdot b} = \overline{a} + \overline{b}$$

Theorem 1.1.5. (**Generalised DeMorgan's Law**) *Let* $\{x_1, x_2, \ldots, x_n\} \subseteq B$*. Then*

$$\overline{x_1 + x_2 + \cdots + x_n} = \overline{x_1} \cdot \overline{x_2} \cdots \overline{x_n}$$
$$\overline{x_1 \cdot x_2 \cdots x_n} = \overline{x_1} + \overline{x_2} + \cdots + \overline{x_n}$$

1.2 Logic Gates

1.2.1 Bubble Logic and Gate Interchangeability

- A gate is said to be universal if it can implement any Boolean function without the need to use any other gate.
- `NOR` and `NAND` gates are `universal`, since we can construct `NOT`, `AND` and `OR` gates using them (by DeMorgan's law).
- DeMorgan's law is used when applying "bubble logic" to circuits. It is the process of applying two consecutive `NOT` operations between the output of one gate and the input of another. Benefit: Ensures circuit only uses `NAND` or `NOR` gates, which are simpler and faster.

1.3 Logical Minimisation

1.3.1 Disjunctive and Conjunctive Normal Forms

Definition 1.3.1. (**Minterm**) For a Boolean function f of n variables $x_1, x_2, \ldots, x_n$, a *product term* $p(x_1, x_2, \ldots, x_n)$ in which each of the n variables appears **once** (either in its complemented or uncompleted form) and for which $f(x_1, x_2, \ldots, x_n) = m(x_1, x_2, \ldots, x_n) = 1$ is called a *minterm* $m(x_1, x_2, \ldots, x_n)$.

Definition 1.3.2. (**Disjunctive Normal Form**) For a Boolean function f, we can express f as function f' such that

$$f'(x_1, \ldots, x_n) = m_1(x_1, \ldots, x_n) + \cdots + m_k(x_1, \ldots, x_n),$$

where $m_i \in M_{in}$ and M_{in} is the set of minterms of f. Then f' is said to be the **disjunctive normal form** of f.

A Boolean function f expressed as function f' such that

$$f'(x_1, \ldots, x_n) = p_1(x_1, \ldots, x_n) + \cdots + p_k(x_1, \ldots, x_n),$$

where $p_1, \ldots, p_k \in P$, then f' is said to be the **sum of products** (SoP) form of f.

Definition 1.3.3. (**Maxterms**) For a Boolean function f of n variables $x_1, x_2, \ldots, x_n$, a *sum term* $s(x_1, x_2, \ldots x_n)$ in which each of the n variables appears **once** (either in its complemented or uncomplemented form) is called a *maxterm* $m(x_1, x_2, \ldots, x_n)$.

Method 1.3.1. (**Finding max terms**)

1. Find the min terms $\overline{M_{in}}$ of the complemented function $\overline{f}$.
2. Apply DeMorgan's laws and the involution law to the min terms to get the max terms M_{ax}.

Definition 1.3.4. (**Conjunctive Normal Form**) For a Boolean function f, we can express f as a function f' such that

$$f'(x_1, \ldots, x_n) = m_1(x_1, \ldots, x_n) \cdots m_k(x_1, \ldots, x_n),$$

where $m_i \in M_{ax}$ and M_{ax} is the set of maxterms of f. Then f' is said to be the **conjunctive normal form** of f.

A Boolean function f expressed a function f' such that

$$f'(x_1, \ldots, x_n) = s_1(x_1, \ldots, x_n) \cdots s_k(x_1, \ldots, x_n).$$

where $s_1, \ldots, s_k \in S$, then f' is said to be the **product of sums** (PoS) form of f.

1.3.2 Karnaugh Maps

- Visual method for simplification of a Boolean function f of up to 5 variables.
- **Grey Code**: A **grey code** is a sequence of bits in which only a single bitch is changed in each consecutive sequence

$$00 \mapsto 0\underbrace{1}_{\text{new bit}} \mapsto \underbrace{1}_{\text{new bit}}1 \mapsto 1\underbrace{0}_{\text{new bit}}.$$

- **Cover**: A term t that has a corresponding set of minterms M_{in} is said to cover a minterm m if and only if $m \in M_{in}$. E.g. from the K-Map above $y \cdot z$ has the corresponding set of minterm $M_{in} = \{\overline{x} \cdot y \cdot z, x \cdot y \cdot z\}$, then $y \cdot z$ covers $x \cdot y \cdot z$ and $\overline{x} \cdot y \cdot z$.
- **Implicant**: A product term t is said to be an implicant of a Boolean function f if $t(x_1, \ldots, x_n) = 1$ then $f(x_1, \ldots, x_n) = 1$
- **Prime Implicant**: A term t is said to be a prime implicant that cannot be covered by a more general (more reduced) implicant.
- **Essential Prime Implicants**: A term t is said to be an essential prime implicant if it covers a minterm m such that no other prime implicant t' covers m.
- **Covering Set**: The covering set C of a Boolean function f is the set of prime implications which contains all essential prime implicants and any other prime implicants required to cover all minterms.

Method 1.3.2. (**Karnaugh Map**)

1. Form a 2-dimensional table as above with 2^n cells where n is the number of input variables in the Boolean function f.

2. Write out the Gray code.
3. Fill in the corresponding minterms M_{in} of f with a 1.
4. Find all prime implicants and then find the essential prime implicants.
    - Note that implicants must be rectangular groups that have sides of length $2^n \times 2^m$ for $m, n \in \mathbb{N}$.
    - Larger the group, the fewer variables in the term for the group
    - Groups can wrap around edges / corners of the K-Map
5. Form a covering set C.
6. `OR` all the terms $t \in C$ to produce a SoP form for f.

Product Of Sums Simplification

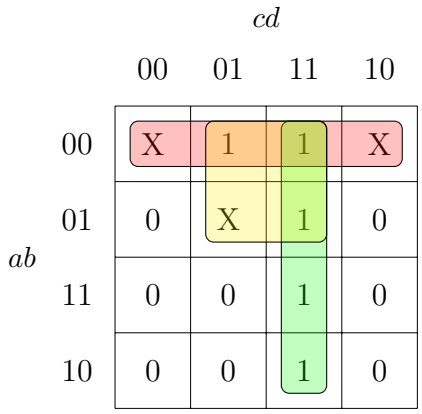
- K-Map gives simplified functions in SoP form, so the Boolean function can be implemented using `AND`, `IN` and `NOT` gates (or only `NAND` gates).
- PoS simplification is required, so the Boolean function can be implemented using `NOR` gates.

Method 1.3.3. (**PoS Simplification**)

1. Find a simplified SoP form for $\overline{f}$ using the K-Map method above.
2. Apply DeMorgan's law and the involution law to produce a PoS form for f.
3. (*Optional*): Apply bubble logic if `NOR` implementation is required.

Don't Care Conditions

- A don't care condition is an input $x_1, x_2, \ldots, x_n$ for a given Boolean function f for which the output does not matter.
- Used in cases where the output doesn't matter or the input can never occur.
- In K-Maps don't care conditions are entered as X's, as shown below:



1.3.3 The Quine-McCluskey Method

1. Group minterms m and don't care conditions by the number of 1s that the binary representation of the minterm contains.
2. Compare each element of a group to each member of the group below it.
3. If two minterms differ by only one bit, tick each minterm and copy the minterm to the next column, replacing the differenting bit with $-$.
4. After all minterms have been compared, mark any unticked minterms with a $*$.
5. Repeat steps 2 - 4 on the new column, but the $-$s must also match.
6. Continue until all minterms are ticked or started.
7. Create a prime implicant chart.
8. Find the essential prime implicants by looking for columns with only one X. Mark the rows where these appear.
9. Cross out the row and columns containing a cross for the essential prime implicants.

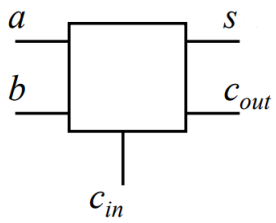
10. Cross out columns for any other crosses in the essential prime implicant's row.

11. Find a way to cover the remaining minterms using as few prime implicants as possible.

1.4 Adders

1.4.1 Full Adders

- Adds two single bit numbers a, b with a carry input c_{in}.
- Produces a sum s and carry output c_{out}.

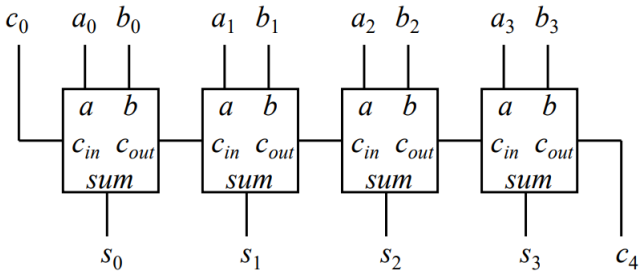


- We have

$$s = c_{in} \oplus (a \oplus b)$$
$$c_{out} = c_{in} \cdot (a \oplus b) + a \cdot b = a \cdot b + (a + b) \cdot c_{in}$$

1.4.2 Ripple Carry Adder

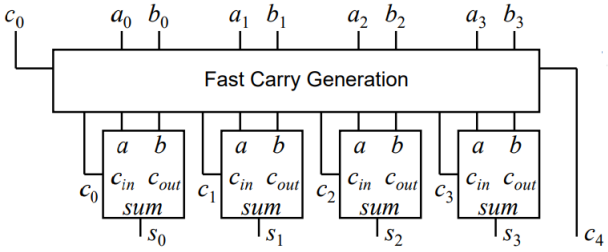
- A ripple carry adder is an n-bit adder where we connect the c_{out}^{i} to c_{in}^{i+1}.



- Compositional approach is slow due to gate propagation delay produced by each full adder.
- Time taken to add 2 number with n bits takes τn seconds, where τ is gate propagation delay produced by a full adder.

1.4.3 Fast Carry Generation

- To reduce propagation delay, we can generate carry signal independently, hence time taken to add 2 numbers with n bits is $\tau + \kappa$ where κ is time taken to calculate carry signals.



- To formulate carry logic, consider c_i and c_{i+1}.

c_i	a_i	b_i	s_i	c_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Split the table into 3 classes.

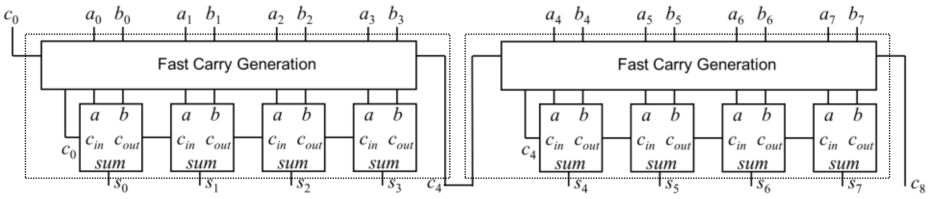
1. **Carry Kill**. The rows where the carry out is always 0.
$$k_i(a_i, b_i) = \overline{a_i} \cdot \overline{b_i}.$$
2. **Carry Propagate**. The rows where the carry out is equal to the carry in.
$$p_i(a_i, b_i) = a_i \oplus b_i.$$
3. **Carry Generation**. The rows where the carry out is generated independently of the carry in.
$$g_i(a_i, b_i) = a_i \cdot b_i.$$

So
$$c_{i+1}(a_i, b_i) = g_i(a_i, b_i) + c_i(a_{i-1}, b_{i-1}) \cdot p_i(a_i, b_i).$$

This recurrence relation gives us
$$c_n = G_{n-1} + P_{n-1} \cdot c_0.$$

where
$$G_n = \sum_{i=0}^{n} g_i \cdot \prod_{j=i+1}^{n} p_j$$
$$P_n = \prod_{i=0}^{n} p_i$$

- For large n, G_n and P_n become complex. So we use 4-bit fast carry generation blocks used to generate c_{i+4} and connect them in a ripple carry style. This is a tradeoff between complexity and speed.



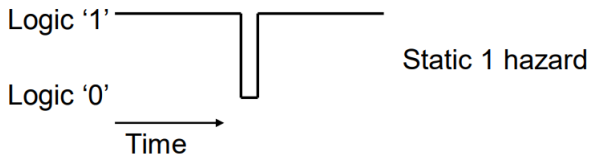
1.5 Gate Propagation Delay and Hazards

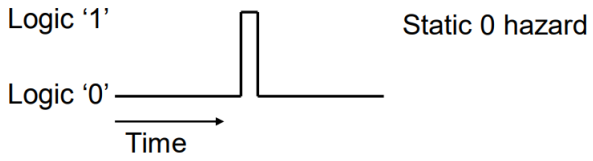
- Multi-level logic circuits reduce implementation complexity and number of gates required. Cumulative gate propagation delay produces **hazards**, brief unwanted changes in output due to multi-level logic.

  Definition 1.5.1. (**Gate Propagation Delay**) The gate propagation delay is defined as the time taken τ for a given gate's output to respond to a change in it's inputs.

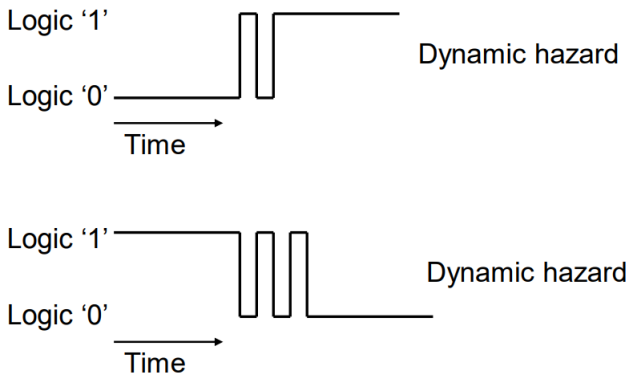
- Two types of hazards:

  1. **Static Hazards**: The output of the circuit undergoes one change of state when one input changes when it is supposed to remain unchanged.





  2. **Dynamic Hazards**: The output of the circuit changes more than once when it is supposed to change just once.



1.5.1 Hazard Removal

1. **Removing Static 1 Hazard**: Suppose the circuit with Boolean function f has a static 1 hazard. Draw the K-Map of f, then add another term t to the function f such that t covers the essential terms and removes the gap between the two essential terms t_1, t_2. The hazard occurs when we transition from t_1 to t_2 (or vice versa)

2. **Removing Static 0 Hazard**: Suppose the circuit with Boolean function f has a static 0 hazard. Draw the K-Map of $\overline{f}$, then add another term t to the function $\overline{f}$ such that t covers the essential terms of $\overline{f}$.

1.6 Selectors

1.6.1 Multiplexor

- A multiplexer (MUX) is a device that selects one of many inputs to output based on a control signal.
- Multiplexer consists of a decoder and an input selector.
- n-to-1 (n:1) MUX requires $\log_2 n$ control signal bits.
- MUX can be used to implement Boolean function that are expressed in their DNF. Implementation can be simplified by using one variable for a data input (e.g. 8:1 to 4:1)

1.6.2 Demultiplexor

- A demultiplexor (DEMUX) performs the inverse of a MUX, maps one input to one of it's output based on a control signal.
- Demultiplexor consists of a decoder and an output selector.
- 1-to-n (1:n) DEMUX requires $\log_2 n$ control signal bits.

1.6.3 Decoders

- A decoder is closely related to a demultiplexor, but it's input is set to logic 1.
- Used as a control signal, so only one system is active at a time based on a control signal
- 1-to-n (1:n) decoder requires $\log_2 n$ control signal bits.
- A 1:n decoder will generate all the minterms having n variables $\implies$ can implement Boolean functions in DNF by `OR`ing the required minterms.

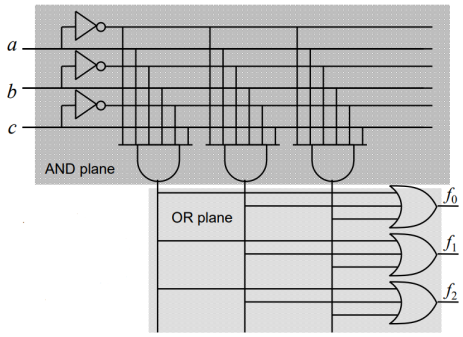
1.7 Memory

1.7.1 ROM

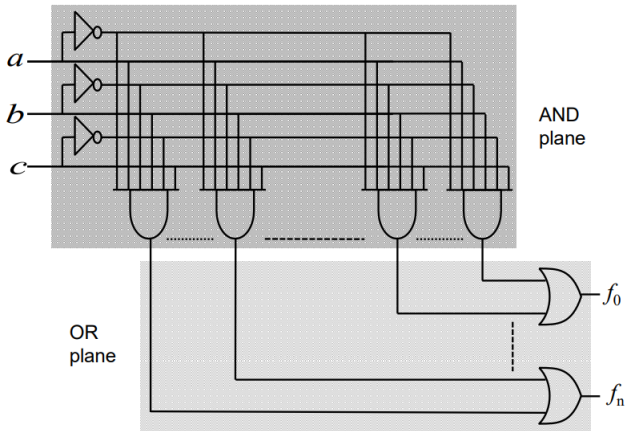
- Written once, read at will.
- Non-volatile
- Minterms can be hard coded in ROM, forming a lookup table.
- Address of n bits used to specify location of output.
- Multiple Boolean functions can be implemented with a single ROM chip
- ROMs can be inefficient if only a few minterms are required, as alot of storage will be filled with 0s

1.7.2 PLA / PAL

- A programmable logic array (PLA) is a device that generates the required minterms using a separate programmable `AND` plane. The outputs from the `AND` plane are the `OR`ed together in the separate programmable `OR` plane.



- Planes are programmed by selectively removing connections controlled by fuses or memory bits.
- A simple version of PLA are programmable array logic (PALs), which have a programmable `AND` plane, but fixed `OR` plane. Simplifies the structure, but at a cost of efficiency.



1.7.3 Busses and Tristate Buffers

- Outputs from ROM, DRAM, SRAM will share data and data busses to the CPU. If more than one device outputs to the same bus $\implies$ data corruption.
- Use tristate buffer on data output of memory devices (output can be 0, 1 or high impedance (not connected)). Used to control which device is allowed to read from or output to a bus $\implies$ only one device uses bus at a time. Controlled using OE (output enable) signal.
- Other control signals are used:
  - WE write enable. Determines whether data is written or read from memory device
  - CS chip select. Determines if a chip is active $\implies$ reduces power consumption

2 Sequential Logic

- **Combination logic circuits** are logic circuits that do not contain loops and their output **only** depends on the current input.
- **Sequential logic circuits** are logic circuits whose output depends on the current input and previous states $\implies$ sequential logic implicitly contains memory elements.

2.1 Memory Elements, Synchronous and Asynchronous Circuits

- A **memory element** is any sequential circuit that stores data (typically one bit per element).
- A snapshot of a memory element is a **state**.
- A memory element with two internal stable states is a **bistable** memory element. e.g. flip-flops and latches.

  Definition 2.1.1. (**Synchronous Circuit**) A synchronous circuit is a sequential circuit in which the output is constrained to change only at a time specified by a global enabling signal (a clock).

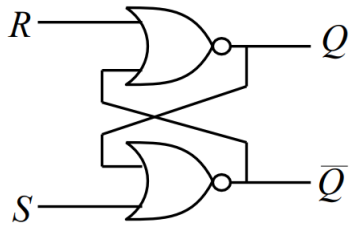
  Definition 2.1.2. (**Asynchronous Circuit**) An asynchronous circuit is any circuit in which the output state changes occur directly in response to changes in the inputs.

2.2 Flip-Flops and Latches

- Flip-flops / latches can either be level-triggered (asynchronous) or edge-triggered (synchronous).

2.2.1 RS Latch

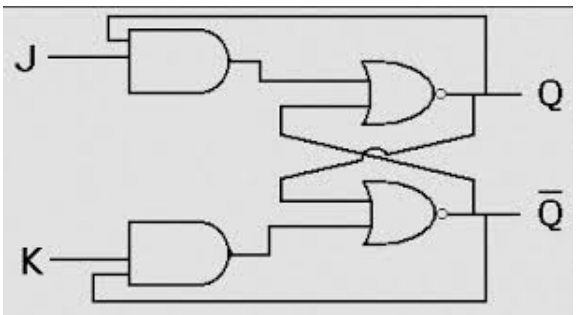
- RS ("Reset - Set") latch is an asynchronous memory element.
- Has 2 inputs: Reset (R) and Set (S) and 2 outputs: Q and $\overline{Q}$.



R	S	Q_{i+1}	$\overline{Q}_{i+1}$	comment
0	0	Q_i	$\overline{Q}_i$	hold
0	1	1	0	set
1	0	0	1	reset
1	1	0	0	illegal (since $Q = \overline{Q} = 0$)

2.2.2 JK Latch

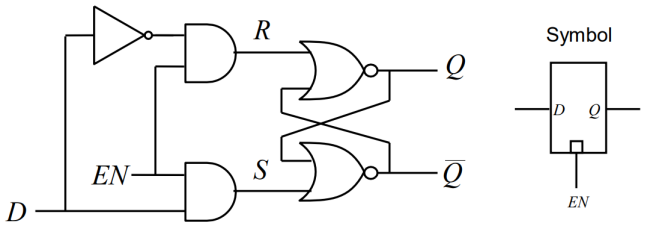
- The JK latch is an RS latch that is made to toggle its output when passed the illegal input combination $R = S = 1$.



J	K	Q_{i+1}	comment
0	0	Q_i	hold
0	1	0	reset
1	0	1	set
1	1	$\overline{Q}_i$	toggle

2.2.3 Transparent D Latch

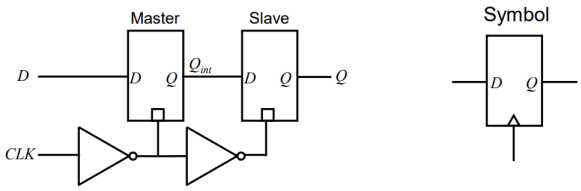
- A transparent D latch is a synchronous RS latch.



- R, S inputs are replaced with D input. Prevents $R = S = 1$ while providing same functionality.
- transparent D latch exhibits transparent behavior when $EN = 1$. When $EN = 0$ latch is opaque and holds current state.
- latch is level-triggered (as opposed to edge-triggered).

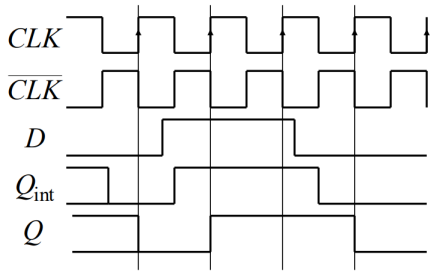
2.2.4 Master-Slave Edge-Triggered D Flip-Flop

- To make a transparent D latch edge triggered, use a master-slave configuration.
- Connect the latches in series and input the CLK input to one of them. This is a master-slave configuration, since the second (slave) latch changes only in response to a change in the first (master) latch.



- **Explanation**:

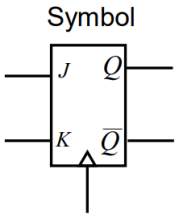
  1. When CLK is logic 0, the $\overline{CLK}$ input into the master latch is logic 1.
  2. As CLK transitions from logic 0 to 1, the $\overline{CLK}$ input into the master latch is logic 0 and the value is now locked into the master latch.
  3. The $\overline{\overline{CLK}}$ input of the slave D latch transitions from 0 to 1. This allows the value locked in the master latch to pass through to the slave latch.
  4. When CLK transitions from logic 1 to 0, the output of the slave is locked. The master can now accept new D values, while the slave stores the value seen at the last rising edge of CLK.



- Removing the first `NOT` gate in the circuit produces a falling-edge triggered master-slave D flip-flop

2.2.5 JK Flip-Flop

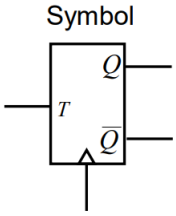
- JK FF is a edge-triggered synchronous JK latch.



- Next state logic: $D = J \cdot \overline{Q} + J \cdot \overline{K} + \overline{K} \cdot Q$.

2.2.6 T Flip-Flop

- If the T input is logic 1, the T flip-flop toggles when ever the clock input CLK is logic 1. When T is logic 0, the flip-flop holds it's current state.



- Implemented using JK FF with $J = K$.
- Next state logic: $D = T \oplus Q$.

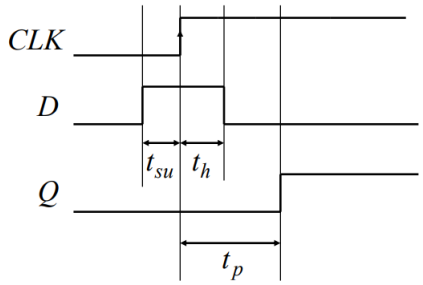
2.2.7 Asynchronous Inputs

- Asynchronous inputs are inputs that take effect independently of any clock / enable input. e.g.
  - Reset / Clear - forces Q to 0
  - Preset / Set - forces Q to 1
- Used to force circuit into a known state.

2.2.8 Timing

Definition 2.2.1. (**Setup Time**) The setup time t_{su} is the minimum time before clock edge that data input must change to register.

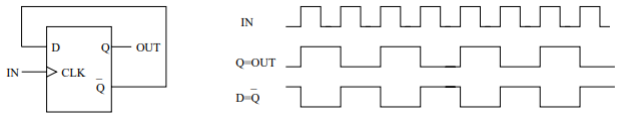
Definition 2.2.2. (**Hold Time**) The hold time t_h is the minimum time after clock edge that data input must remain constant.



2.3 Counters and Shift Registers

2.3.1 Divide by Two Counters

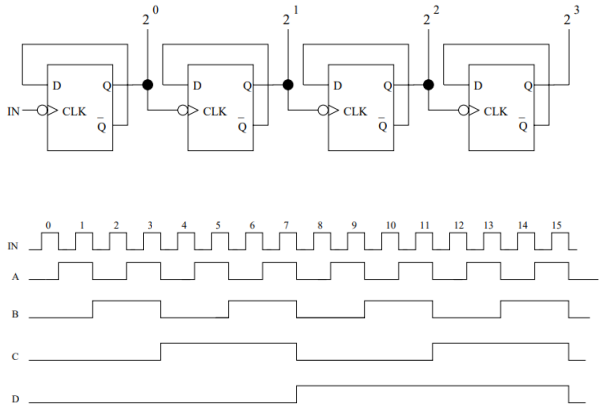
- A edge-triggered D flip-flop is used to implement divide-by-2 counter with next state logic $D = \overline{Q}$.



- Alternatively, T flip-flops with $T = 1$ can be used.

2.3.2 Ripple Counters

- n divide-by-2 counters can be connected in series to produce a "ripple counter", with clock inputs `NOT`ed.
- Also produces a divide-by-2^n counter.



- Benefit: Easy to assemble compared to synchronous counters.
- Disadvantage: CLK must propagate through entire chain of flip-flops before correct result is achieved $\implies$ CLK frequency is limited to prevent miscounting. $T < n\tau$ where n is the number of flip-flops and τ is propagation delay of a single flip-flop.

2.3.3 Synchronous Counters

- Synchronous counters prevent propagation delay from cascading through the counter by having all clock inputs to all the flip-flops connected to a single CLK signal. This ensures *synchronicity*.
- Disadvantage: Requires more complex next-state combination logic.
- Next state logic: For the i-th flip-flop input, we have

$$D_i = Q_i \oplus A_i.$$

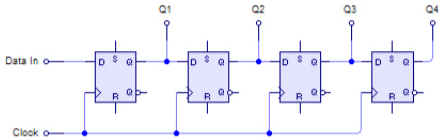
where

$$A_i = \sum_{j=0}^{i-1} Q_j$$
$$D_0 = \overline{Q_0}$$

2.3.4 Shift Registers

- A shift register is a chain of D flip-flips where output of a flip-flop Q is connected to input D of the successor.
- So bit array stored in register is shifted by one position each clock cycle.

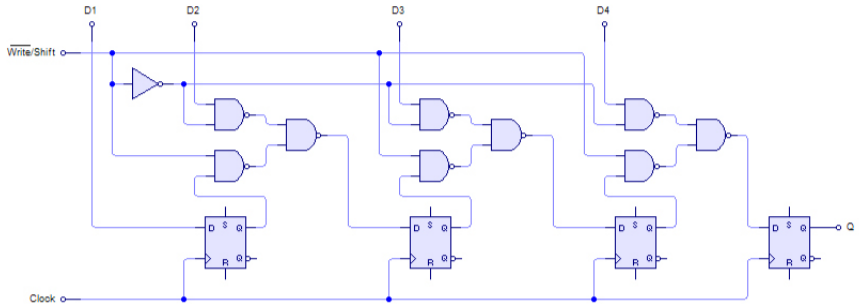
Serial-In Parallel-Out Shift Registers



- ith input bit shifts to the $N - i$ th output after N clock cycles. $\implies$ serial data converted into a parallel format.

Parallel-in Serial-Out Shift Registers

- When $\overline{Write}$ is logic 0, we can write the parallel data into the the shift register.
- When $Shift$ is logic 1, for N bits, we have N clock cycles to output the parallel data in serial order.

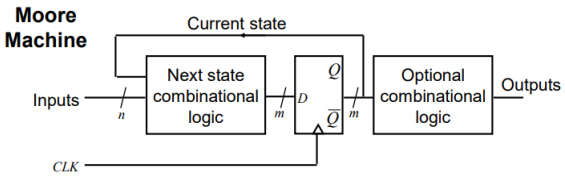


- The combinational logic prior to each flip-flop can be replaced with a 2:1 MUX with a $\overline{Write}/Shift$ input as the control signal.
- Alternatively, preset and clear asynchronous inputs may be used to write the data into the shift register.

2.4 State Machines

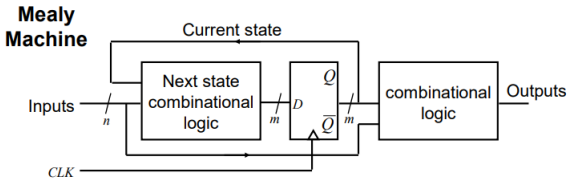
- **Finite State Machine**: A deterministic machine that produces outputs which depend on it's internal state and external inputs.
- Two types of FSMs: Moore machines and Mealy machines.

1. **Moore Machines**: A finite state machine whose outputs are determined by it's current state.



Moore machines are synchronous. Guaranteed timing characteristics and are glitch free.

2. **Mealy Machine**: A finite state machine whose outputs are determined by it's current state and the current inputs.



Mealy machines are asynchronous. They are more powerful.

2.4.1 State Assignment

- If we have n states, we require at least $\log_2 n$ flip-flops to encode states.
- **Sequential State Assignment**: We assign the states in an increasing natural binary count.

  Uses n flip-flops for 2^n states.
- **Sliding State Assignment**: Each next state of a given state is shifted to the left/right by 1 e.g.

c	b	a	c'	b'	a'
0	0	0	0	0	1
0	0	1	0	1	1
0	1	1	1	1	1
1	1	1	1	1	0
1	1	0	1	0	0
1	0	0	0	0	0

Uses n flip-flops for $2n$ states.

- **State Register Assignment**: States are assigned to produce a shift register. e.g.

c	b	a	c'	b'	a'
0	0	1	0	1	0
0	1	0	1	0	0
1	0	0	0	0	1

Uses n flip-flops for n states.

- **One Hot State Encoding**: One flip-flop is assigned for each state. Trade off between efficiency, cost and complexity. (less complex, higher cost, more efficient)

2.4.2 Self-Starting

- Illegal (unused) states are considered as don't care states during the design process. But FSM could possibly start in one of these illegal states.
- Check whether FSM will reach a legal state from any of the illegal states. If so, the FSM is self-starting.
- If not, then add additional combinational logic to transition from an illegal state to a legal state or use asynchronous set and reset inputs to set the flip-flops into a valid state upon startup. (valid state could be stored in ROM).

2.4.3 Elimination of Redundant States

Method 2.4.1. (**Elimnation of Redundant States**)

1. Draw a table of current state, next state for each possible input and output for each possible input. e.g.

Current state	Next state		Output	
	$X = 0$	$X = 1$	$X = 0$	$X = 1$
A	A	B	0	0
B	C	C	0	0
C	A	B	0	0

2. Remove any states that cannot be distinguished from another (e.g. A and C).

3. Update the table. e.g.

Current state	Next state		Output	
	$X = 0$	$X = 1$	$X = 0$	$X = 1$
A	A	B	0	0
B	A	A	0	0

4. Repeat until all rows are distinct.

2.4.4 Programmable FSMs

GAL

- Generic Array Logic (GAL) devices are similar to PALs but include a D flip-flop on the output of the `OR` gates.
- The output of the flip-flops are inputs to the `AND` plane (used for next state logic).

FPGAs

- Field Programmable Gate Arrays (FPGAs) are the latest type of programmable logic.
- An array of configurable logic blocks (CLBs) surrounded by input-output blocks (IOBs).
- Routing channels controlled by programmable switching matrices (SMs) connect CLBs to CLBs and IOBs.
- CLBs contain lookup tables (LUT), multiplexers and D flip-flops.
- Programmed by specifying contents of LUTs and select signals for MUXs.

3 Circuits

3.1 Electricity

3.1.1 Current

Definition 3.1.1. (**Current**) An electric current I is defined as the rate at which charge flows through a surface (the cross section of a wire, for example).

$$I = \frac{\mathrm{d}Q}{\mathrm{d}t}.$$

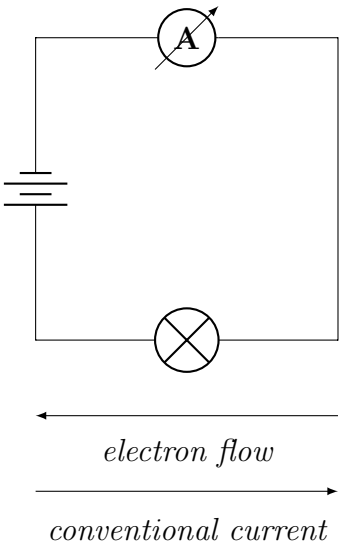
measured in **amperes** (A).

3.1.2 Moving Charges

- Charge has two types, **positive** (+) and **negative** (−), carried by protons and electrons respectively.
- Charge is conversed; the net charge of an isolated system is constant.
- The unit of charge is coulombs (C).
- Charge is quantized; it comes in integer multiples of the **elementary charge** $e = 1.6 \times 10^{-19}$ coulombs.
- An electron has a charge of $-e$ and a proton has a charge of $+e$.

Conventional Current

- In a metal lattice, *delocalized* electrons flow when one end of the conductor is positive and the other is negative $\implies$ electrons flow to positive end.
- (Conventional) current flows from positive terminal to the negative terminal, the opposite direction to electron flow.



Mean Drift Velocity

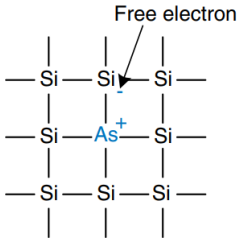
- As electrons move in a wire, they collide with $+ve$ metal ions $\implies$ electrons don't have a constant velocity. Consider mean velocity.

  Theorem 3.1.1. The current I in a conductor with cross sectional area A, number density n and mean drift velocity of charge carriers $\overline{\mathbf{v}}$ of charge q is given by
  $$I = Anq\|\overline{\mathbf{v}}\|.$$

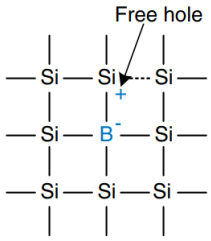
  Proof. A conductor of volume V with number density n has charge
  $$Q = nqV.$$
  So by definition of current we have
  $$I = \frac{\mathrm{d}}{\mathrm{d}t} nqV.$$
  Note that
  $$\|\overline{\mathbf{v}}\|\,\mathrm{d}t = \frac{V}{A}.$$
  Hence $I = Anq\|\overline{\mathbf{v}}\|$. □

3.1.3 Materials

- **Insulators**:
  - Crystalline structure.
  - Electrons are strongly bound, unable to move.
  - When pd is applied, crystal will distort, but no charge flow occurs.
- **Conductors**:
  - Lattice structure.
  - Electrons are weakly bound, free to move.
  - When pd is applied, lattice will distort, charge flow occurs.
- **Semiconductors**(Silicon):
  - Group 4 element (4 electrons in valent shell), so crystalline lattice. No free electrons.
  - By adding *dopant* atoms, we can introduce free charges.
  - **n-type silicon**:
    - Doped with Arsenic (group 5) that donates a free electron.



  - **p-type silicon**:
    - Doped with Boron (group 3). Neighboring electron from an Si atom moves to B atom, producing B^- ion and a *hole* at the Si atom.



* Hole can move about the lattice. A hole is a lack of an electron $\implies$ electrons are moving.
* Hole acts like a positive charge.

3.2 Circuit Theory

3.2.1 Electrical Resistance

Definition 3.2.1. (**Electrical Resistance**) Resistance is defined as the measure of the degree to which a conductor opposes the flow of charge carriers through that conductor, measured in **ohms** (Ω).

$$R = \frac{V}{I}.$$

Theorem 3.2.1. (**Ohm's Law**) Ohm's law states for a conductor at a constant temperature, the potential difference across the conductor is directly proportional to the current through the conductor.

$$V \propto I.$$

The *current-voltage* (I-V) graph of an ohmic (linear) device consists of a straight line through the origin with a positive gradient.

3.2.2 Potential difference and Electromotive Force

Definition 3.2.2. (**Potential Difference**) Potential difference V is defined as the rate of electrical energy transferred W per unit of charge Q.

$$V = \frac{\mathrm{d}W}{\mathrm{d}Q}.$$

measured in **volts** (V).

Potential difference is described as the work done by charge carriers, whereas e.m.f is used to describe the work done on charge carriers.

Definition 3.2.3. (**Electromotive Force**) Electromotive force ε is defined as the rate of energy W transferred to electrical energy per unit of charge Q.

$$\varepsilon = \frac{\mathrm{d}W}{\mathrm{d}Q}.$$

measured in **volts** (V).

3.2.3 Kirchhoff's Laws

Theorem 3.2.2. (**Kirchhoff's first law**) Kirchhoff's first law states that for any point in an electrical circuit, the sum of directed currents at the point is zero.

$$\sum I_i = 0 \tag{3.1}$$

Proof. Follows from conservation of charge. □

Theorem 3.2.3. (**Kirchhoff's second law**) Kirchhoff's second law states the sum of directed potential differences around any closed loop is zero.

$$\sum V_i = 0 \tag{3.2}$$

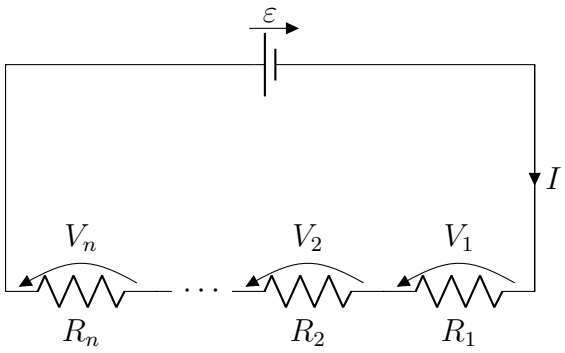
Proof. Follows from the conservation of energy. □

3.2.4 Combining Resistors

Theorem 3.2.4. (**Resistors in series**) n resistors connected in a series arrangement with resistances $R_1, R_2, \ldots, R_n$ are isomorphic to a single resistor with a resistance of R_s ohms, such that

$$R_s = \sum_{k=1}^{n} R_k. \tag{3.3}$$

Proof. Consider the following circuit.



Applying Kirchhoff's second law, we have

$$\varepsilon = \sum_{k=1}^{n} V_k$$

and from Kirchhoff's first law, the current through each resistor is equal. If we now apply Ohm's law, we obtain

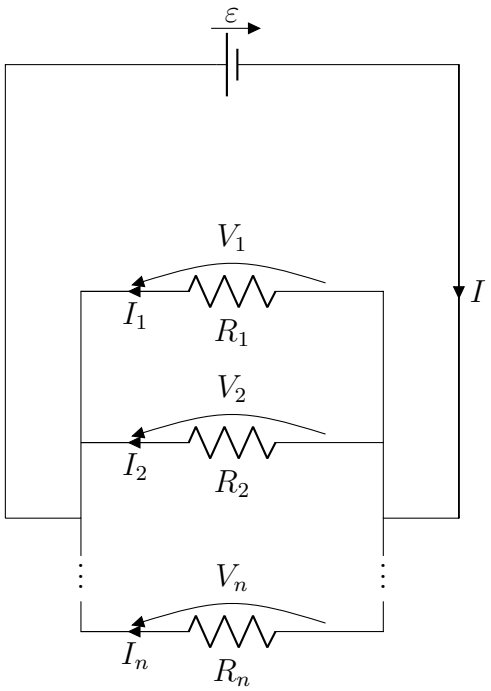
$$IR_s = \sum_{k=1}^{n} IR_k$$
$$R_s = \sum_{k=1}^{n} R_k$$

□

Theorem 3.2.5. (**Resistors in parallel**) n resistors in parallel with resistances $R_1, R_2, \ldots, R_n$ are isomorphic to a single resistor with a resistance of R_p ohms, where

$$\frac{1}{R_p} = \sum_{k=1}^{n} \frac{1}{R_k}. \tag{3.4}$$

Proof. Consider the following circuit.



Applying Kirchhoff's first and second law, we have

$$I = \sum_{k=1}^{n} I_k \ \textit{and}\ \varepsilon = V_1 = V_2 = \ldots = V_n,$$

combining these with Ohm's Law yields

$$\frac{I}{\varepsilon} = \sum_{k=1}^{m} \frac{I_k}{\varepsilon}$$
$$\frac{1}{R_p} = \sum_{k=1}^{m} \frac{1}{R_k}$$

□

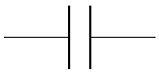
3.2.5 Solving I-V Characteristics

- Connect the device to a potential divider.
- Plot the IV characteristic of the device.

- Plot the IV characteristic of the resistor in the potential divider circuit.
- Find the potential difference V across the device when the resistor characteristic intersects the device characteristic

3.3 Capacitors

Definition 3.3.1. (**Capacitor**) An electrical component used to store and release (separated) electrical charge, consisting of 2 metal plates separated by a layer of insulating material called a dielectric.



3.3.1 Capacitance

Definition 3.3.2. (**Capacitance**) Capacitance is the charge stored per unit of potential difference, measured in **farads** (F).

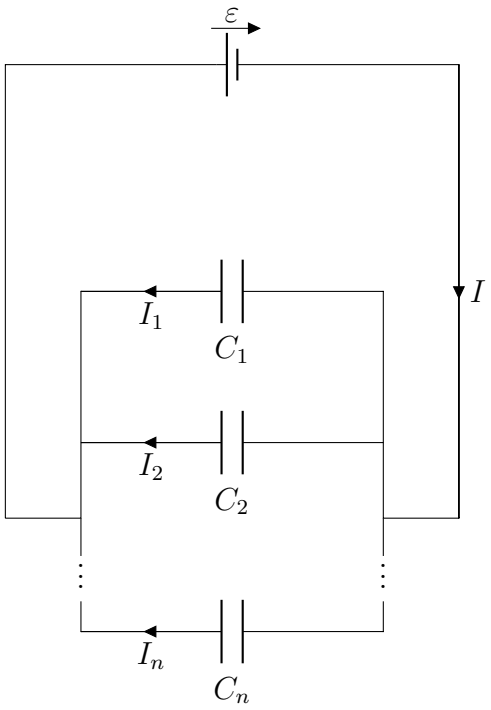
$$C = \frac{Q}{V} \tag{3.5}$$

3.3.2 Combining Capacitors

Theorem 3.3.1. (**Capacitors in parallel**) n capacitors in parallel with potential difference V across then and capacitance's $C_1, C_2, \ldots, C_n$ are isomorphic to a single capacitor with a capacitance C_p, where

$$C_p = \sum_{k=1}^{n} C_k. \tag{3.6}$$

Proof. Consider the following circuit.



Applying Kirchhoff's first law and the definition of current to this circuit, we have

$$I = \sum_{k=1}^{n} I_k \ and \ Q = It$$

combining these, it follows that

$$Q_p = \sum_{k=1}^{n} Q_k$$

where Q_k is the charge stored in the k^{th} capacitor. From the definition of capacitance and Kirchhoff's second law we find that $Q_k = \varepsilon C_k$. Thus

$$C_p = \sum_{k=1}^{n} C_k.$$

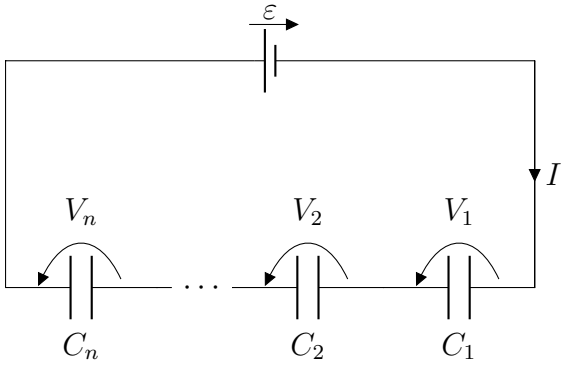
□

Theorem 3.3.2. (**Capacitors in Series**) n capacitors in a series arrangement with potential differences $V_1, V_2, \ldots, V_n$ across them and capacitances

$C_1, C_2, \ldots, C_n$ are isomorphic to a single capacitors with a capacitance C_s, such that

$$\frac{1}{C_s} = \sum_{k=1}^{n} \frac{1}{C_k}. \tag{3.7}$$

Proof. Consider the following circuit.



Applying Kirchhoff's second law, we have

$$\varepsilon = \sum_{k=1}^{n} V_k$$

and from Kirchhoff's first law, the current I entering each capacitor is equal to the current leaving each capacitor, from this it follows that each capacitor stores the same charge Q coulombs. If we now apply the definition of capacitance, we find that $V_k = \frac{Q}{C_k}$, so

$$\frac{Q}{C_s} = \sum_{k=1}^{n} \frac{Q}{C_k}$$
$$\frac{1}{C_s} = \sum_{k=1}^{n} \frac{1}{C_k}.$$

□

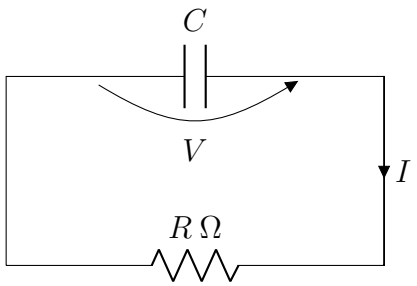
3.3.3 RC Circuits

Theorem 3.3.3. (**Discharging Capacitors**) A discharging capacitor with capacitance C and resistance R in the discharge circuit can be modelled using

$$Q_t = Q_0 \exp\left(-\frac{t}{\tau}\right), \tag{3.8}$$

where Q_t is the charge stored on each plate at time t and $\tau = RC$ (the time constant).

Proof. Let us consider a charged capacitor of Q_0 coulombs in the following discharge circuit



We state Ohm's Law across the resistor, $V = IR$. Now the potential difference V is just q/C, where q is the charge stored on the positive capacitor plate at some time during the discharge. Applying the definition of current to this circuit, we have

$$\begin{aligned} V &= -R\frac{\mathrm{d}Q}{\mathrm{d}t} \\ \frac{1}{Q}\frac{\mathrm{d}Q}{\mathrm{d}t} &= -\frac{1}{\tau}, \end{aligned}$$

This differential equation is solved trivially as follows

$$\begin{aligned}
\int_{Q_0}^{Q_t} \frac{1}{Q}\frac{\mathrm{d}Q}{\mathrm{d}t}\,\mathrm{d}t &= -\int_0^t \frac{1}{\tau}\,\mathrm{d}t \\
\int_{Q_0}^{Q_t} \frac{1}{Q}\,\mathrm{d}Q &= -\int_0^t \frac{1}{\tau}\,\mathrm{d}t \\
\ln\left|\frac{Q_t}{Q_0}\right| &= -\frac{t}{\tau} \\
Q_t &= Q_0 \exp\left(-\frac{t}{\tau}\right)
\end{aligned}$$

□

Corollary 3.3.3.1. (**Current and Voltage Equations**) From definition of capacitance and current, the decay functions for voltage and current can be derived

$$V_t = \frac{Q_0}{C}\exp\left(-\frac{t}{\tau}\right) \tag{3.9}$$

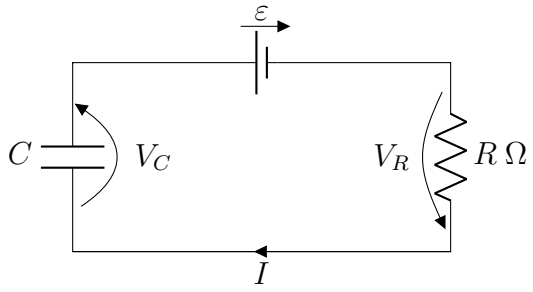
$$I_t = \frac{Q_0}{\tau}\exp\left(-\frac{t}{\tau}\right) \tag{3.10}$$

Theorem 3.3.4. (**Charging Capacitors**) A charging capacitor with capacitance C and resistance R in the charging circuit can be modelled using

$$Q(t) = C\varepsilon\left(1 - \exp\left(-\frac{t}{\tau}\right)\right) \tag{3.11}$$

where Q_t is the charge stored in the capacitor at time t and τ denotes the time constant.

Proof. Consider the following circuit.



Using Kirchhoff's second law and Ohm's law, we write

$$\begin{aligned}
\varepsilon - V_C &= R\frac{\mathrm{d}Q}{\mathrm{d}t} \\
\frac{\varepsilon C - Q}{C} &= R\frac{\mathrm{d}Q}{\mathrm{d}t} \\
\frac{1}{Q - \varepsilon C}\frac{\mathrm{d}Q}{\mathrm{d}t} &= -\frac{1}{\tau}
\end{aligned}$$

Which can be solved as shown below;

$$\begin{aligned}
\int_0^{Q_t} \frac{1}{Q - \varepsilon C}\,\mathrm{d}Q &= -\frac{1}{\tau}\int_0^t \mathrm{d}t \\
\ln\left|\frac{Q_t - \varepsilon C}{-\varepsilon C}\right| &= -\frac{t}{\tau} \\
\frac{Q_t - \varepsilon C}{-\varepsilon C} &= \exp\left(-\frac{t}{\tau}\right) \\
Q_t &= \varepsilon C\left(1 - \exp\left(-\frac{t}{\tau}\right)\right)
\end{aligned}$$

□

Corollary 3.3.4.1. (**Current and Voltage Equations**) From definition of capacitance and current, the exponential charging functions for voltage and current can be derived

$$V_t = \varepsilon\left(1 - \exp\left(-\frac{t}{\tau}\right)\right) \tag{3.12}$$

$$I_t = \frac{\varepsilon}{R}\exp\left(-\frac{t}{\tau}\right) \tag{3.13}$$

Definition 3.3.3. (**Time Constant**) The time constant τ is defined as the time take for the initial value Q_0, I_0, V_0 to decreases to e^{-1} of the initial value when discharging. So

$$\tau = RC.$$

measured in **seconds** (s).

3.4 Diodes and MOSFETs

3.4.1 Diodes

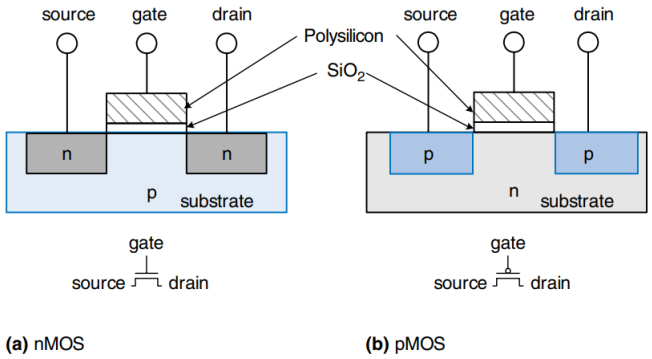
- Combining p-type and n-type silicon produces a p-n junction, known as a *diode*.
- The p-type region is the *anode* and the n-type is the *cathode*.
- When a junction is created, n-type electrons diffuse across to the p-type silicon to fill holes (vice versa).
- This leaves the positive charges in the n-type silicon and negative charges in the p-type silicon, producing an electric field $\mathbf{E}_{dep}$ from the n-type silicon to the p-type silicon. This field prevents further diffusion.
- The region of this field is known as the *depletion region*.
- When $V_{anode} > V_{cathode}$, i.e. an electric field $\mathbf{E}$ that opposes $\mathbf{E}_{dep}$, the field $\mathbf{E}$ accelerates electrons and holes towards the depletion region, thus making it smaller.

  This causes a current flows through the diode from the anode to the cathode. The diode is in forward biased.
- When $V_{anode} < V_{cathode}$, i.e. an electric field $\mathbf{E}$ that is parallel to $\mathbf{E}_{dep}$. This increases the size of depletion region. The diode is reverse biased, no current flows.

3.4.2 Metal Oxide Semiconductor Field Effect Transistors (MOSFETs)

3.4.3 n-channel MOSFETs

- Formed of a section of p-type silicon substrate with a source and drain section of n-type silicon.



- The gate, consisting of two metal plate (or polycrystalline silicon) and a dielectric (a capacitor) is placed on top of the substrate, used to control the transistor.
- **Operation**:
  - (**Off**) $V_G \leq V_{Substrate}$: the p-n junctions (diodes) between the source or drain and the substrate are in reverse biased. So $I_{D \to S} = 0$.
  - (**On**) $V_G > V_{Substrate}$:
    - Electric field **E** forms, attracting negative charge to bottom plate of gate.
    - If the potential difference $V_{G \to Sub}$ is sufficiently large, the negative charge *inverts* the region (the *channel*) from p-type to n-type.
    - Electrons now flow from S to D. So $I_{D \to S} > 0$.
- Substrate is tied to GND, so $V_{Substrate} = 0$.
- nMOS transistors pass 0's well.

3.4.4 p-channel MOSFETs

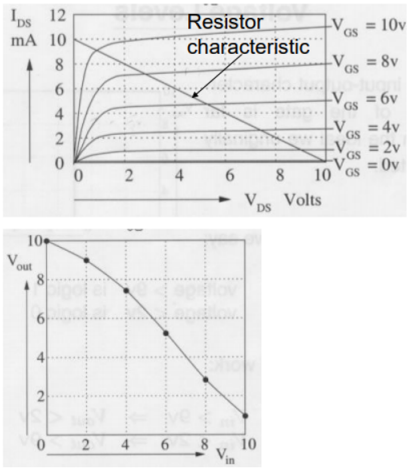
- Similar to n-channel MOSFETS, except the n and p-type silicon is reversed.
- Substrate is tied to V_{DD}.
- pMOS transistors pass 1's well.
- **Operation**:
  - (**Off**) $V_G \geq V_{Substrate}$
  - (**On**) $V_G < V_{Substrate}$

3.5 nMOS Logic

3.5.1 n-channel MOSFET Characteristics

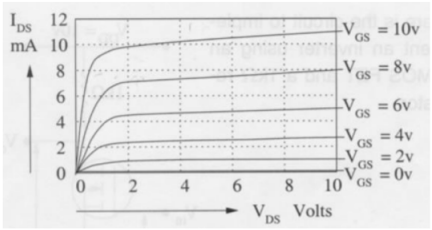
Voltage Characteristic

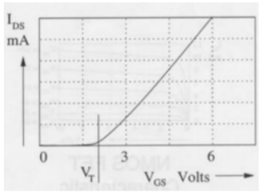
- Plot $I_{D\to S}V_{D\to S}$ characteristics for n-channel MOSFETs for various gate voltages V_G.
- Plot resistor IV characteristic in potential divider.
- Plot relationship of V_G against $V_{D\to S}$ (Or V_{out} if $V_S = 0$).



Current Characteristic

- Plot $I_{D\to S}V_{D\to S}$ characteristics for n-channel MOSFETs for various gate voltages V_G.
- Plot resistor IV characteristic in potential divider.
- Draw vertical line in IV characteristic for constant $V_{D\to S}$. Plot voltage V_G against current $I_{D\to S}$.



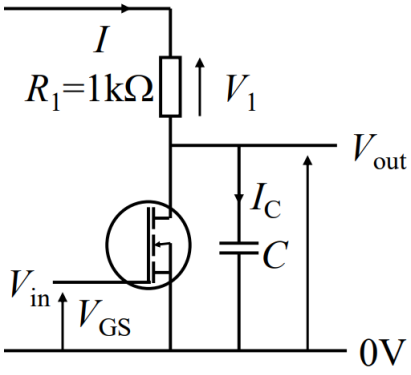


- Threshold voltage V_T.

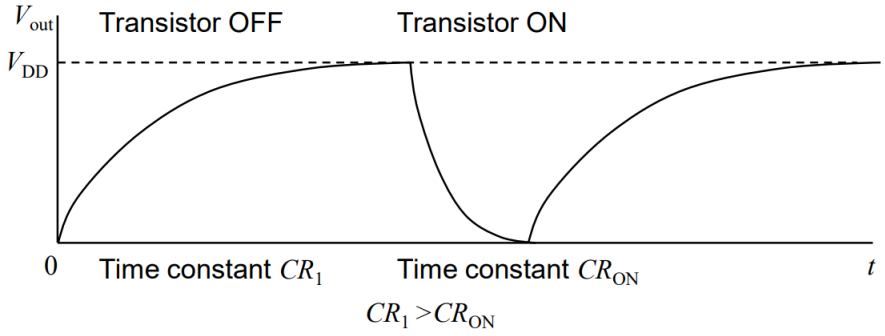
3.5.2 nMOS Inverter

Issues

- nMOS Inverter is not ideal.
- Use voltage thresholds for binary characteristic.
- Speed :
  - Due to parallel wires in circuits, we have parasitic capacitance. This is a finite capacitance C to ground.



  - When nMOS is **on**, it has a low resistance R_{On}. The capacitor C discharges through R_{On}.
  - When nMOS is **off**, it charges through R_D.
  - Since $R_D \gg R_{On}$, the time constant $\tau_{charge} \gg \tau_{discharge}$.



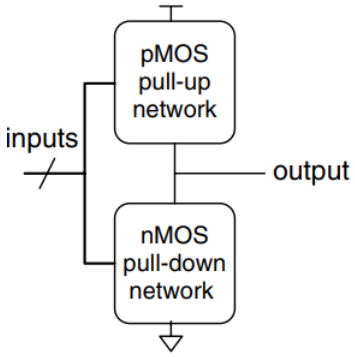
  - Limits switching speed of transistor.

- Power :
  - When nMOS is **off**, no current $\implies V_R = 0$.
  - When nMOS is **on**, current $I_{D\to S}$. So power dissipated is $P = I_{D\to S}V_R$.

3.6 CMOS Logic

- Complementary MOS (CMOS) logic overcomes the problems of nMOS logic.

- General Structure:



- The networks are either in series or parallel configurations.
- If both networks are on, a short circuit would exist between V_{DD} and GND.
- If both networks are off, the output would *float*.
- This is prevented using the rule of *conduction components*, that states that if pMOS network is in parallel, nMOS must be in series (and vice versa).
- Power is still dissipated when switching

$$P = \frac{1}{2} C V_{DD}^2 f.$$