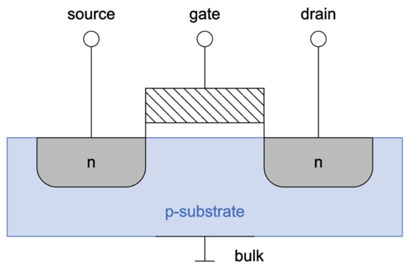
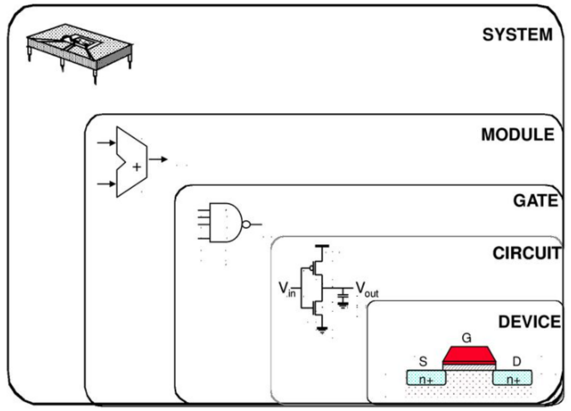
Lecture 2: FPGA fabric, memories and high-level synthesis

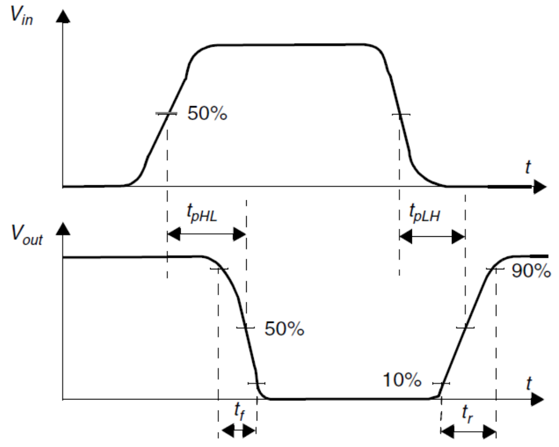
Emad Samuel Malki Ebeid

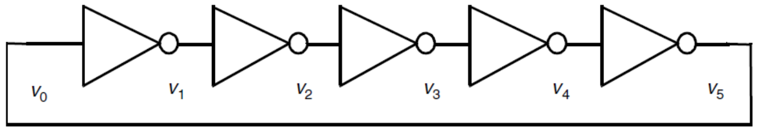
Summary of lecture 1



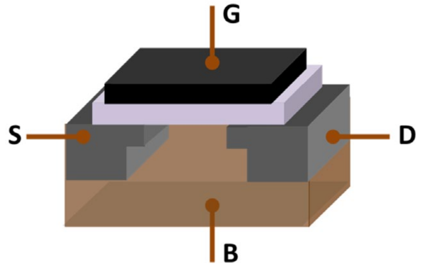
- Transistor revolution
- IC issues
  - Power dissipation
  - Power distribution
  - Clock distribution
  - Noise
  - etc.





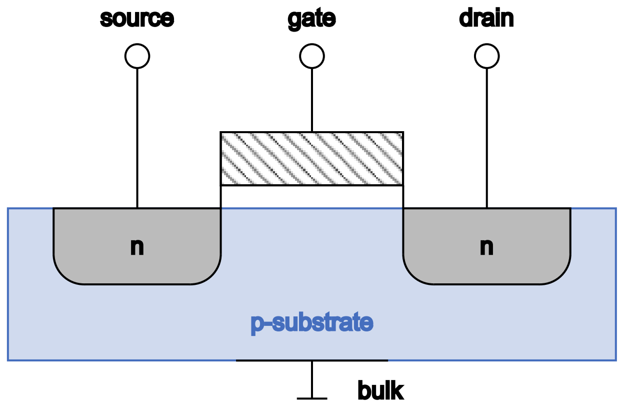


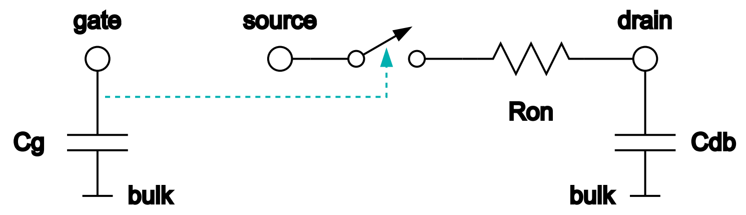
$$Clock\ Frequency = \frac{1}{2 * t_p * N}$$

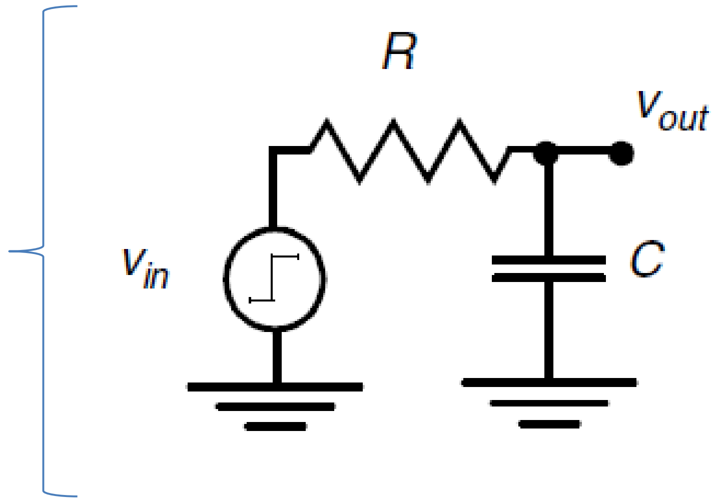


Propagation delay of the first-order *RC* networks

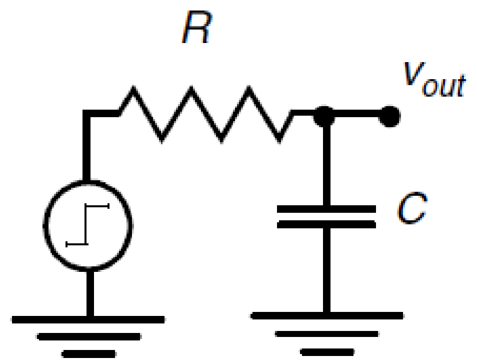
- The RC step response is a fundamental behavior of all digital circuits and thus digital circuits are often modeled as first-order *RC* networks.
- The propagation delay of the circuit is of considerable interest.







Propagation delay



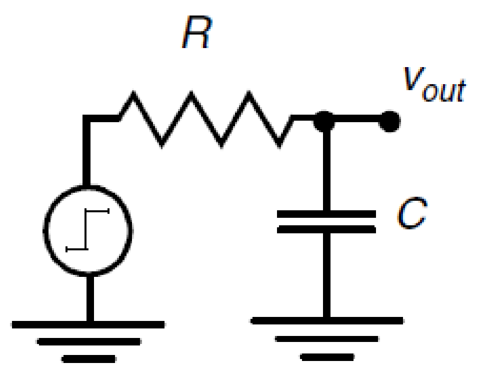
First order RC network

- $V_{in} = R * i(t) + v_{out}(t)$
- $v_{out}(t) = v_c(t) = \frac{1}{C} \int_0^t \boldsymbol{i(t)} dt$
- Hence, $V_{in} = R * \boldsymbol{i(t)} + \frac{1}{C} \int_0^t i(t) dt$

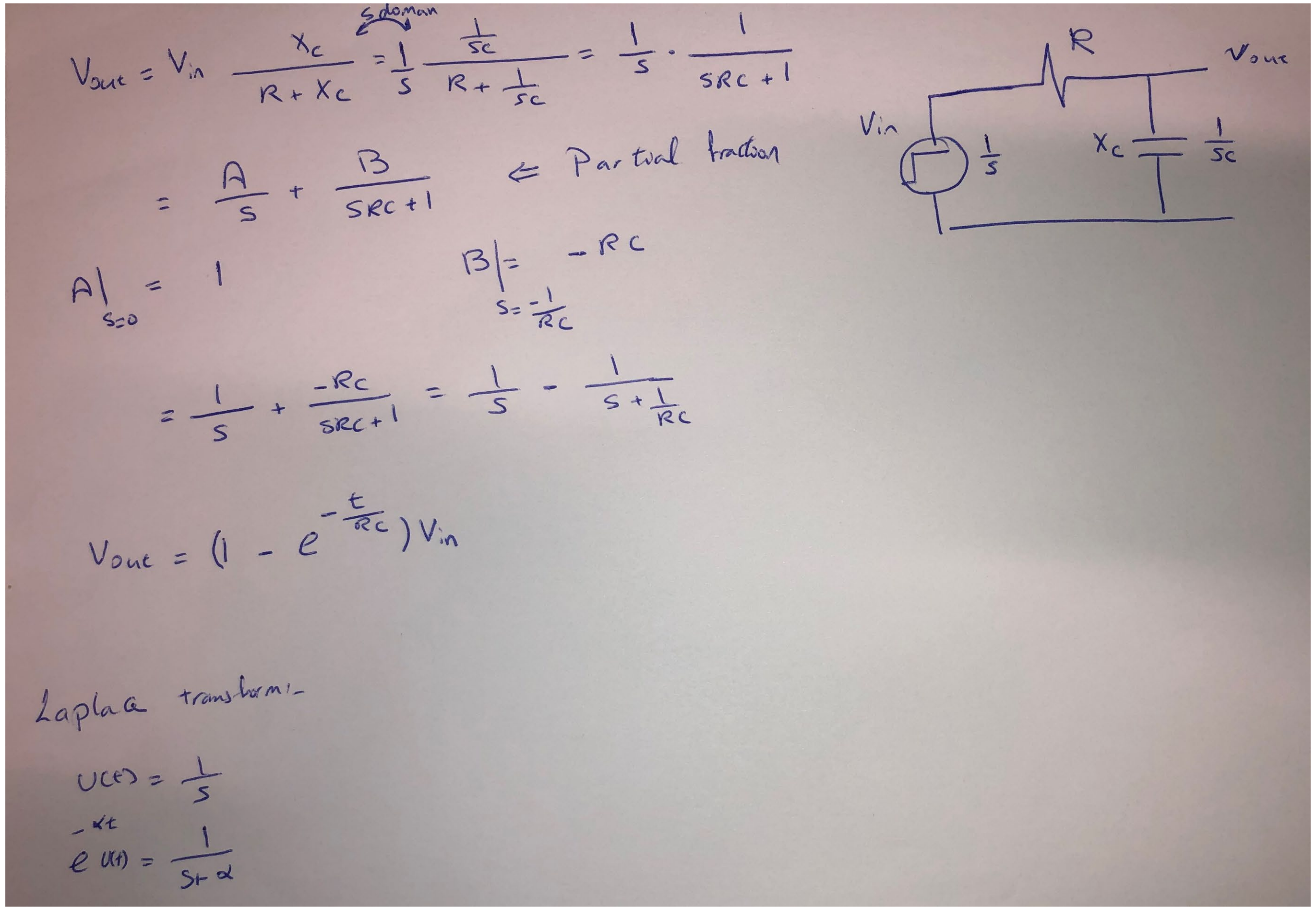
To calculate i(t), take the derivative

- $0 = R \frac{di(t)}{dt} + \frac{1}{C}\ i(t) \Rightarrow \frac{di(t)}{dt} + \frac{1}{RC}\ i(t) = 0$ (solve the diff. Equations)
- $i(t) = I_0 e^{-\frac{t}{RC}}\ \&\, when\ t = 0, I_0 = \frac{V_{in}}{R}$
- $v_{out}(t) = \frac{1}{C} \int_0^t \frac{V_{in}}{R} e^{-\frac{t}{RC}} dt = V_{in}\left(1 - e^{-\frac{t}{RC}}\right) = V_{in}(1 - e^{-\frac{t}{\tau}})$ $\quad where\ \tau = RC\ (time\ constant)$

Propagation delay



First order RC network (another solution)



s domain

$$V_{out} = V_{in} \frac{X_c}{R + X_c} = \frac{1}{s} \frac{\frac{1}{sC}}{R + \frac{1}{sC}} = \frac{1}{s} \cdot \frac{1}{sRC + 1}$$

$$= \frac{A}{s} + \frac{B}{sRC + 1} \Leftarrow \text{Partial fraction}$$

$$A\Big|_{s=0} = 1 \qquad B\Big|_{s=-\frac{1}{RC}} = -RC$$

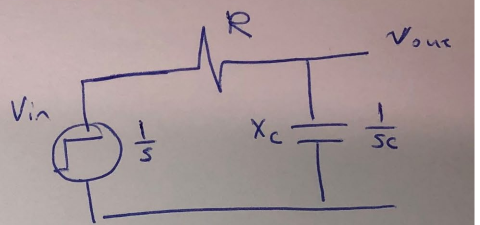
$$= \frac{1}{s} + \frac{-RC}{sRC + 1} = \frac{1}{s} - \frac{1}{s + \frac{1}{RC}}$$

$$V_{out} = \left(1 - e^{-\frac{t}{RC}}\right) V_{in}$$

Laplace transform:-

$$u(t) = \frac{1}{s}$$

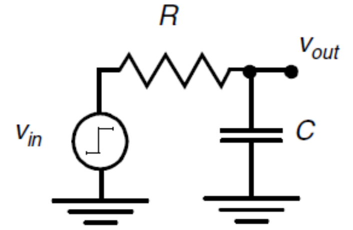
$$e^{-\alpha t} u(t) = \frac{1}{s + \alpha}$$



Power and Energy Consumption

- It determines how much energy is consumed per operation and heat the circuit dissipates.
- Design decisions depends on:
  - power-supply capacity
  - battery lifetime
  - supply-line
  - sizing
  - packaging
  - cooling

Power and Energy Consumption



- Dissipation measures:

$$P_{peak} = i_{peak}V_{supply} = \max[p(t)]$$

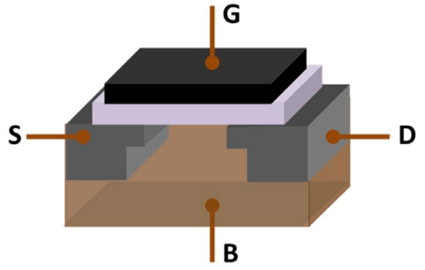
$$P_{av} = \frac{1}{T}\int_0^T p(t)dt = \frac{V_{supply}}{T}\int_0^T i_{supply}(t)dt$$

Where:

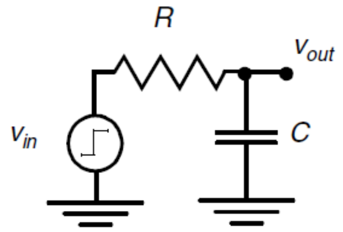
- p(t) is the instantaneous power
- i_{supply} is current being drawn from the supply voltage V_{supply}

- *higher the number of switching events, the higher the dynamic power consumption.*

PDP (power-delay product): energy consumed by the gate *per switching event*



Energy Dissipation of First-Order *RC* Network



- Total energy delivered by the source

$$E_{in} = \int_0^\infty i_{in}(t)v_{in}(t)dt = V\int_0^\infty C\frac{dv_{out}}{dt}dt = (CV)\int_0^V dv_{out} = CV^2$$

- Energy stored in the capacitor

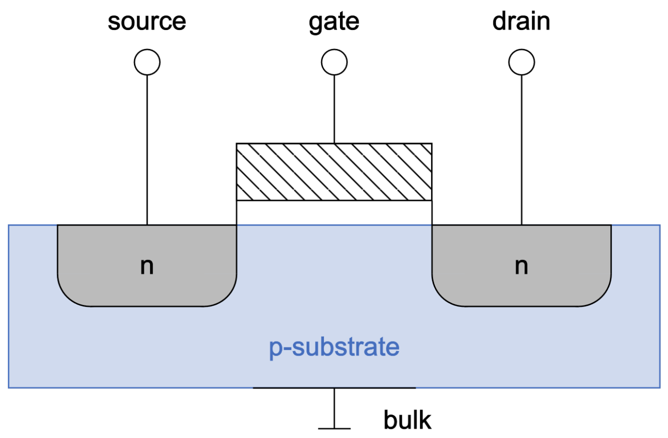
$$E_C = \int_0^\infty i_C(t)v_{out}(t)dt = \int_0^\infty C\frac{dv_{out}}{dt}v_{out}dt = C\int_0^V v_{out}dv_{out} = \frac{CV^2}{2}$$

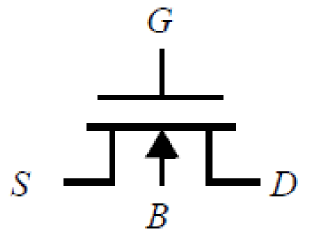
- Where is the other half?

MOSFETs

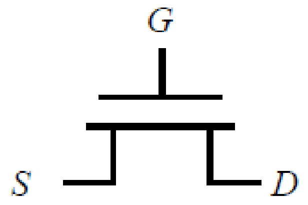
MOSFET

- The metal-oxide-semiconductor field-effect transistor (MOSFET or MOS, for short).
- Its major asset from a digital perspective is that the device performs very well as a **switch**

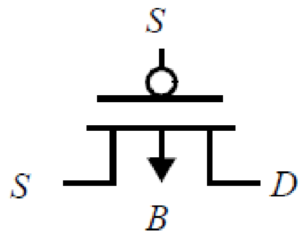




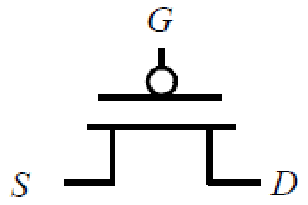
(a) NMOS transistor as 4-terminal device



(b) NMOS transistor as 3-terminal device



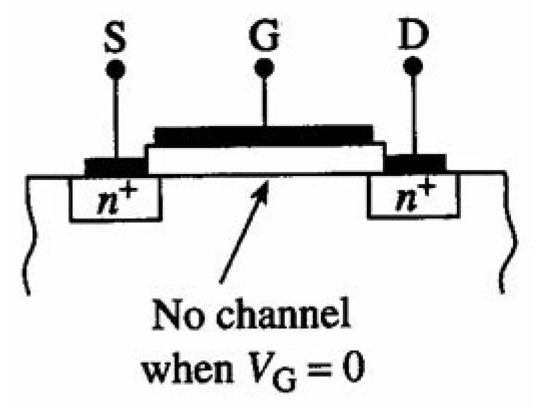
(a) PMOS transistor as 4-terminal device

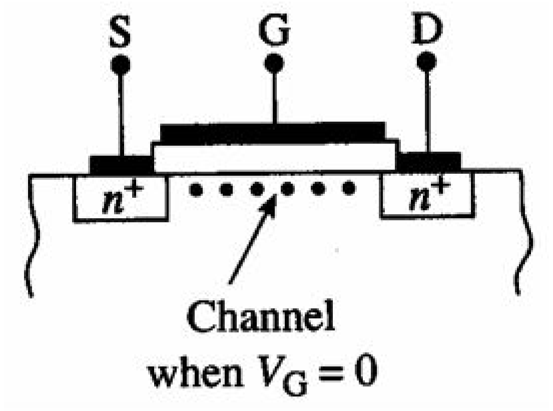


(d) PMOS transistor as 3-terminal device

Types

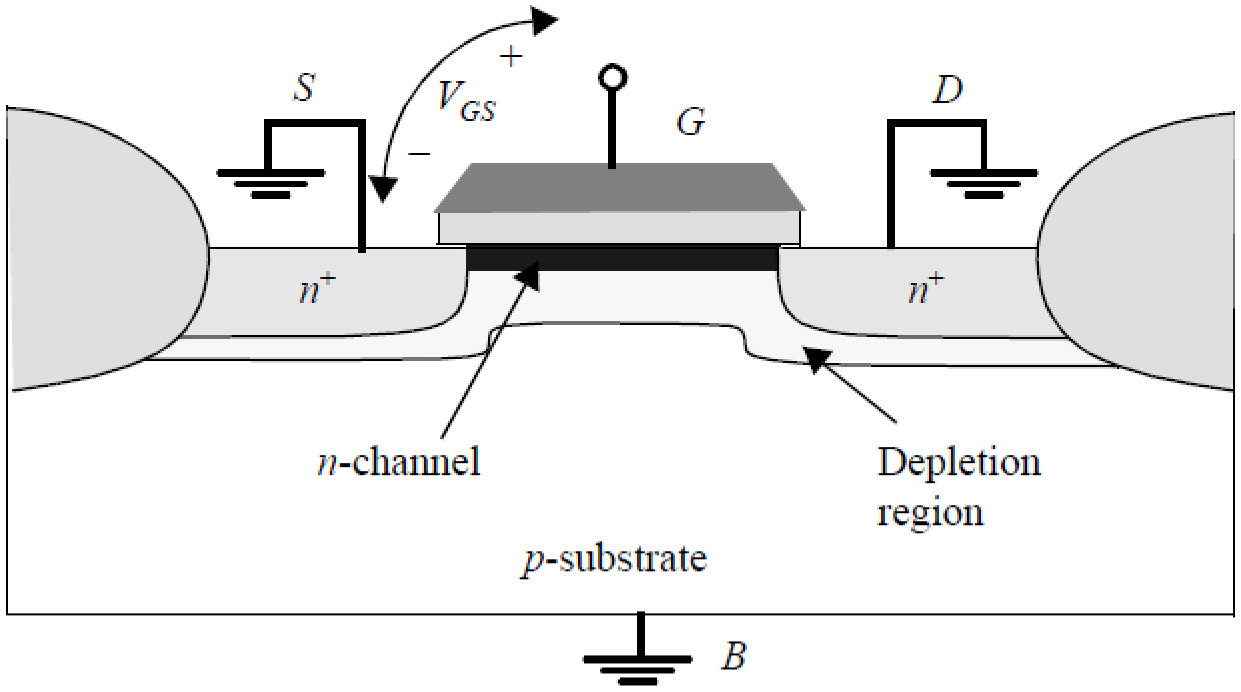
- N-MOS & P-MOS
  - Enhancement-mode MOSFET is off at zero gate–source voltage (no channel).
  - Depletion-mode MOSFET, is normally *ON* at zero gate–source voltage (channel).

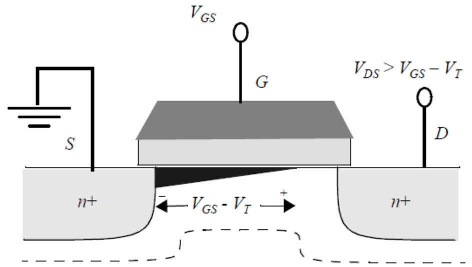




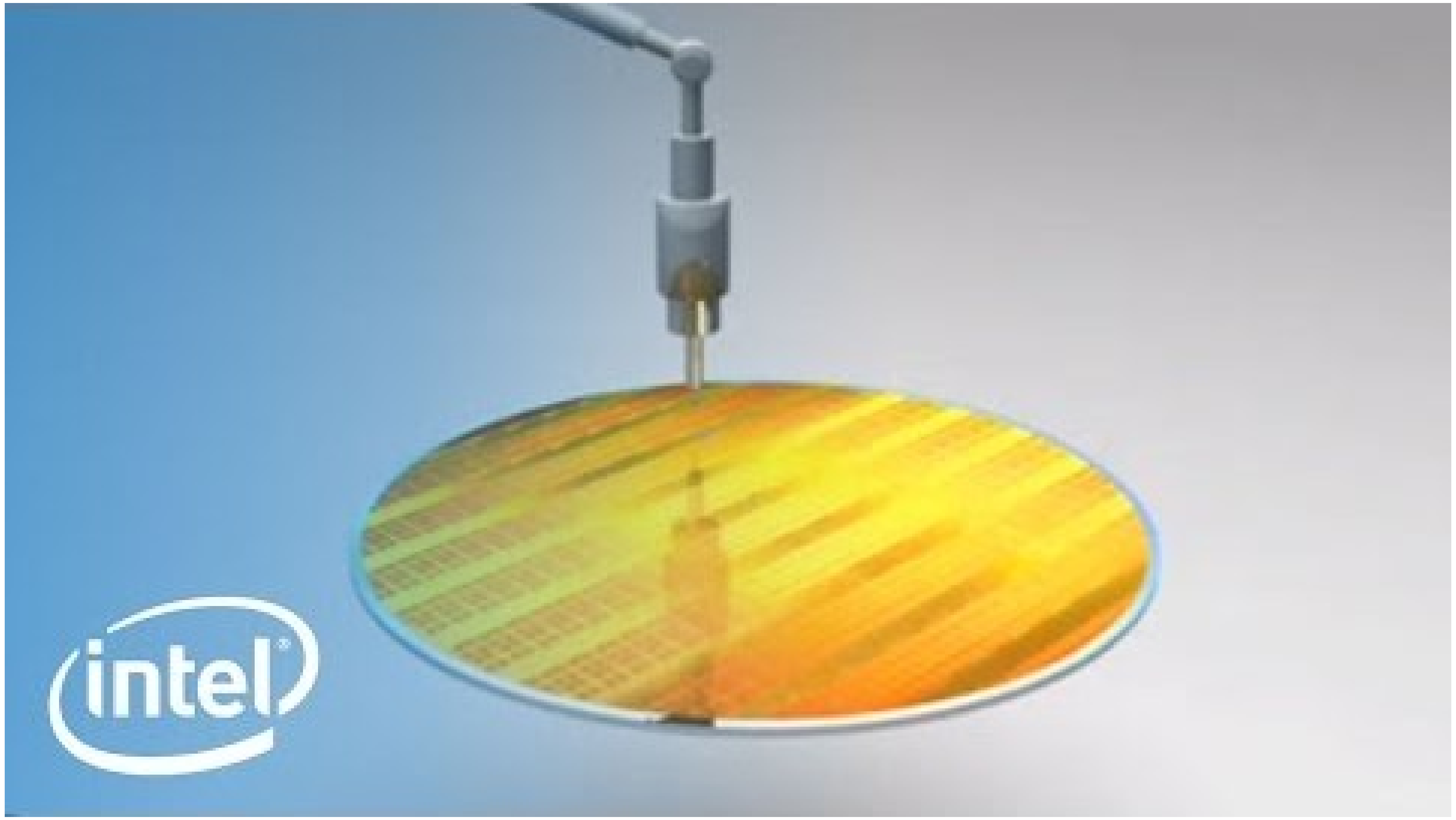
The Threshold Voltage

- Cut off region
  - $V_{GS} = 0$
- Linear region (ohmic region)
  - $V_{GS} > V_T$
  - $V_{DS} < V_{GS} - V_T$
- Saturation region
  - $V_{GS} > V_T$
  - $V_{DS} > V_{GS} - V_T$





Semiconductor manufacturing process



https://www.youtube.com/watch?v=Q5paWn7bFg4

Semiconductor manufacturing process

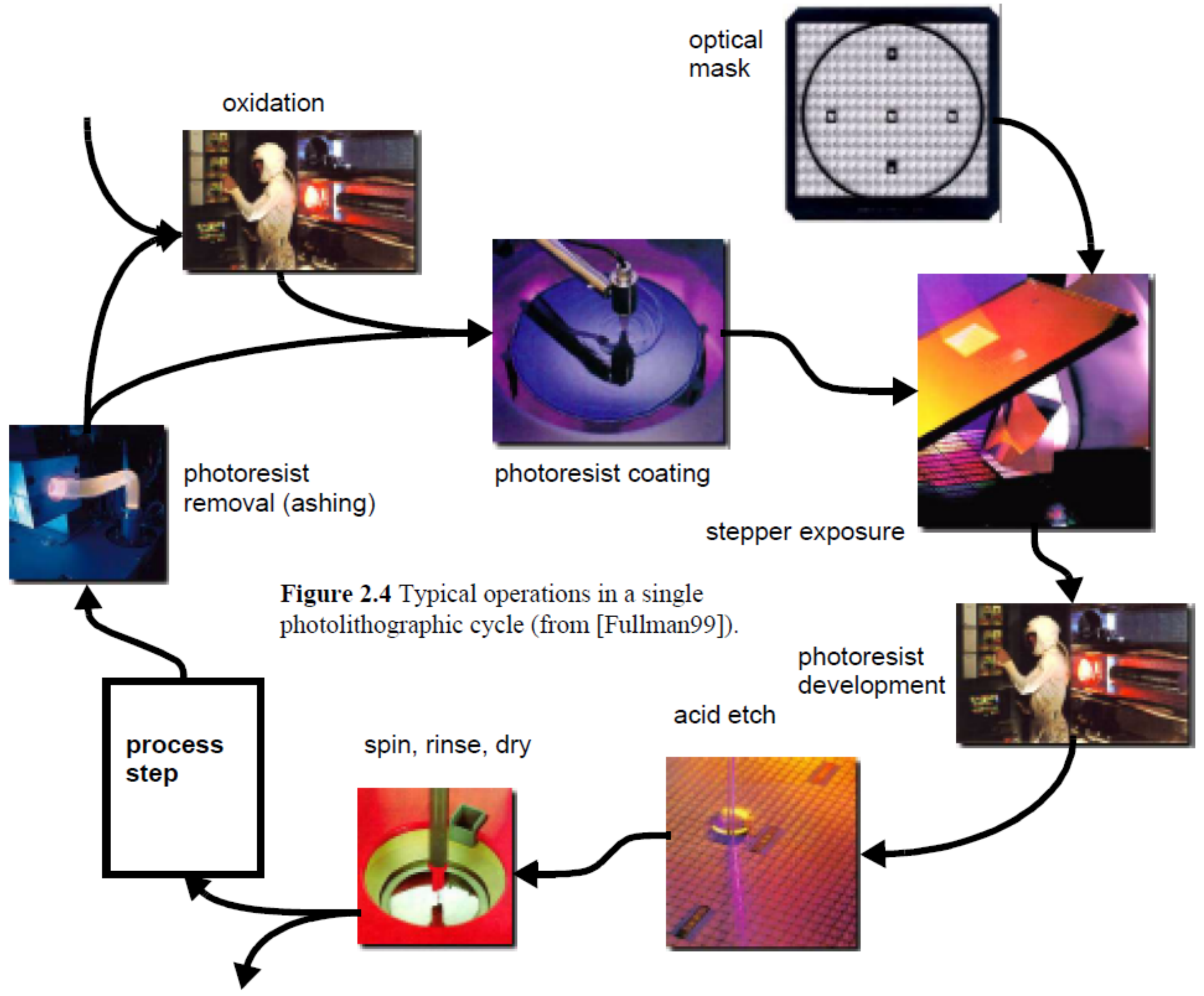
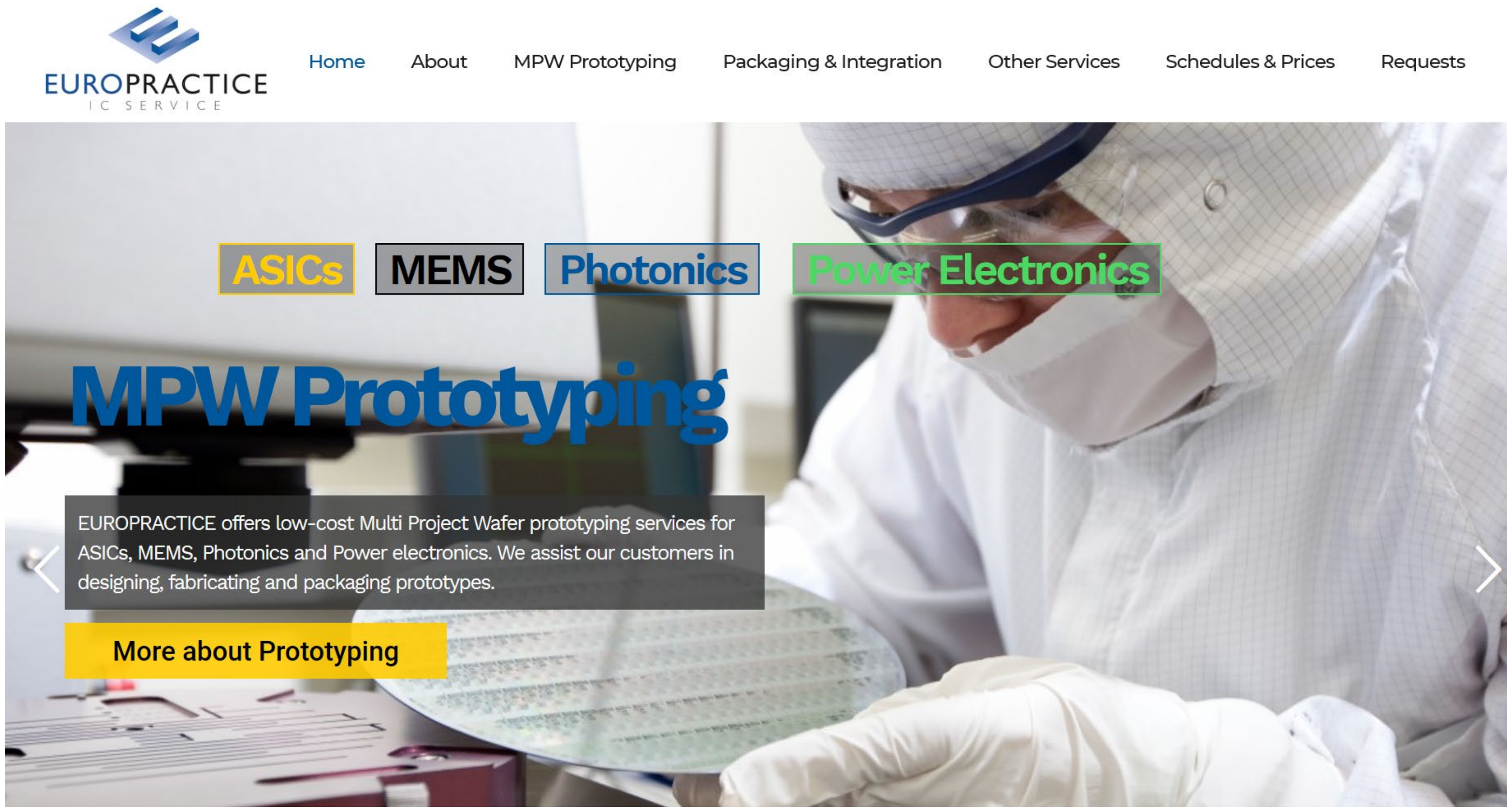


Figure 2.4 Typical operations in a single photolithographic cycle (from [Fullman99]).

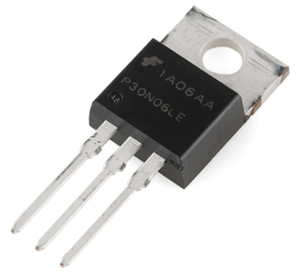
Build ”your” IC

- http://europractice-ic.com/ IC services for research institutes

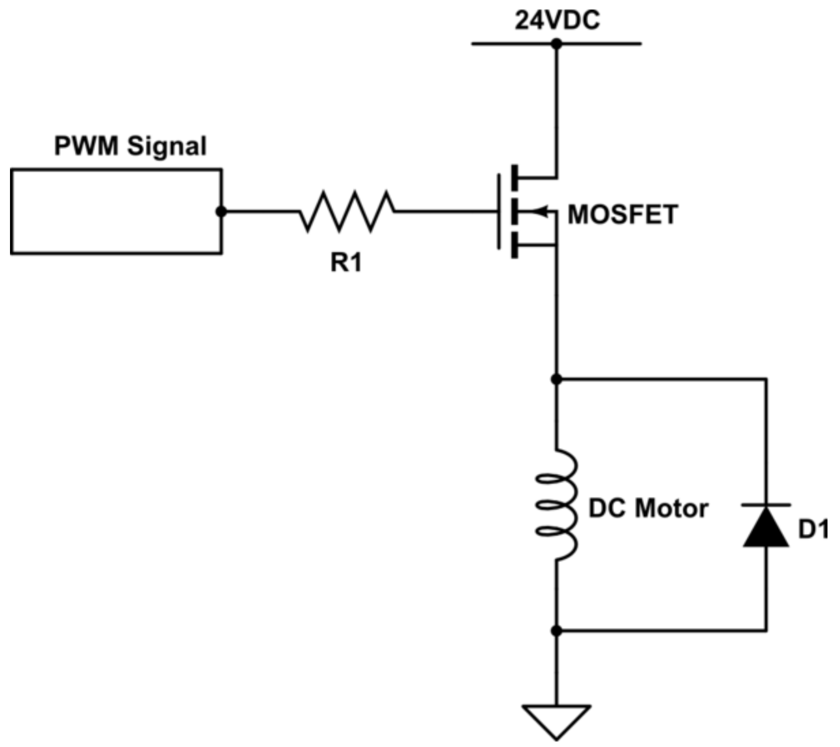


EUROPRACTICE offers low-cost Multi Project Wafer prototyping services for ASICs, MEMS, Photonics and Power electronics. We assist our customers in designing, fabricating and packaging prototypes.

Applications

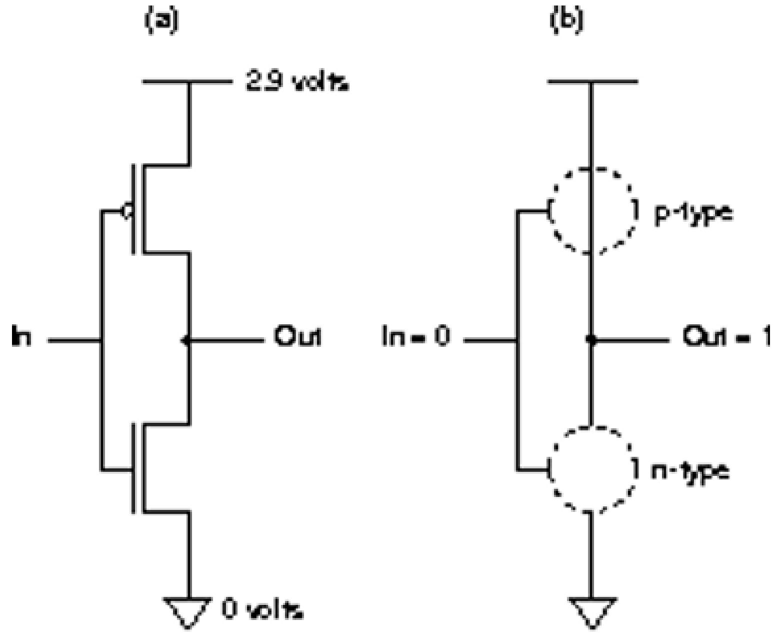


- DC motor drive
- DC relay
- Inverter
- Amplifier
- passive element
  - resistor, capacitor and inductor
- digital circuit



Logic gates

- Inverter



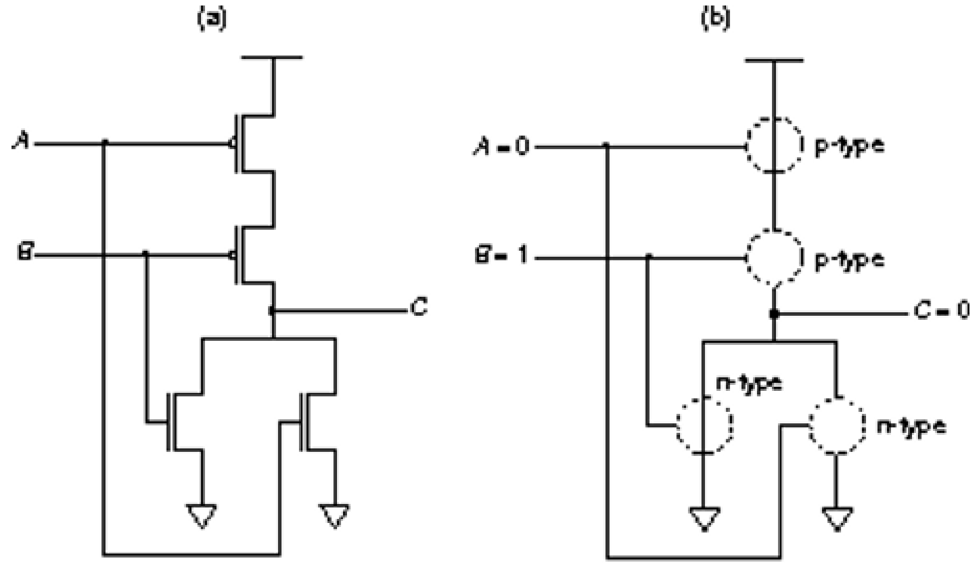
(c)

In	Out
0 volts	2.9 volts
2.9 volts	0 volts

(d)

In	Out
0	1
1	0

- NOR



(c)

A	B	C
0 volts	0 volts	2.9 volts
0 volts	2.9 volts	0 volts
2.9 volts	0 volts	0 volts
2.9 volts	2.9 volts	0 volts

(d)

A	B	C
0	0	1
0	1	0
1	0	0
1	1	0

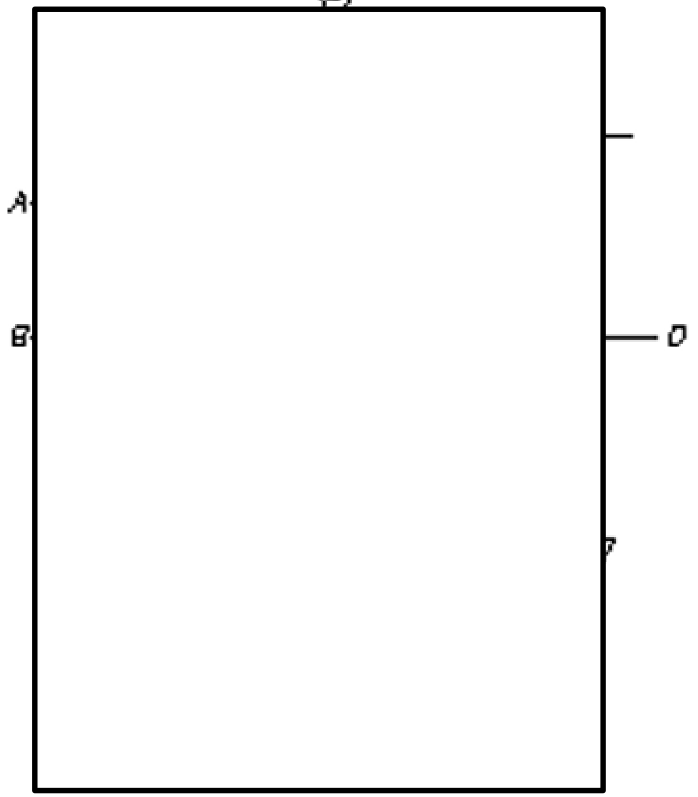
Logic gates

- AND

(10 minutes)

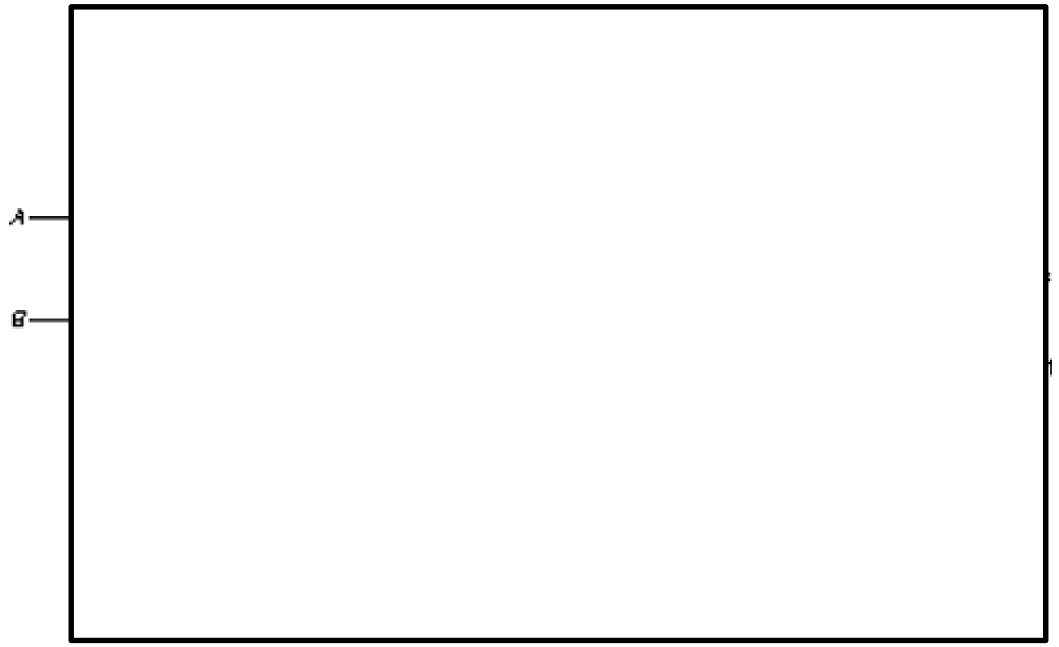
- OR

(a)



(b)

A	B	C	D
0	0	1	0
0	1	1	0
1	0	1	0
1	1	0	1

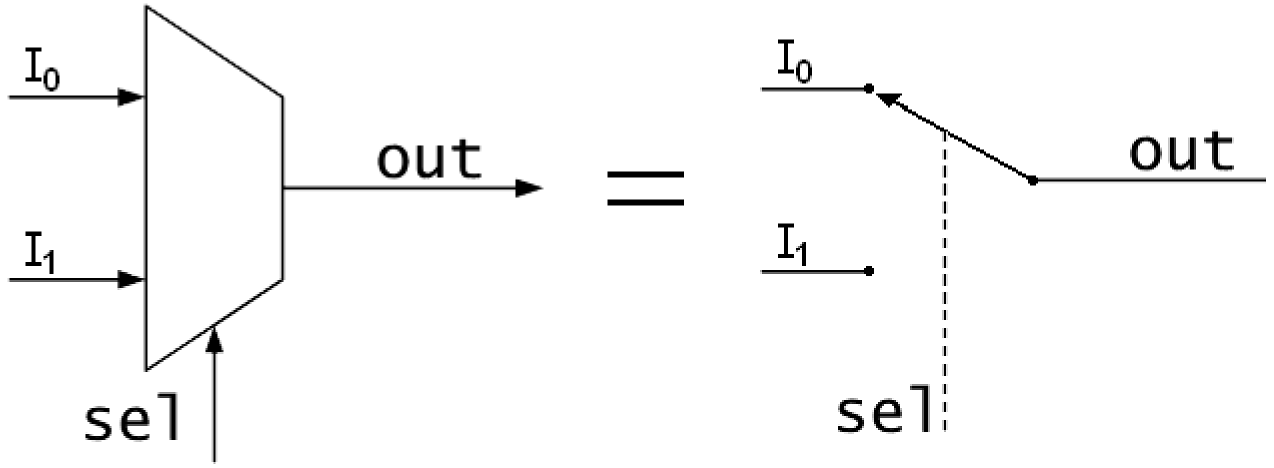


(c)

A	B	C	D
0	0	1	0
0	1	0	1
1	0	0	1
1	1	0	1

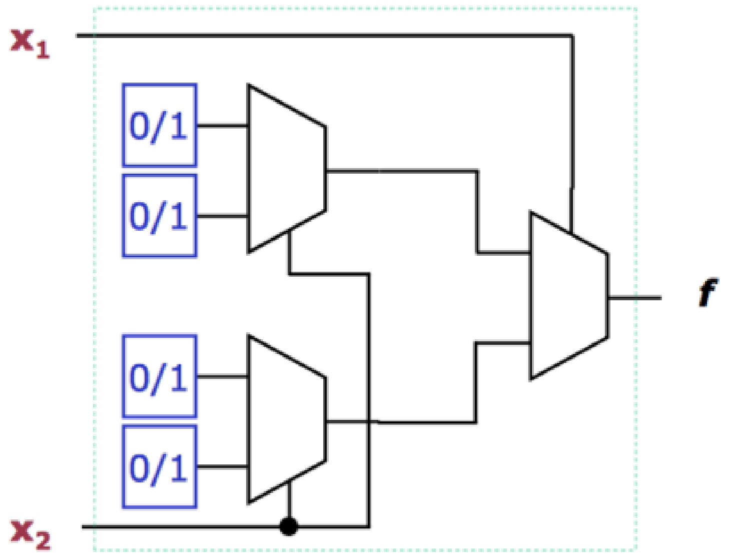
Multiplixer

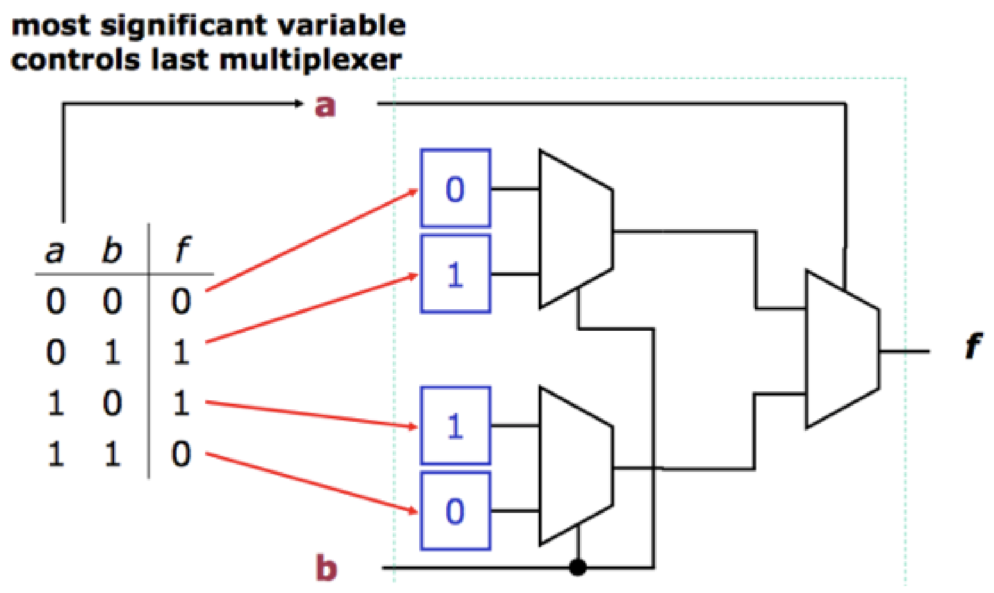
- It function is to select one of the inputs and connect it to the output



Draw the logic diagram
(5 minutes)

Look-up tables (LUT)

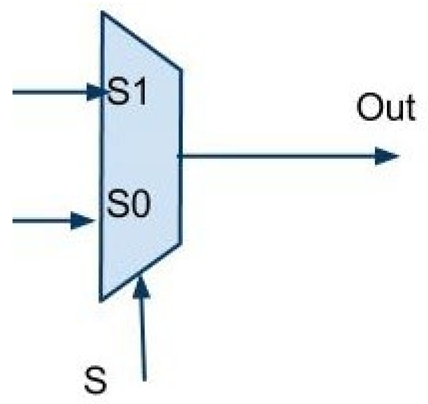




a	b	f
0	0	0
0	1	1
1	0	1
1	1	0

LUT of logic gates

- AND
- OR
- Invert
- XOR

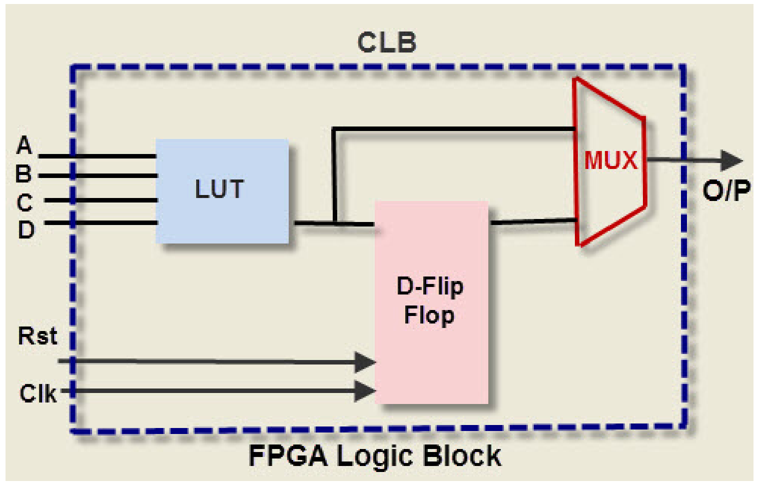


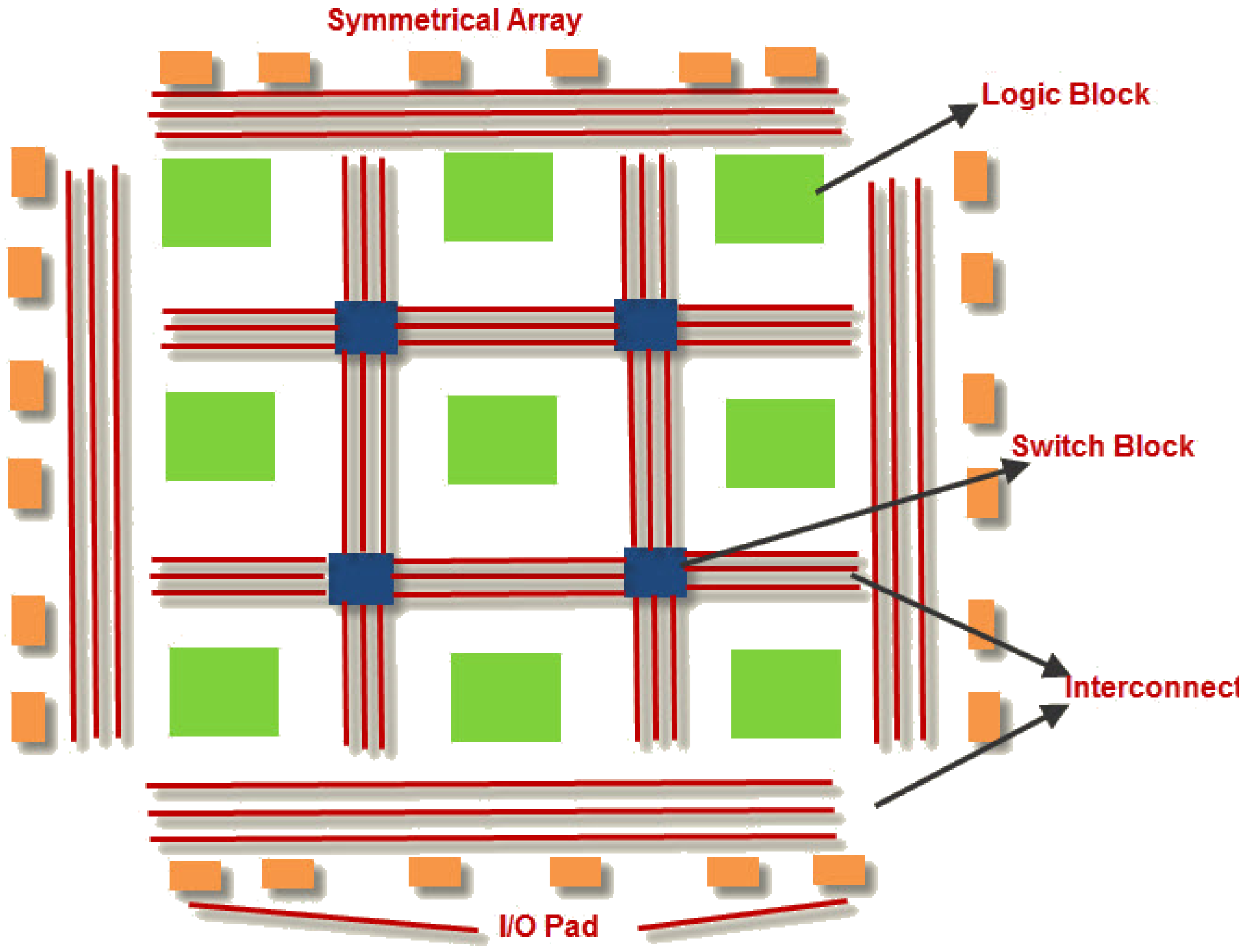
FPGA

- Field Programmable Gate Array (FPGA) is an integrated circuit containing gate matrix which can be programmed by the user "in the field" without using expensive equipment.
- An FPGA contains a set of programmable logic gates and rich interconnect resources, making it possible to implement complex digital circuits.
- FPGA devices are produced by a number of semiconductor companies:
  - Xilinx, Altera, Actel, Lattice, QuickLogic and Atmel.
- Configuration bitstream can be stored in FPGA using various technologies.
  - The majority of FPGAs is based on SRAM (Static RAM).

FPGA

- Field Programmable Gate Array
- The general FPGA architecture consists of three types of modules. They are I/O blocks or Pads, Switch Matrix/ Interconnection Wires and Configurable logic blocks (CLB).



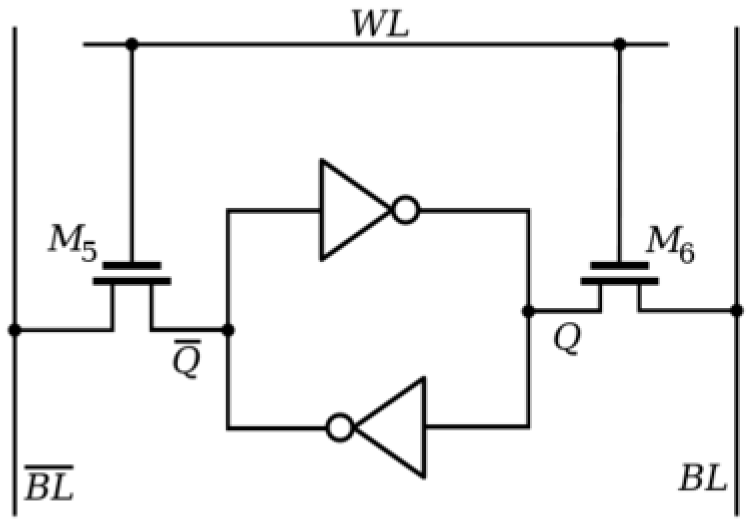


Memory cells: RAM

RAM is a **volatile** memory that the data is eventually lost when the memory is not powered

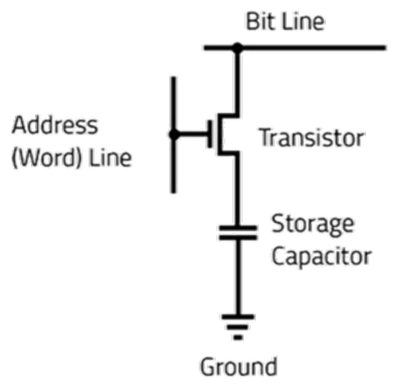
SRAM: Static random-access memory

- Speed ↑ size (bigger)
- Cache memory
- Low power consumption



DRAM: Dynamic random-access memory

- Speed ↓ size (smaller)
- Main memory
- Charge leakage
- High power consumption
- Refreshed frequently (few milliseconds)

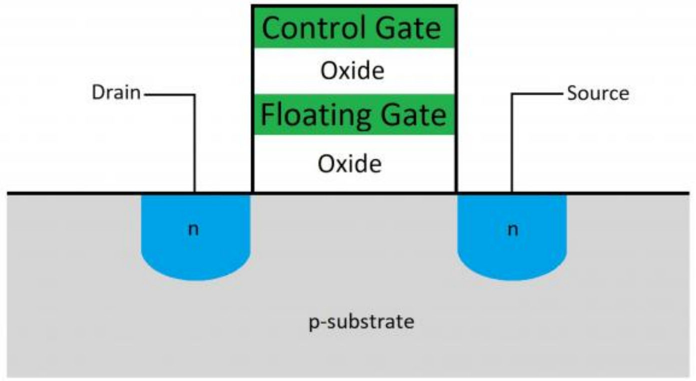


Single Memory Cell

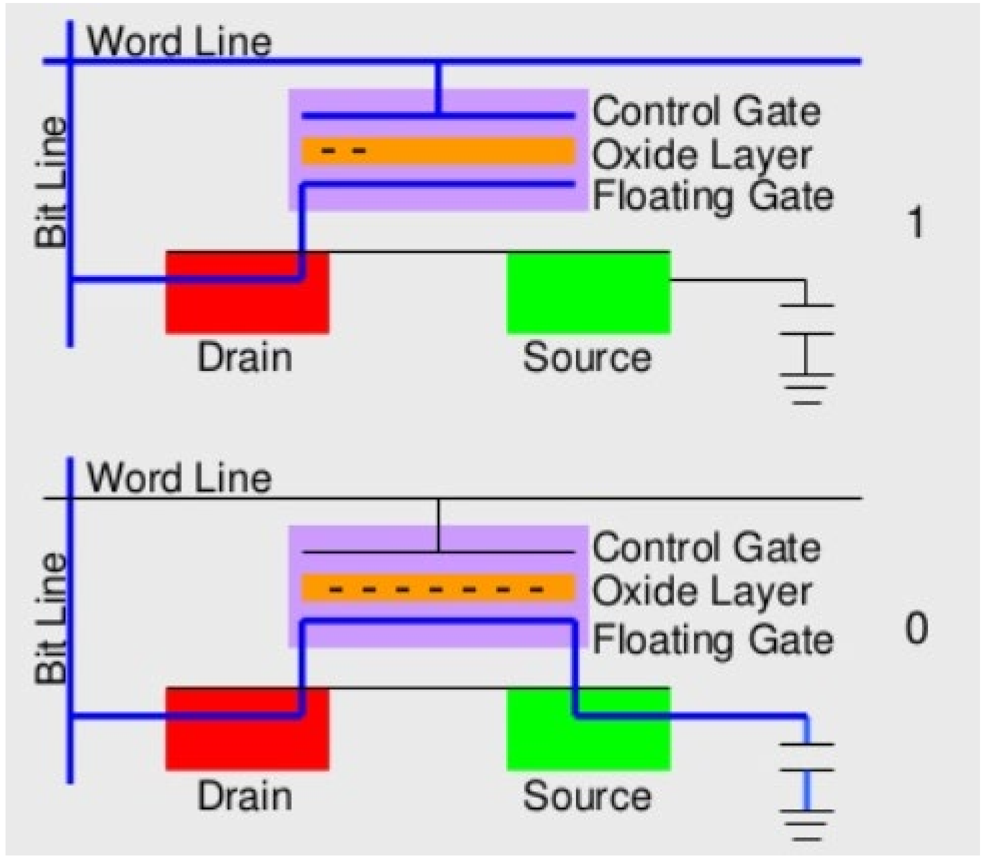
SRAM-based FPGAs

- It stores logic cells configuration data in the static memory
- Since SRAM is volatile, the FPGAs must be programmed (configured) upon start
- There are two basic modes of programming:
  - **Master mode**, when FPGA reads configuration data from an external source, such as an external Flash memory chip.
  - **Slave mode**, when FPGA is configured by an external master device, such as a processor (via a dedicated configuration interface or via a boundary-scan (JTAG) interface).
- SRAM-based FPGAs with an internal flash memory:
  - it contains internal flash memory blocks, thus eliminating the need to have an external non-volatile memory. It uses flash only during startup to load data to the SRAM configuration cells.

Programmable switch FPGAs

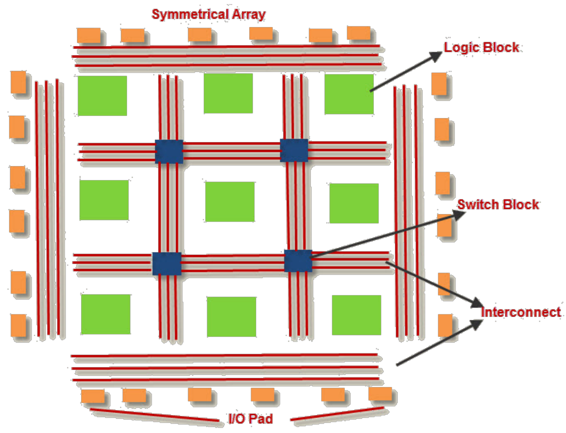


- It uses flash as a primary resource for configuration storage, and doesn't require SRAM
- Switch is a floating-gate transistor that can be turned off by injecting charge onto its floating gate
- It has a limit number of reprogramming
- More secure than SRAM-FPGA



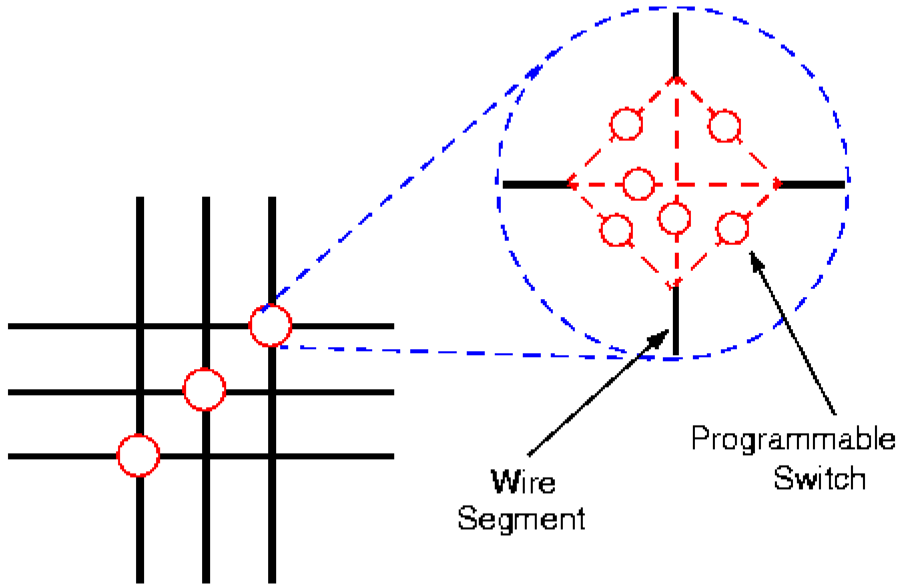
Fuse-based FPGA

- One time program (OTP) FPGA
- The fuse makes or breaks link between two wires
- More secure than SRAM (no load from an external device)
- In high radiation environments (space or nuclear reactors), radiation events can cause SRAM, that contains the program, to change state but not in fused FPGA.

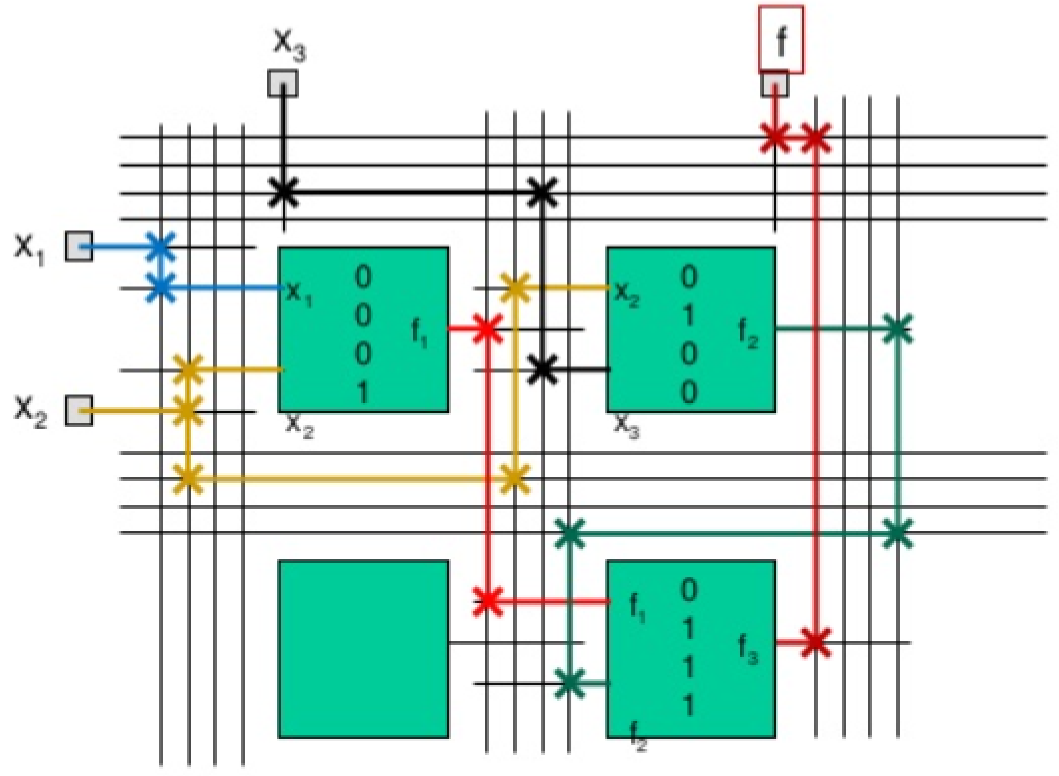


Programmable switch matrix

- All internal connections are composed of metal segments with programmable switching points and switching matrices to implement the desired routing.



Example: FPGA programming

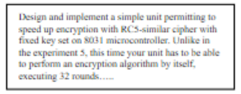


$$f_1 = x_1 x_2$$
$$f_2 = \overline{x_2} x_3$$
$$f = x_1 x_2 + \overline{x_2} x_3$$

Other FPGA Building Blocks

- Clock distribution: element clock skew
- Embedded memory blocks
- Special purpose blocks:
  - DSP blocks: Hardware multipliers, adders and registers
  - Embedded microprocessors/microcontrollers
  - High-speed serial transceivers

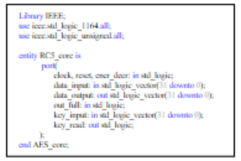
Design flow


Design and implement a simple unit permitting to speed up encryption with RC5-similar cipher with fixed key set on 8031 microcontroller. Unlike in the experiment 5, this time your unit has to be able to perform an encryption algorithm by itself, executing 32 rounds…..

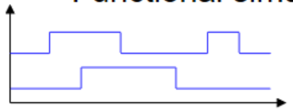
Specification / Pseudocode

On-paper hardware design

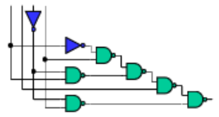
VHDL description (Your Source Files)


```
Library IEEE;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity RC5_core is
        port(
            clock, reset, encr_decr: in std_logic;
            data_input: in std_logic_vector(31 downto 0);
            data_output: out std_logic_vector(31 downto 0);
            out_full: in std_logic;
            key_input: in std_logic_vector(31 downto 0);
            key_read: out std_logic;
        );
end AES_core;
```

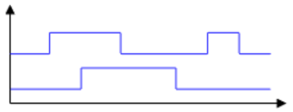
Functional simulation



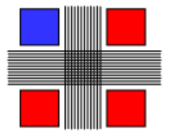
Synthesis



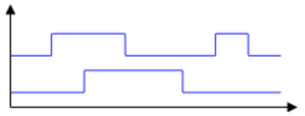
Post-synthesis simulation



Implementation

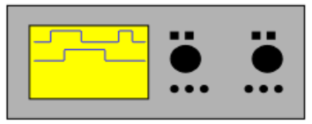


Timing simulation

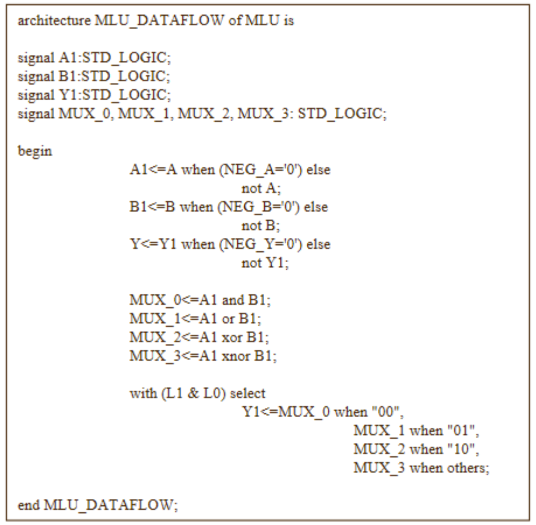


Configuration

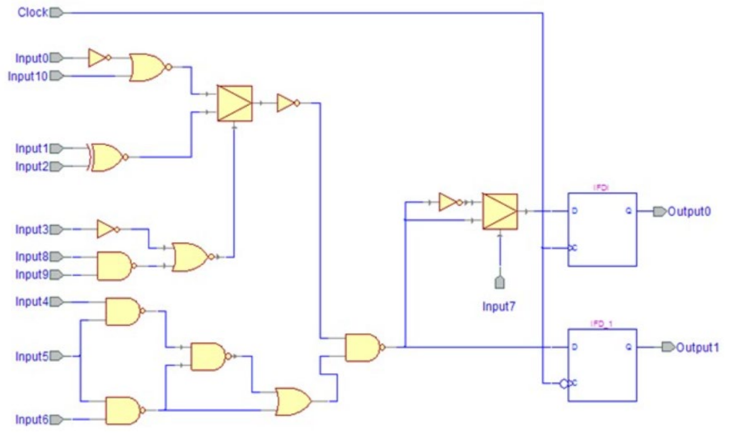
On chip testing



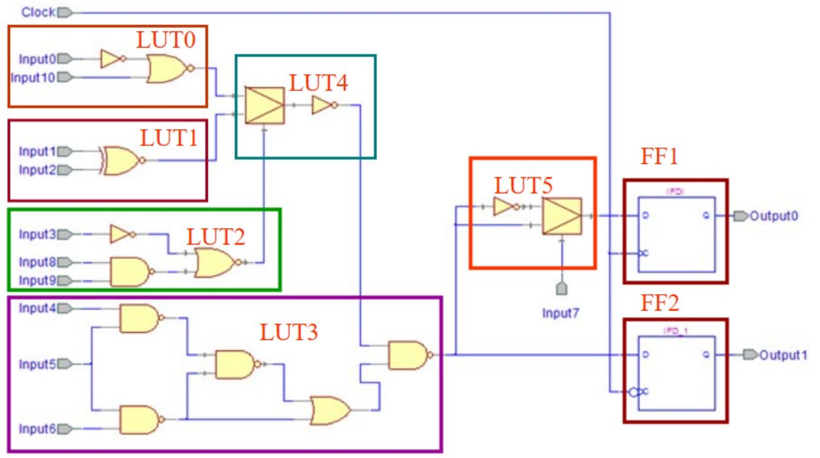
Synthesis



```
architecture MLU_DATAFLOW of MLU is

signal A1:STD_LOGIC;
signal B1:STD_LOGIC;
signal Y1:STD_LOGIC;
signal MUX_0, MUX_1, MUX_2, MUX_3: STD_LOGIC;

begin
                A1<=A when (NEG_A='0') else
                                not A;
                B1<=B when (NEG_B='0') else
                                not B;
                Y<=Y1 when (NEG_Y='0') else
                                not Y1;

                MUX_0<=A1 and B1;
                MUX_1<=A1 or B1;
                MUX_2<=A1 xor B1;
                MUX_3<=A1 xnor B1;

                with (L1 & L0) select
                                Y1<=MUX_0 when "00",
                                                MUX_1 when "01",
                                                MUX_2 when "10",
                                                MUX_3 when others;

end MLU_DATAFLOW;
```

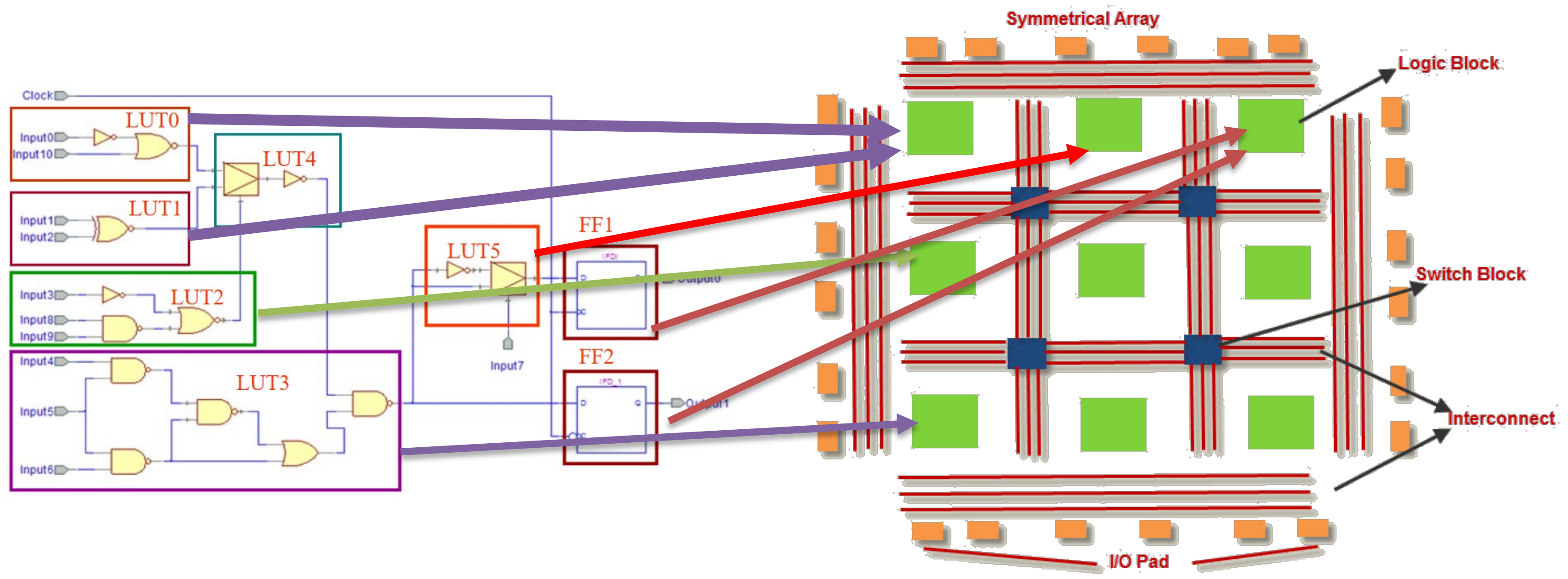


Circuit netlist



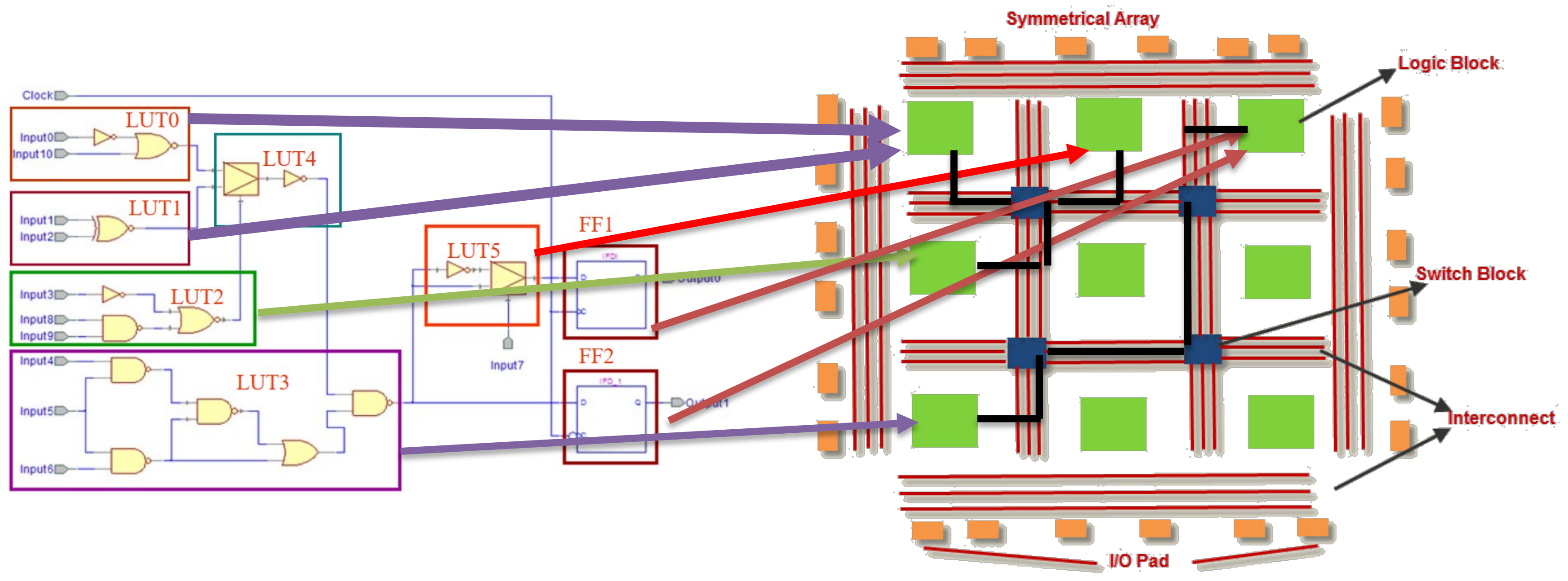
Mapping

Placing



Clock
LUT0
Input0
Input10
LUT4
LUT1
Input1
Input2
LUT2
Input3
Input8
Input9
LUT3
Input4
Input5
Input6
LUT5
Input7
FF1
FF2
Output0
Output1
Symmetrical Array
Logic Block
Switch Block
Interconnect
I/O Pad

Routing

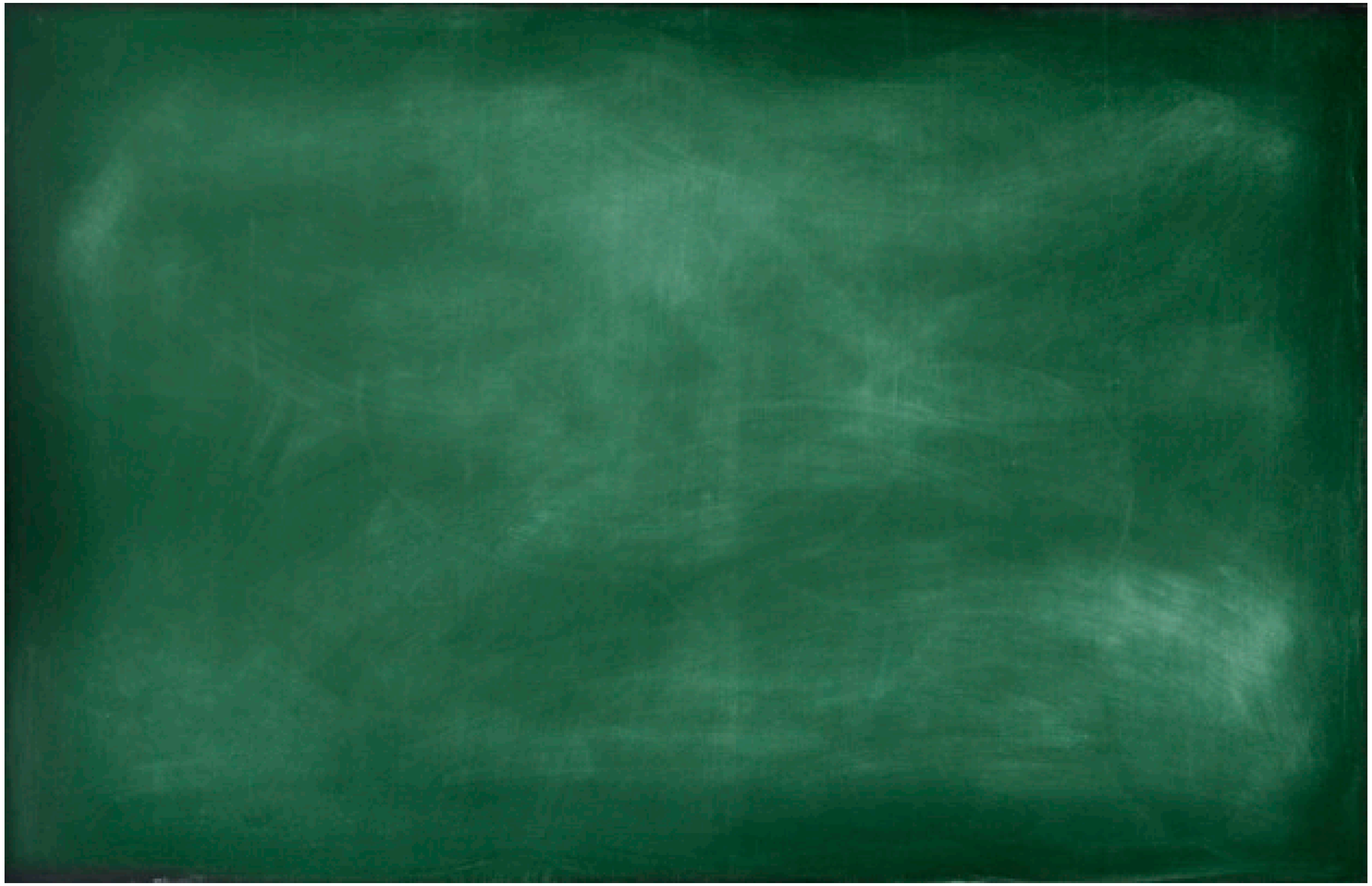


Clock
LUT0
Input0
Input10
LUT4
LUT1
Input1
Input2
LUT5
FF1
Output0
Input3
LUT2
Input8
Input9
Input7
Input4
LUT3
FF2
Input5
Output1
Input6
Symmetrical Array
Logic Block
Switch Block
Interconnect
I/O Pad

Configuration

- Once a design is implemented, the synthesis tool generates a file that the FPGA can understand
- This file is called a bit stream: a BIT file (.bit extension)
- The BIT file can be downloaded directly to the FPGA, or can be converted into a PROM file which stores the programming information

Look to the black board notes



Functional simulation

- Build the follows
  - Inverter
  - Dec 2-4
  - Mux 4-1
  - 4 bit Counter
- Build keypad interface
  - Block design
    - Connect buses
    - Connect keypad module