Programmable Logic Devices

Instructor:

Professor Jim Plusquellic

Text:

The Design Warrior's Guide to FPGAs, Devices, Tools and Flows, Clive "Max" Maxfield, ISBN: 0-7506-7604-3

Modeling, Synthesis and Rapid Prototyping with the Verilog HDL, Michael D. Ciletti, ISBN: 0-13-977398-3

Supplementary texts:

Advanced Digital Logic Design Using Verilog, State Machines and Synthesis for FPGAs, Sunggu Lee, ISBN: 0-534-55161-0

Web: http://www.cs.umbc.edu/~plusquel/415

Course Description

This course covers:

- Programmable device technologies and architectures
- Hardware description languages: Verilog
- Design flows
- Finite state machines (FSM)
- System design considerations
- FPGA hardware

Prerequisites:

CMPE 212: Digital Logic Design

CMPE 310: Systems Design

Familiarity with Test & Measurement equipment

Familiarity with FSM design

PLDs, ASICs and FPGAs

FPGA definition:

Digital integrated circuit that contains *configurable* blocks of logic and *configurable* interconnects between these blocks.

Key points:

Manufacturer does NOT determine functionality, rather it is the designer who defines it after the device is fabricated via programming.

FPGAs can be configured at least once, many are reprogrammable.

PLDs vs. ASICs

PLDs (programmable logic devices): subdivided into SPLDs (simple or just PLDs) and CPLDs (complex).

ASICs (application specific integrated circuits): are devices whose functionality is hardwired, and are not reprogrammable.

PLDs, ASICs and FPGAs

PLDs vs. ASICs

A PLD's internal architecture is predetermined by the manufacturer but is designed so that it can be configured by engineers in the field.

PLDs have a limited number of logic gates (in comparison to FPGAs) and can implement only simpler logic functions.

ASICs offer the ultimate in size, number of transistors, complexity and performance.

However, they are extremely time-consuming and expensive to design.

Plus, the design is *frozen* in silicon, requiring a new version if changes are needed.

FPGAs occupy the *middle ground* between PLDs and ASICs.

The are programmable but contain millions of logic gates, allowing large and complex functions to be implemented.

PLDs, ASICs and FPGAs

FPGAs vs ASICs

The cost of an FPGA design is *much lower* than that of an ASIC (given the ASIC is not produced in large numbers and cannot amortize this cost).

Changing the design is also *much easier* with an FPGA, and the time-to-market much shorter (than an ASIC).

Functions of FPGAs today.

Prototype ASIC designs or verify the physical implementation of a new algorithms.

Some include embedded microprocessor cores, high-speed I/O interfaces, and other features.

Are used to implement just about anything, including communications devices and software-defined radios; radar, image and other DSP applications; up to system-on-chip (SoC) components.

PLDs, ASICs and FPGAs

FPGAs are eating into 4 major market segments:

- ASIC and custom silicon
- Digital signal processing (DSP)
- Embedded microcontrollers
- Physical layer communications

And have created a new market themselves

- Reconfigurable computing

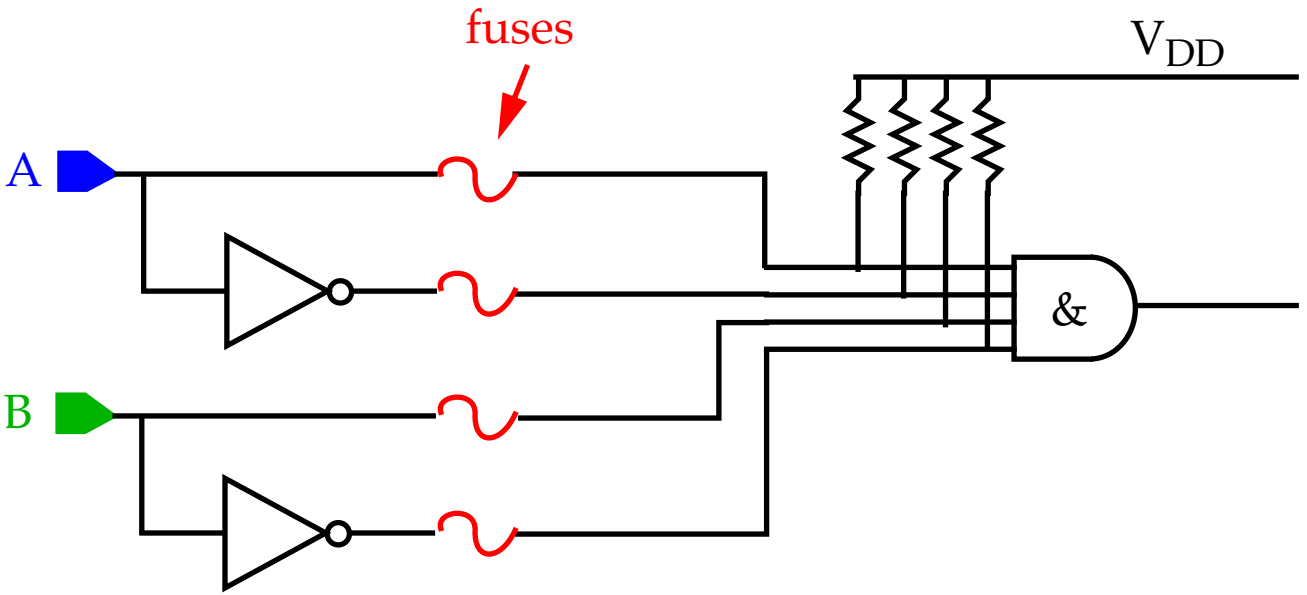
  This refers to exploiting the parallelism and reconfigurability of FPGAs to *hardware accelerate* software algorithms.

  Applications here span hardware simulation to cryptography.

Fusible Link Technologies

How are PLDs and FPGAs programmed?

One of the first techniques, **fusible-link technology**:



All of the fuses are initially intact (after manufacturing).

Design engineers selectively remove undesired fuses by applying pulses of relatively high voltage and current to the device's inputs.

These devices are *one-time* programmable, and not used by FPGAs.

Antifuse Technologies

Here, the unprogrammed device has links which are very high in resistance.

The compliment of *fusible link* technology.

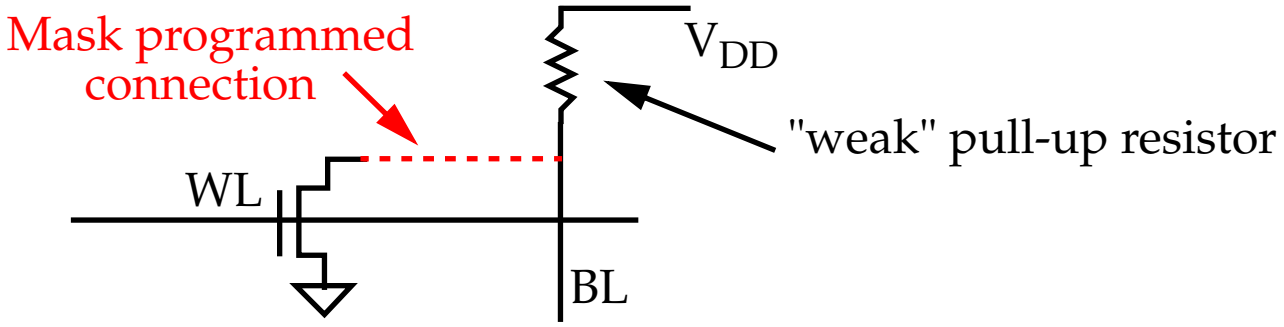
Connections are selectively *grown* by applying pulses of relatively high voltage and current to the device's inputs.

Converts *highly resistive* amorphous silicon to *conducting* polysilicon.

Mask-programmed devices

Basic ROMs (non-volatile read-only memory as opposed to RAM which are read-write and volatile) are mask-programmable.

Data in ROMs is hard-coded, using photo-masks that define the metalization structure (and connectivity)



Mask-Programmed Devices

ROM can be *preconstucted* and used to satisfy multiple customers.

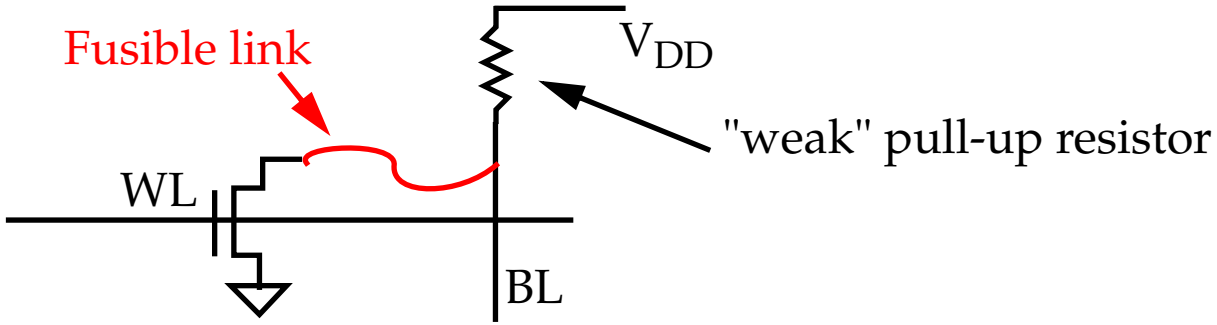
Customization per customer involves changing a *single* photo-mask that define which connections are present and which are absent.

PROMs

The problem with mask-programmable devices:

- They are expensive (unless produced in very large quantities)
- Are of limited use in development environments, where contents may need to be modified.

Programmable read only memory (PROM) was invented to address these shortcomings.



PROMs

Similar to *fusible link* technology, all fusible links are in place after manufacture.

Placing a logic 1 on the word line would cause all column bit lines to pull down to logic 0.

Designer *blows* fuses where he wants logic 1s to be output.

These devices were originally intended for use as memories, to store constant programs and data.

However, they were also used to implement logic functions, such as *lookup tables* and *state machines*.

They were cheap and were used to fix bugs and test new implementations.

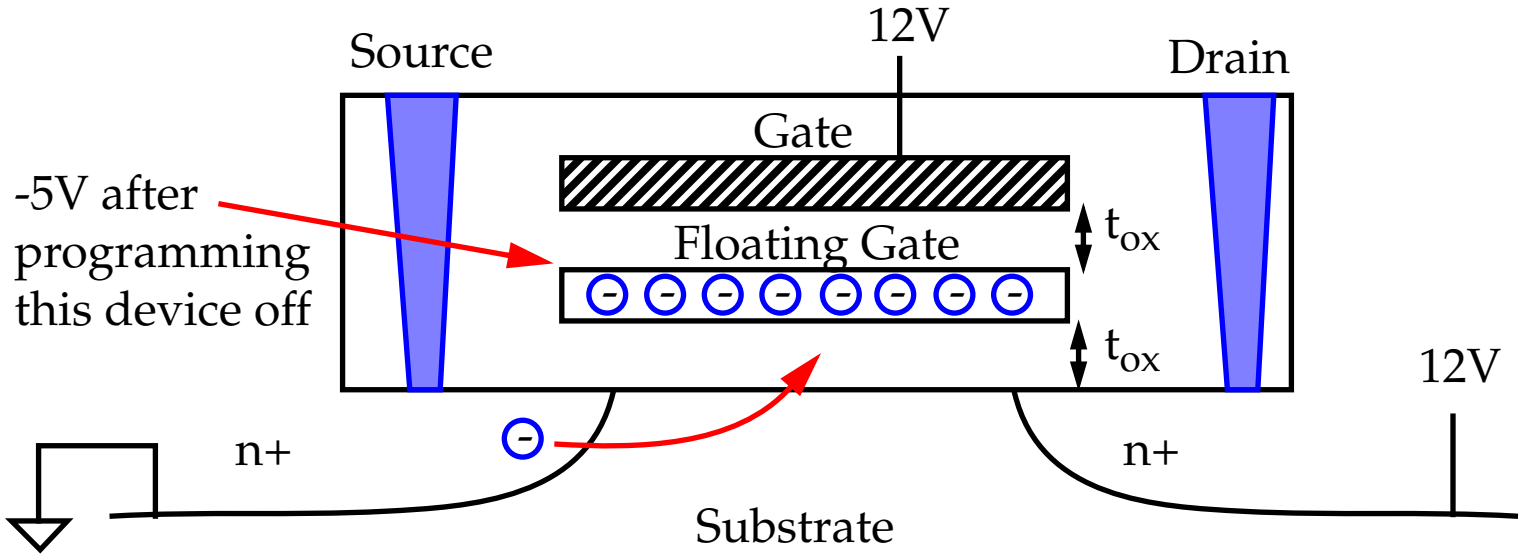
Simply burn a new device and plug it in.

Other general-purpose PLDs became available over time (more on this later).

EPROM-based Technologies

Devices based on fusible link or antifuse could only be programmed a single time.

Erasable programmable read-only memory (EPROM) was introduced by Intel in '71.



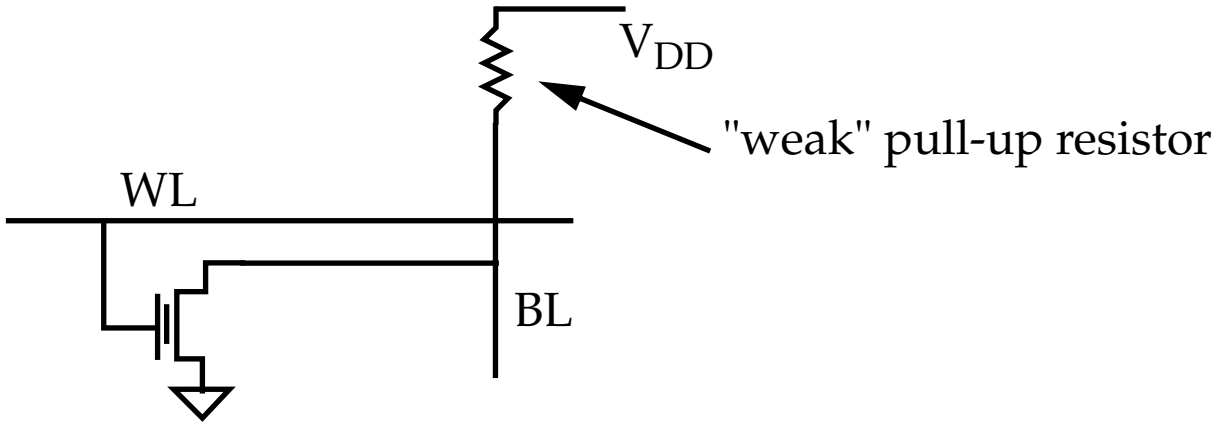
Accomplished by introducing a *floating gate* to the basic structure of a MOS transistor.

When unprogrammed, the floating gate has no effect.

EPROM-based Technologies

Programming involves placing a high voltage between gate and drain.

Hot electrons accumulate on the floating gate, disabling the transistor.



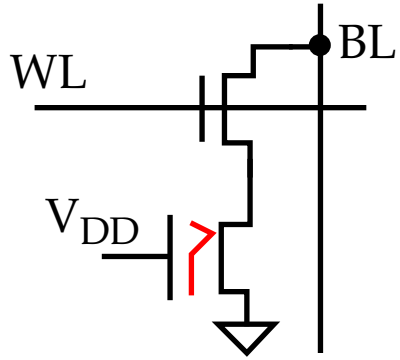
Besides being much smaller than fusible links, EPROM cells can be reprogrammed.

This is accomplished by discharging the electrons from the floating gate using a *UV* light source.

Disadvantages: Unprogramming takes about 20 minutes and packages were relatively expensive (contained a quartz window).

EEPROM-based Technologies

Electrically erasable programmable read-only memories (EEPROMS) came next.



Same basic architecture but now each cell requires two transistors.

The upper access transistor was needed because it was not possible to electrically remove the electrons from the fl oatinggate in a precise manner.

Removing too many left the gate positively charged and *always on*.

Flash memory solved this problem by introducing monitoring circuits.

"Flash" originally coined to refl ect rapid erasure times compared to EPROM.

SRAM-based Technologies

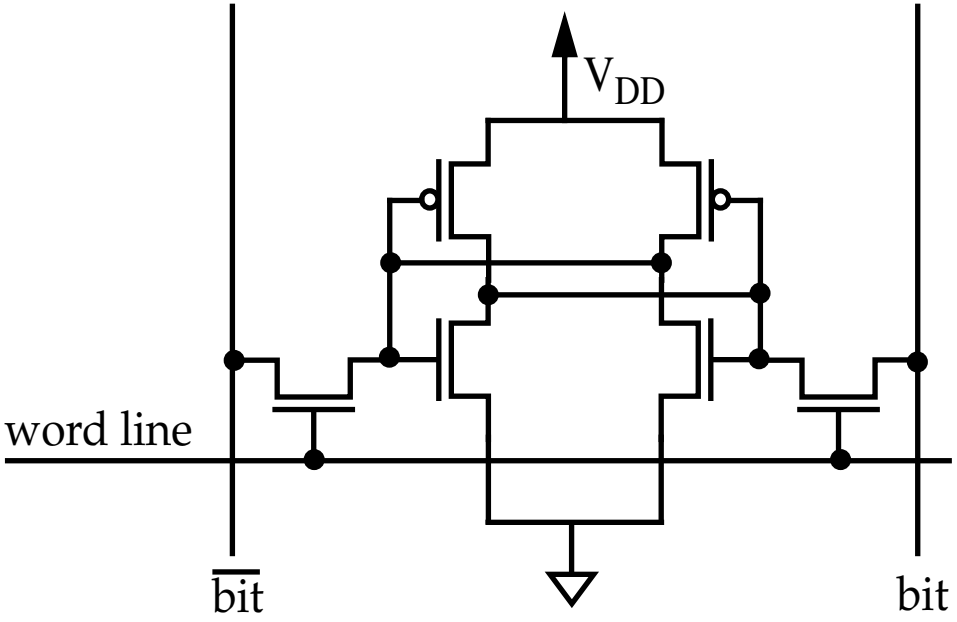
Two basic versions: *dynamic RAM* (DRAM) and *static RAM* (SRAM).

A DRAM cell consists of a single transistor and a capacitor, which stores a 0 or 1.

Dynamic because the caps loose charge over time and need refreshed.

SRAM, on the other hand, does not need a *periodic refresh* cycle.

Once loaded, the value will remain until power is turned off.

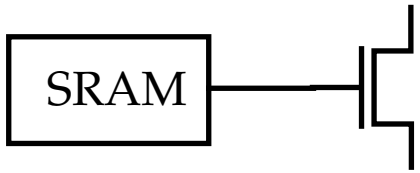


An SRAM cell

SRAM-based Technologies

The VLSI course will cover the details of operation.

For this course, we use it as shown below:



SRAM cell's value *enables* or *disables* control transistor.

Disadvantages include the large number of transistors per SRAM cell and its *volatility*, program state is lost when power is turned off.

Advantages include very fast programming and near limitless number of reprogrammings.

Future: *Magnetic RAM* uses two ferromagnetic layers separated by a thin insulating layer.

Have potential to combine high speed of SRAM, storage capacity of DRAM and *nonvolatility* of FLASH, while using small amounts of power.

Summary

Technology	Predominantly associated with...
Fusible-link	SPLDs
Antifuse	FPGAs
EPROM	SPLDs and CPLDs
EEPROM and FLASH	SPLDs and CPLDs (some FPGAs)
SRAM	FPGAs (some CPLDs)