Programmable Logic Applications and Architectures

After this lecture you should be able to: explain the growth of digital electronics; select software versus hardware and PLDs versus ASICs to solve a particular problem; explain the terms: Moore's Law, ASIC, CPLD, FPGA, feature size, VLSI, fabless, wafer, die, NRE, FPGA, LE and LUT.

Hardware vs Software

When would you design hardware instead of writing software?

Software-based solutions are generally preferred due to:

- Lower Development Cost. Existing hardware (e.g. a microcontroller) can be customized for a new application with software.
- Shorter Development Time. Software can be coded more quickly than equivalent hardware can be designed.
- Lower Risk. Errors can be fixed and new features added by deploying a new version of the software.
- More Expertise. There are many more[1] software developers than digital hardware engineers.

However, there are several situations for which a hardware solution is typically necessary:

- Low Delay. When the system must respond within a few clock cycles (e.g. a few nanoseconds) a hardware implementation will be necessary. A typical example is high-speed interfaces such as Ethernet.
- Computational Throughput. A hardware processor (or co-processor) will be required when the computational complexity of the problem exceeds the capability of a sequential processor because multiple operations must be completed per clock cycle. An example is a GPU (Graphics Processing Unit) that can process multiple pixels per clock cycle.
- Energy Efficiency. Implementing an algorithm on a CPU will require more cycles and more registers. All else being equal[2] a software solution will consume more energy.
- Correctness. Software can be easily modified to correct errors or introduce new features. However, this increases the motivation to release designs that are not fully tested and the likelihood that errors will be introduced into existing designs.

Exercise 1: Would you use hardware or software to implement: A new calculator? A digital watch? A controller for a kitchen appliance? An Ethernet interface? For Cryptocurrency "mining"? For an aircraft's automated landing system?

Introduction

Digital ICs have increased in complexity at an exponential rate over the last 40 years. This growth has been at a rate predicted by "Moore's Law" – an observation that digital IC complexity per unit area seems to double every 2 to 3 years.

Moore's law does not apply to analog ICs. This is because the die area required for an analog IC (e.g. an op-amp) is determined by factors such as its voltage and power rating rather than by the minimum transistor size.

The steady decline in the cost of digital relative to analog electronics has resulted in modern electronic devices implementing almost all functionality using digital rather than analog electronics. The main exception is interface electronics, including power electronics.

Digital ICs can be classified by the number of gates or transistors on an IC (e.g. SSI, LSI and VLSI standing for small, large and very large scale integration). Application-specific ICs (ASICs) are ICs designed for

[1] Probably more than 10 times as many based on a comparison of Statistics Canada NOC 2174 (programmers) and 2147 (computer engineers).

[2] Rarely the case.

a specific application (e.g. a graphics processor for a video card or the WLAN transceiver in a cell phone) as opposed to being suited to many different applications (e.g. the CPU in a PC or the microcontroller in an appliance). Many ASICs include a general-purpose CPU as well as memory and peripheral interfaces and application-specific components such as graphics or signal processors. These are often called “System on a Chip” (SoC).

Digital ICs are also classified by the “feature size” of their masks measured in nanometers. In 2023, ICs with 3 nm features sizes started manufacturing.

Due to the cost of the equipment required to manufacture ICs with such small feature sizes, only a handful of companies own fabrication plants (“fabs”) that manufacture digital ICs. Instead, most semiconductor companies are “fabless.” These companies design and sell their ICs but use “fabs” to manufacture them.

IC Manufacturing

IC’s are manufactured on (typically) 300 mm diameter wafers of crystalline silicon. Each wafer is put through dozens of manufacturing steps where dopants are diffused into the silicon and alternating layers of insulating (dielectric) and conductive (metal) materials are deposited to build the circuit. Each step requires a “mask” that is used to expose a photoresist so the processing can be limited to specific areas.

The one-time costs of preparing for production is called Non-Recurring Engineering (NRE). ASIC NRE, primarily preparing a mask set for a digital IC, is very high (e.g. $\$10^6$) because of the very small dimensions involved.

Exercise 2: What improvement in number of transistors per unit area would be achieved by reducing the feature size from 7 nm to 5 nm? Approximately how many 5x5 mm die fit on a 300 mm wafer? How many 200x200 nm gates fit on the die?

PLDs

Programmable Logic Devices PLDs[3] are Integrated Circuits (ICs) that can be configured after manufacturing to implement different logic functions. Unlike software that consists of operations on fixed word sizes, drawn from a limited instruction set and performed in sequence, a PLD’s logic functions can be defined in more detail and can be performed in parallel if necessary.

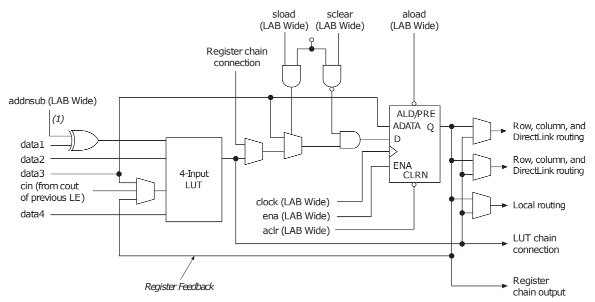
PLDs are often categorized into three major types: PALs (Programmable Array Logic, now obsolete), CPLDs (Complex PLDs) and FPGAs (Field-Programmable Gate Arrays). CPLDs have limited functionality and are often used as “glue logic.” FPGAs are more capable and can often replace ASICs. The different types of PLDs are described below.

The two largest PLD companies are Xilinx[4] with about 50% of the market and Intel[5], with about 30% of the market.

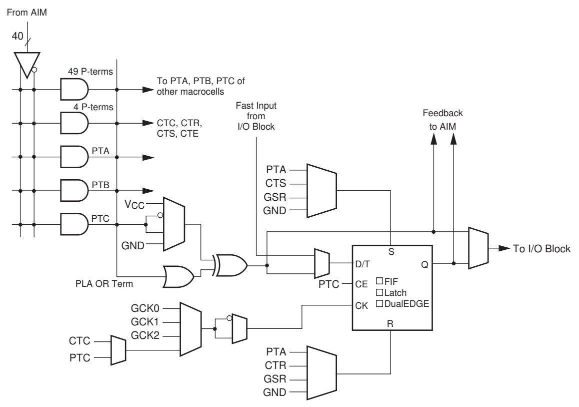
PLD Architectures

CPLD

A CPLD is composed of small programmable logic blocks with combinational logic at the input to a flip-flop. The Intel logic blocks are called Logic Elements. An example is the Intel MAX-V LE:



The Xilinx logic blocks are called Macrocells (MC):



[3] Not to be confused with a Programmable Logic Controller, or PLC, which is a piece of equipment used for industrial control applications rather than an IC.

[4] Acquired by AMD.

[5] Formerly Altera.

The main difference is the structure of the combinational logic – either one 4-input look-up table (LUT) per LE in for Intel CPLDs or a larger programmable logic array (PLA) driving 16 MC for Xilinx.

In both cases a programmable interconnect matrix allows the inputs and outputs of logic blocks to be interconnected. A wide range of logic functions can be implemented by configuring the LUTs or PLAs and programming the interconnect matrix.

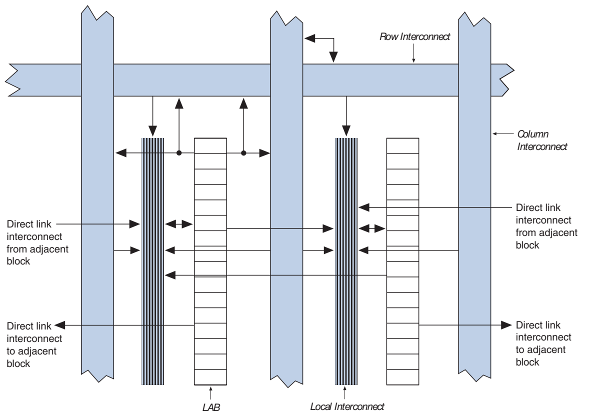
Small CPLDs sell for about $1 or $2 in small quantities (e.g. a Xilinx "Coolrunner" XC2C32A with 32 I/O pins and 32 MC or the Intel MAX V 5M40 with 64 I/O pins and 40 LE).

FPGA

Gate arrays were an attempt to bridge the gap between fully custom ICs and PLDs. The idea was that gates would be laid out in predefined locations on the die and only the last few layers of interconnect would need to be customized for each IC thus reducing the number of custom masks needed for each IC and thus the NRE. This approach is no longer popular.

This idea developed into a Field-Programmable Gate Array. An FPGA is a PLD that contains a large number (thousands to hundreds of thousands) of simple logic elements similar to those in an Intel CPLD. Note that modern FPGAs have no gates – the logic for each LE is implemented using small memories called look-up tables (LUTs).

However, an FPGA's interconnection resources are more limited than those in a CPLD. In keeping with the idea that multiple LEs will be combined to form multi-bit logic functions, logic elements can be connected to their neighbours in logic array blocks (LABs) and these to row and column interconnect buses:


Row Interconnect
Column Interconnect
Direct link interconnect from adjacent block
Direct link interconnect from adjacent block
Direct link interconnect to adjacent block
Direct link interconnect to adjacent block
LAB
Local Interconnect

Complex software is required to fit a design into an FPGA and route the signals between logic elements. Because of the multiple levels of interconnect the propagation delays are harder to predict than for a CPLD. Even if there is sufficient logic, some designs may not 'fit' into an FPGA because of insufficient routing resources.

Modern FPGAs include special-purpose components such as RAM, multipliers, PLL clock generators and high-speed serial I/O in addition to general-purpose logic elements.

Most modern FPGAs have enough logic elements and memory that they can be configured with a "soft" CPU (e.g. Altera's Nios and Xilinx's MicroBlaze). This allows the FPGA to include both software and hardware functions. Some FPGAs include a (hardware-based, typically ARM) CPU core for applications that require both an SoC and programmable logic.

Depending on the version, the FPGA might contain between 6 k and 114 k logic elements and between 180 and 530 I/O pins. Smaller FPGAs sell for under $10 in small quantities. However, large high-performance FPGAs, often used for ASIC prototyping, can cost many thousands of dollars. Due to the high I/O count most FPGAs use ball grid array (BGA) packages.

Note that CPLD's and FPGA's from Xilinx and Intel have not been available (to most customers) for a couple of years.

PLD Configuration

Although most CPLDs have on-board non-volatile configuration memory, most FPGAs use volatile configuration memory which must be reloaded each time the device powers up. The FPGA can load itself from an external, typically serial, EEPROM or it can be configured through the JTAG interface. On larger systems that include processors the FPGA is often configured by software running on the processor and in this case the FPGA configuration can be changed as part of a firmware update.

Technology Selection

PLD vs ASIC

If the decision is to use hardware, when would you use programmable logic instead of designing a custom IC (an ASIC)?

In most cases all of the required functionality will already be available as part of an existing "off-the-shelf" ASIC or SoC. For example, many SoCs intended for cell phones include peripheral interfaces and graphics accelerators.

However, in some cases the required hardware function is not available and in this case a decision must be made whether to design a new IC or use an FPGA. The decision depends on the following factors:

- Performance. This includes clock speed and power consumption. An ASIC (or custom full-custom IC) will execute faster and use less power that the same function programmed into an FPGA.

- Sales volume. The NRE for designing the IC (e.g. masks) must be recovered from sales. If the sales volumes are low then the NRE costs per unit may be too high.

- Time to Market (TTM). It takes many months to design, verify and manufacture a custom IC. If TTM is short then an FPGA may a better option.

- Flexibility. If the functionality of the IC is likely to change then a PLD may be a better option. The design can even be changed in the field after the product has been delivered to the customer.

- Risk. Errors in an IC design will require a "re-spin" of the IC resulting in additional costs and delays. For a small company this can be fatal.

Because of the above considerations, FPGAs tend to be used in non-consumer products that have relatively low volumes and are relatively cost-insensitive. Examples of typical application areas include telecommunications infrastructure (e.g. large Internet routers and cellular base stations), medical equipment (e.g. CAT scanners) and military/aerospace (e.g. avionics).

Exercise 3: Would you use a PLD or ASIC for: A project that had to be completed within a month? That would be expected to sell 100 million units? Whose complete requirements aren't known? A state-of-the-art general-purpose CPU?