Digital System Design

§1 - INTRODUCTION

The objectives of this course are for you to learn:

- How to go about designing complex, ***high speed*** digital systems (not just circuits)?
- How to use some of the modern CAD tools to help with the design?
- How to implement such designs using programmable logic (e.g. FPGAs)?
- How to read data sheets and make sense of them?
- How do digital building blocks (such as memory chips, microprocessors, arithmetic circuits etc.) work?
- How to interface to microprocessors and computers (from hardware point of view)?
- How to deal with testing of complex systems?

The course syllabus is divided into six main sections:

1) Programmable Logic

In most digital systems, the circuitry can be divided into two major sections: datapath and control. Datapath circuits perform operations on data (e.g. adding numbers together, counting events etc.) while the control circuitry generates the control signals needed to ensure that such operations are performed at the right times and in the right order. The datapath circuitry is generally fairly straightforward. In fact most commonly used functions are usually available as a library component which are parameterisable. The control circuitry is more challenging and interesting. We will be examining how both datapath circuits and control circuit can be designed using FPGA and CPLD chips.

2) Arithmetic Circuits

You use adders, multipliers etc. all the time. How do they work? What are the different trade-offs in choosing one circuit instead of another? How do floating-point arithmetic circuits work? To understand how arithmetic circuits work, it is necessary to relate logical operations performed on the bits that make up a digital number to the arithmetic effects that they have on the number's value.

3) Data Encoding

Complex digital systems are invariably broken down into subsystems. We shall look at the ways in which information is transmitted between subsystems, how errors can be detected and corrected.

4) High Speed Design Issues

Modern digital systems run at system clock rate of 66 MHz and above, and on-chip clock rate of well over 100 MHz. We will examine issues relating to logic interfacing, metastability, transmission line effects etc. which affect system performance.

5) Architectures

Given an algorithm, how can it be implemented with digital hardware? Architectures are the different ways that an algorithm can be implemented as circuits. For example, one might use parallel, serial-parallel, array-type architecture. In particular, we will be examining some well-known architecture for digital hardware, e.g. distributed arithmetic based circuits and cordic based circuits.

6) Testing

Hardware is getting denser. Packaging technology is improving to so much that signal pins are either difficult or impossible to get at! Testing complex digital circuits become extremely difficult. You will learn about a new international standard known as JTAG or boundary-scan for digital board testing.

Books

There are many books available on Digital Circuit Design - most of them range from pretty rotten, to simplistic, to moderate, to good! No single book was found to contain all the materials covered on this course. The course is therefore supported by full printed notes. However, I recommend you to consider the following well-written textbooks:

"Digital Design Principles and Practices", 4th Edition (Sept 2005), John F. Wakerly, Prentice Hall.

This is a new edition of a well established textbook. It covers a significant portion of the materials taught on this course. At ~£45 (www.whsmith.co.uk), this a bargain. Recommended purchase if you have not already done so!

"Contemporary Logic Design", Gaetano Boriello, Randy H. Katz, August 2004, Prentice Hall.

Good coverage on finite state machines and computer architectures. (~£39)

"High-Speed Digital Design - A handbook of black magic", Howard G. Johnson, Prentice Hall, 1993; ISBN 0-13-395724-1 (£61). The best practical guide to designing and building very high speed digital circuits. Expensive reference for your company to buy.

Course Work Assessment

The most effective way to learn about digital design is actually doing it! As part of this course, I will require a piece of course work to be submitted. The course work involves designing a chip to decode JPEG images in hardware using Xilinx Virtex-2 Pro FPGA board. Details will follow later.

§2 - DESIGN METHODOLOGIES & IMPLEMENTATION TECHNOLOGIES

2.1 Different Levels of Design

It is possible to design a digital system using ad hoc, intuitive method, using pen and paper, and a bit of brain cells. However, if the system is of moderate to high complexity, ad hoc methods do not usually lead to working systems (definitely not right-first-time), let alone optimised systems! Much has been learned from ***structured***, ***top-down*** design method developed for software engineering. Hardware engineers are now approach design much more systematically than before. In addition, computers are getting cheaper and more powerful. CAD tools are also becoming more available, even to (cash-starved) Universities. It is now possible to design complex digital circuits systematically on a computer, simulate the entire design down to component level and verify that everything works before even considering making any hardware!

Designing a complex system can be approached at different levels. Figure 2.1 shows the divisions between them with some relevant explanations:

Design Levels	*Design Descriptions*	*Primitive Components*	*Theoretical Techniques*
Algorithmic	Specifications High-level lang. Math. equations	Functional blocks 'black boxes'	Signal processing theory Control theory Sorting algorithm
Functional	VHDL, Verilog FSM language C/Pascal	Registers Counters ALU	Automata theory Timing analysis
Logic	Boolean equations Truth tables Timing diagrams	Logic gates Flip-flops	Boolean algebra K-map Boolean minimization
Circuit	Circuit equations Transistor netlist	Transistors Passive comp.	Linear/non-linear eq. Fourier analysis

Figure 2.1 Levels of Design

2.2 The Design Process

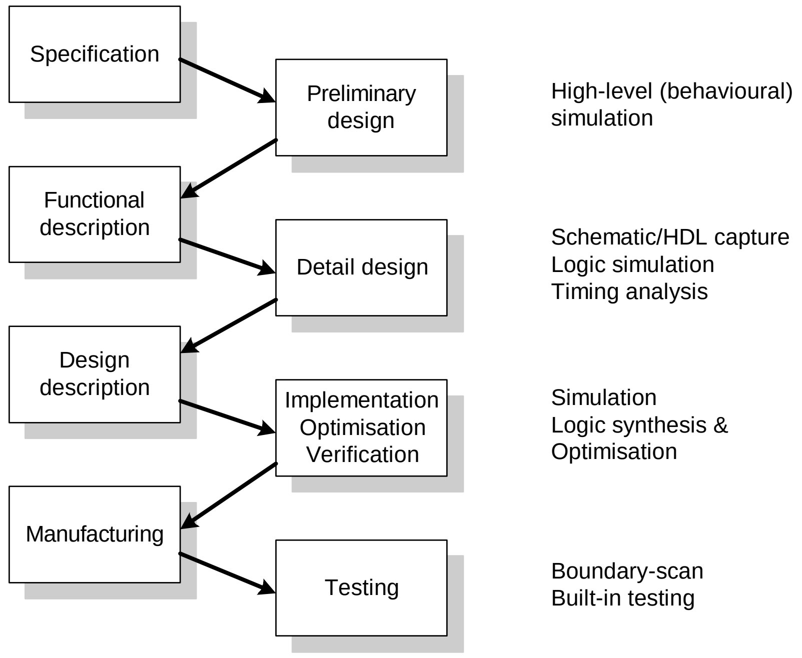

Figure 2.2 The Top-down Design Process

❑ **Top-down design strategies**
- Refine Specification successively
- Decompose each component into small components
- Lowest-level primitive components
- Over-sold methodology - only works with plenty of experience

❑ **Bottom-up design strategies**
- Build-up from primitive components
- Combined to form more complex components
- Risk wrong interpretation of specifications

❑ **Mixed strategies**
- Mostly top-down, but also bits of bottom-up
- Reality: need to know both top level and bottom level constraints

2.3 Design Descriptions

Designs can be described in many ways. Representation and notations are extremely important because they can affect how designs can be view and refined. The most appropriate description for a design depends on which level of design one is engaging in. For example, top-level specification could be in natural language (as in many formal specifications), in equations form (such as difference equation for a digital filter), in data-flow diagrams (as in software), in behavioural languages such as AHDL or VHDL.

It also depends on what type of circuit you are designing. For example, the easiest way to define a finite-state machine is a special finite-state machine language (such as AHDL on Altera's Maxplus-II). To obtain an overview of the entire system, some form of graphical block diagram is usually very helpful.

Very roughly, design descriptions can be divided into graphical description using schematic capture or language description using some form of hardware description language. Here is a comparison between the two:

Schematic capture	***Hardware Description Languages***
Good for multiple data flow Give overview picture Relates to hardware better Doesn't need to be good in computing High information density Back annotations possible Mixed analogue/digital possible	Flexible & parameterisable Excellent for optimisation & synthesis Direct mapping to algorithms Excellent for datapaths Readily interfaced to optimiser Easy to handle and transmit (electronically)
Not good for algorithms *Not good for datapaths* *Doesn't interface well in optimiser* *No good for synthesis software* *Difficult to reuse* *Not parameterisable*	*Essentially serial representation* *May not show overall picture* *Often need good programming skill* *Divorce from physical hardware* *Need special software*

2.4 Design tools

CAD design tools are commonly available. They range from cheap PC-based PCB layout tools, to medium cost PC-based schematic capture and simulation programs to expensive workstation based packages. You, in your earlier years, have already been exposed to Altera's MaxPlus-II Design System. On this course, you will be using Xilinx's ISE for all the designs, a version of which can be downloaded from the web free of charge! Both systems are available on any PC installed in all the teaching and computer laboratories in the Department.

The Altera and Xilinx design systems contain the most of the following features:

- Schematic Capture
- Hardware Description Language Entry
- Logic Synthesis & Optimisation
- Timing Analysis
- Parameterised Library Components
- Hierarchical Design Management
- Symbol Editing
- Simulation with Timing
- Autoplacement and Routing
- Floorplan Editing
- Reporting

2.5 Implementation Technologies

Good designs cannot be divorced from implementation and technology. How can one turn a digital design into real circuits? Figure 2.3 shows a typical digital system and its constituent parts. It usually has a microprocessor, a bank of memory (RAM or ROM), other off-the-shelf complex circuits such as ADC, DAC, Modem chip etc., and glue-logic which connect all these parts together. The glue logic generates the necessary control signals for the rest of the system. In general, a digital system can be implemented with five types of components:

- ***Standard parts***
- ***Special off-the-shelf parts***
- ***Programmable Logic Devices***
- ***Programmable Gate Arrays***
- ***Custom Integrated Circuits***

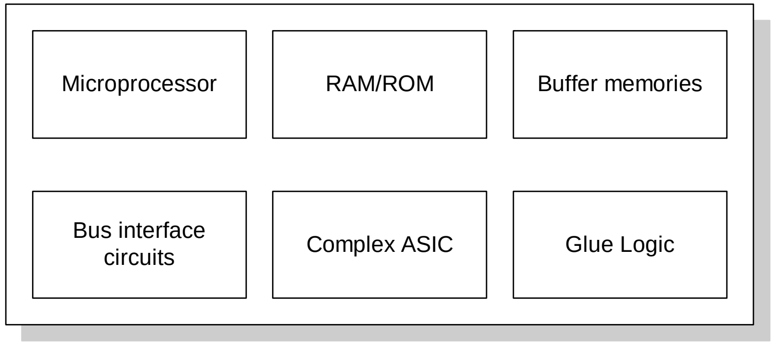

Figure 2.3 The constituent parts of a typical digital system

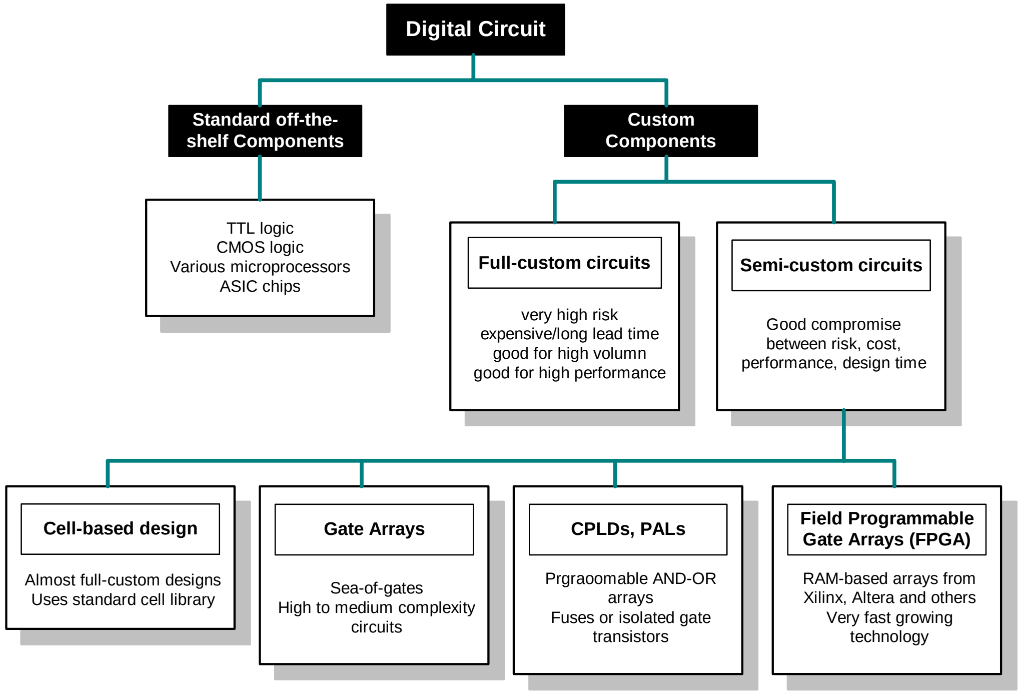
Digital Circuit
Standard off-the-shelf Components
Custom Components
TTL logic
CMOS logic
Various microprocessors
ASIC chips
Full-custom circuits
very high risk
expensive/long lead time
good for high volumn
good for high performance
Semi-custom circuits
Good compromise between risk, cost, performance, design time
Cell-based design
Almost full-custom designs
Uses standard cell library
Gate Arrays
Sea-of-gates
High to medium complexity circuits
CPLDs, PALs
Prgraoomable AND-OR arrays
Fuses or isolated gate transistors
Field Programmable Gate Arrays (FPGA)
RAM-based arrays from Xilinx, Altera and others
Very fast growing technology

Figure 2.4 Implementation Technologies