

# Overview of Digital Design

Dr DC Hendry

January 2006

# Outline I

- 1 Course Information:
- 2 Digital Systems
- 3 Design Techniques
  - Manual Techniques
  - The Hardware Description Language Approach
- 4 Implementation Techniques
  - Discrete Components
  - VLSI Devices

## Outline II

- FPGAs

- 5 Complexity

- Hierarchy
- Abstraction

- 6 Challenges

## Course Information:

① Lecturer: Dr David Hendry.

## Course Information:

- 1 Lecturer: Dr David Hendry.
- 2 Three meetings per week, normally two lectures and one tutorial.

## Course Information:

- 1 Lecturer: Dr David Hendry.
- 2 Three meetings per week, normally two lectures and one tutorial.
- 3 Also laboratories, as advertised on the level 3 notice board.

## Course Information:

- 1 Lecturer: Dr David Hendry.
- 2 Three meetings per week, normally two lectures and one tutorial.
- 3 Also laboratories, as advertised on the level 3 notice board.
- 4 **Recommended texts: one of**

## Course Information:

- 1 Lecturer: Dr David Hendry.
- 2 Three meetings per week, normally two lectures and one tutorial.
- 3 Also laboratories, as advertised on the level 3 notice board.
- 4 Recommended texts: one of
  - Zwolinski “Digital System Design with VHDL” Pearson.

## Course Information:

- 1 Lecturer: Dr David Hendry.
- 2 Three meetings per week, normally two lectures and one tutorial.
- 3 Also laboratories, as advertised on the level 3 notice board.
- 4 Recommended texts: one of
  - Zwolinski “Digital System Design with VHDL” Pearson.
  - Dewey “Analysis and Design of Digital Systems with VHDL” PWS

## Course Information:

- 1 Lecturer: Dr David Hendry.
- 2 Three meetings per week, normally two lectures and one tutorial.
- 3 Also laboratories, as advertised on the level 3 notice board.
- 4 Recommended texts: one of
  - Zwolinski “Digital System Design with VHDL” Pearson.
  - Dewey “Analysis and Design of Digital Systems with VHDL” PWS
  - **Floyd “Digital Fundamentals with VHDL” Pearson.**

## Course Information:

- 1 Lecturer: Dr David Hendry.
- 2 Three meetings per week, normally two lectures and one tutorial.
- 3 Also laboratories, as advertised on the level 3 notice board.
- 4 Recommended texts: one of
  - Zwolinski “Digital System Design with VHDL” Pearson.
  - Dewey “Analysis and Design of Digital Systems with VHDL” PWS
  - Floyd “Digital Fundamentals with VHDL” Pearson.
- 5 Web Page: <http://www.eng.abdn.ac.uk/~eng186/eg3560>

# Digital Systems

- All data represented by (a large number) of '0's and '1's.

# Digital Systems

- All data represented by (a large number) of '0's and '1's.
- High repeatability without adjustment. Adjustment requires human intervention and expensive test equipment.

# Digital Systems

- All data represented by (a large number) of '0's and '1's.
- High repeatability without adjustment. Adjustment requires human intervention and expensive test equipment.
- Tolerant to temperature and other variations such as power supply voltage.

# Digital Systems

- All data represented by (a large number) of '0's and '1's.
- High repeatability without adjustment. Adjustment requires human intervention and expensive test equipment.
- Tolerant to temperature and other variations such as power supply voltage.
- Provides an element of noise immunity.

# Digital Systems

- All data represented by (a large number) of '0's and '1's.
- High repeatability without adjustment. Adjustment requires human intervention and expensive test equipment.
- Tolerant to temperature and other variations such as power supply voltage.
- Provides an element of noise immunity.
- Early systems physically massive, the equivalent of today's PC requires a number of rooms to contain.

# Digital Systems

- All data represented by (a large number) of '0's and '1's.
- High repeatability without adjustment. Adjustment requires human intervention and expensive test equipment.
- Tolerant to temperature and other variations such as power supply voltage.
- Provides an element of noise immunity.
- Early systems physically massive, the equivalent of today's PC requires a number of rooms to contain.
- **Early systems unreliable - based on vacuum tube technology.**

# Digital Systems

- All data represented by (a large number) of '0's and '1's.
- High repeatability without adjustment. Adjustment requires human intervention and expensive test equipment.
- Tolerant to temperature and other variations such as power supply voltage.
- Provides an element of noise immunity.
- Early systems physically massive, the equivalent of today's PC requires a number of rooms to contain.
- Early systems unreliable - based on vacuum tube technology.
- **Early systems costly compared to analogue systems.**

## Effect of VLSI

- VLSI = Very Large Scale Integration devices, also known as *silicon chips*.

## Effect of VLSI

- VLSI = Very Large Scale Integration devices, also known as *silicon chips*.
- Essentially a photographic process incorporating many millions of transistors on one silicon chip.

## Effect of VLSI

- VLSI = Very Large Scale Integration devices, also known as *silicon chips*.
- Essentially a photographic process incorporating many millions of transistors on one silicon chip.
- 5mm × 5mm chip can contain about 3 million logic gates. Early microprocessors used less than 10,000 logic gates. Typical chip size from 1mm × 1mm to 25mm × 25mm.

## Effect of VLSI

- VLSI = Very Large Scale Integration devices, also known as *silicon chips*.
- Essentially a photographic process incorporating many millions of transistors on one silicon chip.
- 5mm x 5mm chip can contain about 3 million logic gates. Early microprocessors used less than 10,000 logic gates. Typical chip size from 1mm x 1mm to 25mm x 25mm.
- Manufacturing cost of such a device about \$5 in quantity.

## Effect of VLSI

- VLSI = Very Large Scale Integration devices, also known as *silicon chips*.
- Essentially a photographic process incorporating many millions of transistors on one silicon chip.
- 5mm x 5mm chip can contain about 3 million logic gates. Early microprocessors used less than 10,000 logic gates. Typical chip size from 1mm x 1mm to 25mm x 25mm.
- Manufacturing cost of such a device about \$5 in quantity.
- Design cost however could be \$5M. Design time needs to be short to meet market opportunities.

# The Design Problem

- Typical designs involve many millions of logic gates.

# The Design Problem

- Typical designs involve many millions of logic gates.
- To correct an error in a silicon chip takes at least three months and about \$0.5M.

# The Design Problem

- Typical designs involve many millions of logic gates.
- To correct an error in a silicon chip takes at least three months and about \$0.5M.
- Design times must be short to meet market opportunities.

# The Design Problem

- Typical designs involve many millions of logic gates.
- To correct an error in a silicon chip takes at least three months and about \$0.5M.
- Design times must be short to meet market opportunities.
- Design Automation, through the use of CAD (Computer Aided Design) tools *is essential*.

# Design Techniques

- **Manual Techniques.** Early design techniques, including CAD supported techniques, required designers to carry out manual (paper and pencil) calculations and then enter schematics onto paper, or, into a computer.

# Design Techniques

- Manual Techniques. Early design techniques, including CAD supported techniques, required designers to carry out manual (paper and pencil) calculations and then enter schematics onto paper, or, into a computer.
- Synthesis Based Techniques. Almost all designs now use *synthesis tools* to convert a textual description of the required design into a *netlist* (a list of logic gates and the connections between).

# Manual Techniques

- 1 Starts with pen and paper techniques - K-Maps, manual design of FSMs (Finite State Machines) and manual manipulation of Boolean Equations.

# Manual Techniques

- 1 Starts with pen and paper techniques - K-Maps, manual design of FSMs (Finite State Machines) and manual manipulation of Boolean Equations.
- 2 Early design decisions difficult to reverse.

# Manual Techniques

- 1 Starts with pen and paper techniques - K-Maps, manual design of FSMs (Finite State Machines) and manual manipulation of Boolean Equations.
- 2 Early design decisions difficult to reverse.
- 3 Early CAD tools added schematic capture which permitted accurate simulation of the design.

# Manual Techniques

- 1 Starts with pen and paper techniques - K-Maps, manual design of FSMs (Finite State Machines) and manual manipulation of Boolean Equations.
- 2 Early design decisions difficult to reverse.
- 3 Early CAD tools added schematic capture which permitted accurate simulation of the design.
- 4 Use of hierarchy could reduce design complexity to a degree.

# Manual Techniques

- 1 Starts with pen and paper techniques - K-Maps, manual design of FSMs (Finite State Machines) and manual manipulation of Boolean Equations.
- 2 Early design decisions difficult to reverse.
- 3 Early CAD tools added schematic capture which permitted accurate simulation of the design.
- 4 Use of hierarchy could reduce design complexity to a degree.
- 5 Increasing design size however meant that synthesis based techniques were seen as essential.

# The HDL Approach

- HDL = Hardware Description Language, such as VHDL or Verilog. Also newer languages such as System-C or Superlog.

# The HDL Approach

- HDL = Hardware Description Language, such as VHDL or Verilog. Also newer languages such as System-C or Superlog.
- Similar to software languages, write code in a high level language, then compile. In this case the compiler is called a *Synthesis tool*, and the output is a netlist.

# The HDL Approach

- HDL = Hardware Description Language, such as VHDL or Verilog. Also newer languages such as System-C or Superlog.
- Similar to software languages, write code in a high level language, then compile. In this case the compiler is called a *Synthesis tool*, and the output is a netlist.
- **Easier design exploration, various approaches to a design can be efficiently attempted.**

# The HDL Approach

- HDL = Hardware Description Language, such as VHDL or Verilog. Also newer languages such as System-C or Superlog.
- Similar to software languages, write code in a high level language, then compile. In this case the compiler is called a *Synthesis tool*, and the output is a netlist.
- Easier design exploration, various approaches to a design can be efficiently attempted.
- Much improved productivity, include possibility of *IP core business*.

# The HDL Approach

- HDL = Hardware Description Language, such as VHDL or Verilog. Also newer languages such as System-C or Superlog.
- Similar to software languages, write code in a high level language, then compile. In this case the compiler is called a *Synthesis tool*, and the output is a netlist.
- Easier design exploration, various approaches to a design can be efficiently attempted.
- Much improved productivity, include possibility of *IP* core business.
- Design can be retargeted to another silicon process or to an FPGA (see below).

# Intellectual Property (IP) Cores

- Increasing design size means that even subcomponents are highly complex (for example, a microprocessor, or an FEC (Forward Error Correction) circuit).

# Intellectual Property (IP) Cores

- Increasing design size means that even subcomponents are highly complex (for example, a microprocessor, or an FEC (Forward Error Correction) circuit).
- **Synthesisable subcomponents now sold by one company to another for incorporation into a larger design. See for example AXEON or 4i2i (both local Aberdeen companies).**

# Intellectual Property (IP) Cores

- Increasing design size means that even subcomponents are highly complex (for example, a microprocessor, or an FEC (Forward Error Correction) circuit).
- Synthesisable subcomponents now sold by one company to another for incorporation into a larger design. See for example AXEON or 4i2i (both local Aberdeen companies).
- IP product consists of synthesisable code, simulation testbenches, test support and other CAD tool support.

# Implementation Techniques

- 1 Early systems constructed with discrete components. This gave rise to very expensive, large machines. Early computers occupied a room as large or larger than this lecture theatre.

# Implementation Techniques

- 1 Early systems constructed with discrete components. This gave rise to very expensive, large machines. Early computers occupied a room as large or larger than this lecture theatre.
- 2 VLSI devices incorporate many millions of logic gates onto one piece of silicon, very cost effective, reliable, low power and fast.

# Implementation Techniques

- 1 Early systems constructed with discrete components. This gave rise to very expensive, large machines. Early computers occupied a room as large or larger than this lecture theatre.
- 2 VLSI devices incorporate many millions of logic gates onto one piece of silicon, very cost effective, reliable, low power and fast.
- 3 FPGA (Field Programmable Gate Arrays) provide some millions of logic gates which can be programmed to a particular function in minutes. Good for low to medium volume and for development.

# Field Programmable Gate Arrays

- ① Gate Arrays - these devices consist of an array of logic gates whose function is determined by a number of memory bits.

# Field Programmable Gate Arrays

- 1 Gate Arrays - these devices consist of an array of logic gates whose function is determined by a number of memory bits.
- 2 Field Programmable - the memory bits can be programmed by the end user, typically using a PC or by storing the programming data in non-volatile memory.

# Field Programmable Gate Arrays

- 1 Gate Arrays - these devices consist of an array of logic gates whose function is determined by a number of memory bits.
- 2 Field Programmable - the memory bits can be programmed by the end user, typically using a PC or by storing the programming data in non-volatile memory.
- 3 Provide up to millions of logic gates (but at a high cost).

# Field Programmable Gate Arrays

- 1 Gate Arrays - these devices consist of an array of logic gates whose function is determined by a number of memory bits.
- 2 Field Programmable - the memory bits can be programmed by the end user, typically using a PC or by storing the programming data in non-volatile memory.
- 3 Provide up to millions of logic gates (but at a high cost).
- 4 Often used to prototype VLSI devices.

# Field Programmable Gate Arrays

- 1 Gate Arrays - these devices consist of an array of logic gates whose function is determined by a number of memory bits.
- 2 Field Programmable - the memory bits can be programmed by the end user, typically using a PC or by storing the programming data in non-volatile memory.
- 3 Provide up to millions of logic gates (but at a high cost).
- 4 Often used to prototype VLSI devices.
- 5 Latest devices include embedded multipliers and memories, so providing excellent DSP facilities.

# Field Programmable Gate Arrays

- 1 Gate Arrays - these devices consist of an array of logic gates whose function is determined by a number of memory bits.
- 2 Field Programmable - the memory bits can be programmed by the end user, typically using a PC or by storing the programming data in non-volatile memory.
- 3 Provide up to millions of logic gates (but at a high cost).
- 4 Often used to prototype VLSI devices.
- 5 Latest devices include embedded multipliers and memories, so providing excellent DSP facilities.
- 6 **Industry leader is Xilinx (see [www.xilinx.com](http://www.xilinx.com)).**

# Complexity

**Hierarchy** We use hierarchy to construct a complex component from a number of simpler components. Often the same simpler component is used many times (for example, constructing an N-bit adder from N one bit adders).

# Complexity

**Hierarchy** We use hierarchy to construct a complex component from a number of simpler components. Often the same simpler component is used many times (for example, constructing an N-bit adder from N one bit adders).

**Abstraction** At different points in the design we may for example consider individual bits, or consider those same bits as a single integer, or that integer as one element of a time sequence.

Less detail, highly abstract

Processors

Instructions

RTL

Logic Gate

Transistors

Layout

More detail, many simple blocks

# Semiconductor Industry Roadmap

|                        |      |      |      |      |      |
|------------------------|------|------|------|------|------|
| Year Introduced        | 1999 | 2002 | 2004 | 2008 | 2011 |
| Feature Size <i>nm</i> | 180  | 130  | 90   | 60   | 40   |
| Transistors (Millions) | 110  | 220  | 441  | 1852 | 4568 |

Table: Semiconductor Industry Roadmap - ASIC