Programmable Logic Devices

Tinoosh Mohsenin
CMPE 415
Today

- Administrative items
- Syllabus and course overview
- Digital signal processing overview
Course Description

- Concepts, features and programming programmable logic devices such as FPGAs.
- Hardware Description Languages (HDLs) are used to create designs.
- Advanced topics in logic design
  - Pipelining
  - Memory system design
  - Fixedpoint arithmetic
  - Timing Analysis
  - Low Power Design (if time permits)
Course Description

- Computer Aided Design of large/complex digital system
  - Verilog
  - Xilinx ISE flow
    - Simulation (isim)
    - Synthesis and place & route
  - FPGA verification
    - Spartan 3E

Prerequisite
- CMPE 310 (Systems Design and Programming)
Course Communication

- Email
  - Urgent announcements
- Web page
  - http://www.csee.umbc.edu/~tinoosh/cmpe415/
- Office hours
  - After class, or by appointment
- TAs
  - Amey Kulkarni
- TA hours
Course Description

- Lectures (on board+ slides)
- Handouts/tutorials
- Many Homework/lab projects
  - 8 homework
- Midterm Exam
  - Mid March
- Final exam and (probably final project)
- Quizzes
Lectures

- Ask questions at any time
- Participate in the class (%5 of your grade)
- Please silence phones
- Please hold conversations outside of class
- No computer usage in classroom
Programmable Logic Devices

- Can be programmed after manufacture to provide different functions, unlike application specific integrated circuits (ASICs).

- Examples:
  - Programmable array logics (PAL)
  - Complex programmable logic devices (CPLD)
  - Field Programmable Gate Arrays (FPGA)
Trends in Cellphone Chip Integration

- Chip integration is increasing every generation
  - Cell phone size is decreasing
- Users want more features every generation
- Power budget is very limited

Y. Neuvo, ISSCC 2004
Cellphone Architecture Example

- Cellphone chips have multiple processing cores and support multiple applications and features
  - Ex: Integrated Transceiver: WiFi (802.11a/b/g), Bluetooth, FM

www.phonewreck.com, 10
C.H. Van Berkel, DATE 2009
Smart Health Monitoring: Analysis & Delivery

- **Wearable medical monitoring systems**
  - Reliable and seamless monitoring integrated into patients daily life routine

- **Data analysis**
  - Real-time data analysis and diagnosis for efficient healthcare delivery

- **Data delivery**
  - Real time data transmission to healthcare providers (e.g. nurses, primary care physicians, and first responders) through networks and immediate therapy through smart drug delivery
Military & Aerospace Telemedicine

Ultrasound with DARPA-Vuzix Augmented Reality Goggles

Field Satellite Comm. Operator

SATCOM

Gilat GobaLight Mobile VSAT

Wideband Global SATCOM

Local Teleradiology

Combat Support Hospital

Global Teleradiology

WPAN and WLAN

Wireless Integrated Ultrasound System

First Responder (Line Medic)

Point of Injury

Walter Reed National Military Medical Center or Fort Detrick
Key Objectives

- High performance: 10-100 GOPS
- Energy efficiency: 100-1000 GOPS/W
- Area efficiency: 10-100 GOPS/mm$^2$
- Dynamic reconfigurability and task scheduling
- Harness task and data-level parallelism inherent in embedded workloads
FPGA Market

- Very popular in military, space, communication and customer based companies
  - DOD, DOE, NASA, CISCO

<table>
<thead>
<tr>
<th>rank 1999</th>
<th>vendor</th>
<th>global sales (mio $) [IC Insights Inc.]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1998</td>
</tr>
<tr>
<td>1</td>
<td>Xilinx</td>
<td>629</td>
</tr>
<tr>
<td>2</td>
<td>Altera</td>
<td>654</td>
</tr>
<tr>
<td>3</td>
<td>Lattice</td>
<td>206</td>
</tr>
<tr>
<td>4</td>
<td>Actel</td>
<td>154</td>
</tr>
<tr>
<td>5</td>
<td>Lucent</td>
<td>100</td>
</tr>
<tr>
<td>6</td>
<td>Cypress</td>
<td>41</td>
</tr>
<tr>
<td>7</td>
<td>QuickLogic</td>
<td>30</td>
</tr>
<tr>
<td>8</td>
<td>Atmel</td>
<td>32</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>rank 2000</th>
<th>vendor</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Xilinx</td>
<td>42</td>
</tr>
<tr>
<td>2</td>
<td>Altera</td>
<td>37</td>
</tr>
<tr>
<td>3</td>
<td>Lattice</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>Actel</td>
<td>6</td>
</tr>
</tbody>
</table>
Digital Systems

- Electronic circuits that use discrete representations of information
  - Discrete time and values
Digital Signal Processing vs Analog Processing

- DSP arithmetic is completely stable over process, temperature, and voltage variations
  - Ex: $2.0000 + 3.0000 = 5.0000$ will always be true as long as the circuit is functioning correctly

- DSP energy-efficiencies are rapidly increasing

- Once a DSP processor has been designed in a portable format (gate netlist, HDL, software), very little effort is required to “port” (re-target) the design to a different processing technology. Analog circuits typically require a nearly-complete re-design.

- DSP capabilities are rapidly increasing

- Analog A/D speed x resolution product doubles every 5 years

- Digital processing performance doubles every 18-24 Months (6x to 10x every 5 years)
Common DSP Applications

- Early applications
  - Instrumentation
  - Radar
  - Imaging

- Current applications
  - Audio, video
  - Networking
  - Telecommunications
Common Trends

- Analog based → Digital based
  - Music: records, tapes → CDs
  - Video: VHS, 8mm → DVD, Blu-ray
  - Telephony, cell phones: analog (1G) → digital (2G, 3G, 4G, …)
  - Television: NTSC → digital (DVB, ATSC, ISDB, …)
  - Many new things use digital data and “speak” digital: computers, networks, digital appliances
Basic Digital Circuit Components

- Primitive components for logic design

- AND gate
- OR gate
- inverter
- multiplexer
Sequential Circuits

- Circuit whose output values depend on current \textit{and previous} input values
  - Include some form of storage of values

- Nearly all digital systems are sequential
  - Mixture of gates and storage components
  - Combinational parts transform inputs and stored values
Flipflops and Clocks

- Edge-triggered D-flipflop
  - stores one bit of information at a time

- Timing diagram
  - Graph of signal values versus time
Hierarchical Design

Design → Functional Verification

Architecture Design → Unit Design → Unit Verification

Integration Verification → OK?

N → Y → OK?

N → Y
What we learn by the end of semester

- Processor building blocks
  - Binary number representations
  - Types of Adders
  - Multipliers
  - Complex arithmetic hardware
  - Memories

- Communication algorithms and systems

- Design optimization targeted for FPGA
  - Verilog synthesis to a gate netlist
  - Delay estimation and reduction
  - Area estimation and reduction
  - Power estimation and reduction
  - FPGA implementation and testing
A Simple Design Methodology

Requirements and Constraints

Design

Functional Verification

OK? Y N

Synthesize

Post-synthesis Verification

OK? Y N

Physical Implementation

Physical Verification

OK? Y N

Manufacture

Test

Digital Design — Chapter 1 — Introduction and Methodology
Hierarchical Design

- Circuits are too complex for us to design all the detail at once
- Design subsystems for simple functions
- Compose subsystems to form the system
  - Treating subcircuits as “black box” components
  - Verify independently, then verify the composition
- Top-down/bottom-up design
Synthesis

- We usually design using register-transfer-level (RTL) Verilog
  - Higher level of abstraction than gates
- Synthesis tool translates to a circuit of gates that performs the same function
- Specify to the tool
  - the target implementation fabric
  - constraints on timing, area, etc.
- Post-synthesis verification
  - synthesized circuit meets constraints
Physical Implementation

- Implementation fabrics
  - Application-specific ICs (ASICs)
  - Field-programmable gate arrays (FPGAs)
- Floor-planning: arranging the subsystems
- Placement: arranging the gates within subsystems
- Routing: joining the gates with wires
- Physical verification
  - physical circuit still meets constraints
  - use better estimates of delays
Codesign Methodology

```
Codesign Methodology

OK?

Partitioning

Hardware Requirements and Constraints

Hardware Design and Verification

Software Requirements and Constraints

Software Design and Verification

OK? N

Manufacture and Test

Digital Design — Chapter 1 — Introduction and Methodology
```
Summary

- Digital systems use discrete (binary) representations of information
- Basic components: gates and flipflops
- Combinational and sequential circuits
- Real-world constraints
  - logic levels, loads, timing, area, etc
- Verilog models: structural, behavioral
- Design methodology
Integrated Circuits (ICs)

- Circuits formed on surface of silicon wafer
  - Minimum feature size reduced in each technology generation
  - Currently 90nm, 65nm
  - Moore’s Law: increasing transistor count
  - CMOS: complementary MOSFET circuits
Logic Levels

- Actual voltages for “low” and “high”
  - Example: 1.4V threshold for inputs
Logic Levels

- TTL logic levels with noise margins

\[ V_{OL}: \text{output low voltage} \quad V_{IL}: \text{input low voltage} \]
\[ V_{OH}: \text{output high voltage} \quad V_{IH}: \text{input high voltage} \]
Static Load and Fanout

- Current flowing into or out of an output
  - High: SW1 closed, SW0 open
    - Voltage drop across R1
    - Too much current: $V_O < V_{OH}$
  - Low: SW0 closed, SW1 open
    - Voltage drop across R0
    - Too much current: $V_O > V_{OL}$
  - Fanout: number of inputs connected to an output
    - determines static load
Capacitive Load and Prop Delay

- Inputs and wires act as capacitors

- tr: rise time
- tf: fall time
- tpd: propagation delay
  - delay from input transition to output transition
Other Constraints

- Wire delay: delay for transition to traverse interconnecting wire
- Flipflop timing
  - delay from clk edge to Q output
  - D stable before and after clk edge
- Power
  - current through resistance => heat
  - must be dissipated, or circuit cooks!
Area and Packaging

- Circuits implemented on silicon chips
  - Larger circuit area => greater cost

- Chips in packages with connecting wires
  - More wires => greater cost
  - Package dissipates heat

- Packages interconnected on a printed circuit board (PCB)
  - Size, shape, cooling, etc, constrained by final product
Models

- Abstract representations of aspects of a system being designed
  - Allow us to analyze the system before building it

Example: Ohm’s Law

- \( V = I \times R \)
  - Represents electrical aspects of a resistor
  - Expressed as a mathematical equation
  - Ignores thermal, mechanical, materials aspects
Verilog

- Hardware Description Language
  - A computer language for modeling behavior and structure of digital systems

- Electronic Design Automation (EDA) using Verilog
  - Design entry: alternative to schematics
  - Verification: simulation, proof of properties
  - Synthesis: automatic generation of circuits
Module Ports

- Describe input and outputs of a circuit

![Diagram of a module with ports and logic gates]

Digital Design — Chapter 1 — Introduction and Methodology
module vat_buzzer_struct
(
  output buzzer,
  input above_25_0, above_30_0, low_level_0,
  input above_25_1, above_30_1, low_level_1,
  input select_vat_1);

wire below_25_0, temp_bad_0, wake_up_0;
wire below_25_1, temp_bad_1, wake_up_1;

// components for vat 0
not inv_0 (below_25_0, above_25_0);
or or_0a (temp_bad_0, above_30_0, below_25_0);
or or_0b (wake_up_0, temp_bad_0, low_level_0);

// components for vat 1
not inv_1 (below_25_1, above_25_1);
or or_1a (temp_bad_1, above_30_1, below_25_1);
or or_1b (wake_up_1, temp_bad_1, low_level_1);

mux2 select_mux (buzzer, select_vat_1, wake_up_0, wake_up_1);
endmodule
module vat_buzzer_struct 
  ( output buzzer, 
    input above_25_0, above_30_0, low_level_0, 
    input above_25_1, above_30_1, low_level_1, 
    input select_vat_1 );

assign buzzer =
  select_vat_1 ? low_level_1 | (above_30_1 | ~above_25_1) :
    low_level_0 | (above_30_0 | ~above_25_0);
endmodule
Design Methodology

- Simple systems can be designed by one person using *ad hoc* methods.
- Real-world systems are designed by teams.
  - Require a systematic design methodology.

**Specifies**
- Tasks to be undertaken.
- Information needed and produced.
- Relationships between tasks.
  - Dependencies, sequences.
- EDA tools used.
Design using Abstraction

- Circuits contain millions of transistors
  - How can we manage this complexity?

- Abstraction
  - Focus on relevant aspects, ignoring other aspects
  - Don’t break assumptions that allow aspect to be ignored!

- Examples:
  - Transistors are on or off
  - Voltages are low or high
Embedded Systems

- Most real-world digital systems include embedded computers
  - Processor cores, memory, I/O
- Different functional requirements can be implemented
  - by the embedded software
  - by special-purpose attached circuits
- Trade-off among cost, performance, power, etc.
+ General-purpose; cost of the design is amortized across millions of units
+ Billions of transistors
+ High clock rates
- Very long design times;
  long fabrication times
- Essentially unchangeable

Custom designed integrated circuit

+ Programmable “on the fly”
+ Application-specific; cost of design is amortized over hundreds or thousands of units
+ Millions of transistors
+ Highly parallel
- Lower clock rates

Field-programmable gate array (FPGA)

+ Easily changed
+ Well-known and rich development environment
- “Slow” (depends on code-to-processor mapping)

Software on commodity processor

---

for (int i = 0; i < 32; i += 1) {
    uint32 a = (A >> i) & 1;
    uint32 b = (B >> i) & 1;
    uint32 c = (C >> i) & 1;
    uint32 d = (D >> i) & 1;
    uint32 e = (a << 3) | (b << 2) |
               (c << 1) | d;
    R |= (F >> c) & 1;}

45
Fine grain fabrics and operative elements. We may distinguish two kinds of reconfigurable resources: configurable operation elements and reconfigurable interconnect resources. The overall architecture of reconfigurable interconnect is often called "fabrics". The elementary operation units of fine grain reconfigurable platforms usually have single bit path width: gates and flipflops. This explains the use of different terms "FPGA" (field-programmable gate array), "PLD" (programmable logic device), FPL (field-programmable logic), or, CPLD (complex programmable logic device), which indicate, that the programmable elementary units are gate level (logic level) units. From an EDA point of view this level appears as a methodology of hardwired logic design "on a strange platform", which is not really hardwired.