Digital Design: An Embedded Systems Approach Using VHDL

Chapter 1
Introduction and Methodology

Portions of this work are from the book, Digital Design: An Embedded Systems Approach Using VHDL, by Peter J. Ashenden, published by Morgan Kaufmann Publishers, Copyright 2007 Elsevier Inc. All rights reserved.

VHDL

Digital Design:

- Digital: circuits that use two voltage levels to represent information
- Logic: use truth values and logic to analyze circuits
- Design: meeting functional requirements while satisfying constraints
- Constraints: performance, size, power, cost, etc.

Design using Abstraction

- Circuits contain millions of transistors
  - How can we manage this complexity?
- Abstraction
  - Focus on aspects 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

Billions

Transistors are on or off
- Voltages are low or high

VHDL

Xilinx XC6VSX475T
2,016 DSP48E1 Slices
25 × 18 bit Multipliers
48 bit Accumulators

More Trends.....
FPGAs now give the best GFLOPs/Watt performance.

In a National Science Foundation benchmark, a Stratix IV FPGA delivered 171 GFLOPs and was the clear overall leader in highest GFLOPs/Watt \cite{x}.  

\cite{x} Altera's floating-point solutions for military applications.

April 2009, SS-01058-1.1  
http://www.altera.co.uk/literature/po/ss-military-floating-point.pdf

---

**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.

---

**Binary Representation**

- Basic representation for simplest form of information, with only two states
  - a switch: open or closed
  - a light: on or off
  - a microphone: active or muted
  - a logical proposition: false or true
  - a binary (base 2) digit, or bit: 0 or 1

---

**Binary Representation: Example**

- Signal represents the state of the switch
  - high-voltage => pressed,
  - low-voltage => not pressed
- Equally, it represents state of the lamp
  - lamp_lit = switch_pressed

---

**Binary Representation: Example**

- dark: it's night time
- lamp_enabled: turn on lamp at night
- lamp_lit: lamp_enabled AND dark
- Logically: day time => NOT lamp_lit
Basic Gate Components
- Primitive components for logic design
  - AND gate
  - OR gate
  - inverter
  - multiplexer

Combinational Circuits
- Circuit whose output values depend purely on current input values

Sequential Circuits
- Circuit whose output values depend on current 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

Real-World Circuits
- Assumptions behind digital abstraction
  - Ideal circuits, only two voltages, instantaneous transitions, no delay
  - Greatly simplify functional design
  - Constraints arise from real components and real-world physics
  - Meeting constraints ensures circuits are “ideal enough” to support abstractions

Integrated Circuits (ICs)
- Circuits formed on surface of silicon wafer
  - Minimum feature size reduced in each technology generation
  - Currently 90nm, 65nm, 45nm, 32nm (Intel Westmere)
  - Moore’s Law: increasing transistor count
  - CMOS: complementary MOSFET circuits
Logic Levels

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

Static Load and Fanout

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

Capacitive Load and Prop Delay

- Inputs and wires act as capacitors
  - $t_r$: rise time
  - $t_f$: fall time
  - $t_{pd}$: 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

VHDL

- **VHSIC Hardware Description Language**
  - A computer language for modeling behavior and structure of digital systems
- **Electronic Design Automation (EDA)** using VHDL
  - Design entry: alternative to schematics
  - Verification: simulation, proof of properties
  - Synthesis: automatic generation of circuits

VHDL 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

Entity Declarations

```
library ieee; use ieee.std_logic_1164.all;
entity vat_buzzer is
  port ( above_25_0, above_30_0, low_level_0 : in std_logic;
     above_25_1, above_30_1, low_level_1 : in std_logic;
     select_vat_1 : in std_logic;
     buzzer : out std_logic );
end entity vat_buzzer;
```

Structural Architectures

```
library dld; use dld.gates.all;
architecture struct of vat_buzzer is
  signal below_25_0, temp_bad_0, wake_up_0 : std_logic;
  signal below_25_1, temp_bad_1, wake_up_1 : std_logic;
begin
  -- components for vat 0
  inv_0 : inv (above_25_0, below_25_0);
  or_0a : or2 (above_30_0, below_25_0, temp_bad_0);
  or_0b : or2 (temp_bad_0, low_level_0, wake_up_0);
  -- components for vat 1
  inv_1 : inv (above_25_1, below_25_1);
  or_1a : or2 (above_30_1, below_25_1, temp_bad_1);
  or_1b : or2 (temp_bad_1, low_level_1, wake_up_1);
  select_mux : mux2 (wake_up_0, wake_up_1, select_vat_1, buzzer);
end architecture struct;
```
**Behavioral Architectures**

```vhdl
architecture behavior of vat_buzzer is
begin
  buzzer <=
    low_level_1 or
    (above_30_1 or not above_25_1)
    when select_vat_1 = '1' else
    low_level_0 or
    (above_30_0 or not above_25_0);
end architecture behavior;
```

**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

**A Simple Design Methodology**

1. **Requirements and Constraints**
   - VHDL is used to describe the system
   - Synthesizer translates the system to physical implementation
2. **Design**
   - Functional Verification
   - Post-synthesis Verification
   - Physical Implementation
   - Manufacture
3. **Synthesize**
   - Physical Implementation
   - Test
4. **Manufacture**
   - Synthesized circuit meets constraints

**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

**Hierarchical Design**

1. **Functional Verification**
   - OK?
2. **Unit Design**
   - OK?
3. **Integration Verification**
   - OK?
4. **Architecture Design**
5. **Synthesis**
   - We usually design using register-transfer-level (RTL) VHDL
     - 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**

![Diagram of codesign 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
- VHDL models: structural, behavioral
- Design methodology