EEL 4783: HDL in Digital System Design

Lecture 1: Introduction*

Prof. Mingjie Lin

Overview

- What is an digital embedded system?
- Why HDL (Hardware Description Language)?
- Why take this course?
- Class mechanics
  - Administrative issues
  - Lecture topics
  - Assignment and projects

What is an Digital Embedded System?

- ES from 10,000 Feet Above
  - a computer system designed for specific control functions within a larger system
  - often with real-time computing constraints
  - embedded as part of a complete device often including hardware and mechanical parts

- By contrast, a general-purpose computer
  - designed to be flexible and to meet a wide range of end-user needs

Embedded Computing Systems

- Obvious examples:
  - HDTV
  - Washing Machines
  - Microwave
  - Controllers for other household devices such as A/C
  - Digital watches
  - MP3 players
- Not-so-obvious examples:
  - Automobiles
  - Avionics / Flight control
  - Nuclear Power Plants
  - Medical devices

In-Depth Example: Mobile phones

- Multiprocessor
  - 8-bit/32-bitforUI
  - DSP for signals
  - 32-bit in IR port
  - 32-bit in Bluetooth
- 8-100 MB of memory
- All custom chips
- Power consumption & battery life depends on software

But …

- Dual-core A5 chip
  - package on package (PoP) system-on-a-chip (SoC)
  - 45 nm Dual core GPU PowerVR SGX543MP2 clocked at 200 MHz
- 8MP camera and optics
- IOS 5 and iCloud
- Siri

In-Depth Example: Cars

- Multiple processors
  - Upto100
  - Networked
- Multiple networks
  - Body
  - Engine
  - Telematics
  - Media
  - Safety

Cars

- Function diversity
  - ABS: Anti-lock braking systems
  - Airbags
  - Efficient automatic gearboxes
  - Theft prevention with smart keys
  - Blind-angle alert systems

- Device diversity
  - 8-bit – door locks, lights, etc.
  - 16-bit – most functions
  - 32-bit – engine control, airbags

Little-Known Facts about Cars

- Car electronics is an increasingly important market, requiring new design flows
  - Software is important for value addition
- Comments by major manufacturers
  - Daimler Chrysler: More than 90% of the innovation is from the car electronics (and not from the mechanical parts!)
  - BMW: More than 30% of the manufacturing cost of a car is from the electronic components !
- Reliable/robust ES design flows needed !

Hardware Description Language

This Lecture - HDL

- What and why HDL
- Verilog HDL
- Modelling a simple circuit.
  - Delays
  - Stimulus
- Abstraction Levels
  - Gates
  - Dataflow
  - Proceedural

Hardware Description Language (HDL)

- Basic idea is a programming language to describe hardware
- Initial purpose was to allow abstract design and simulation
  - Design could be verified then implemented in hardware
- Now Synthesis tools allow direct implementation from HDL code.
  - Large improvement in designer productivity

HDL

- HDL allows write-run-debug cycle for hardware development.
  - Similar to programming software
  - Much, much faster than design-implement-debug
- Combined with modern Field Programmable Gate Array chips large complex circuits (100000s of gates) can be implemented.

HDLs

- There are many different HDLs
  - Verilog HDL
  - ABEL
  - VHDL
- Verilog is the most common in US industry
- VHDL is the most common in Europe
  - Large standard developed by US DoD
  - VHDL = VHSIC HDL
  - VHSIC = Very High Speed Integrated Circuit

Verilog HDL

- Verilog HDL is the most common in IC industry
  - Easier to use in many ways = better for teaching
  - C - like syntax
- History
  - Developed as proprietry language in 1985
  - Opened as public domain spec in 1990
    - Due to losing market share to VHDL
  - Became IEEE standard in 1995

Verilog HDL

- Verilog constructs are use defined *keywords*
  - Examples: and, or, wire, input output
- One important construct is the *module*
  - Modules have inputs and outputs
  - Modules can be built up of Verilog primatives or of user defined submodules.

Example: Simple Circuit Diagram

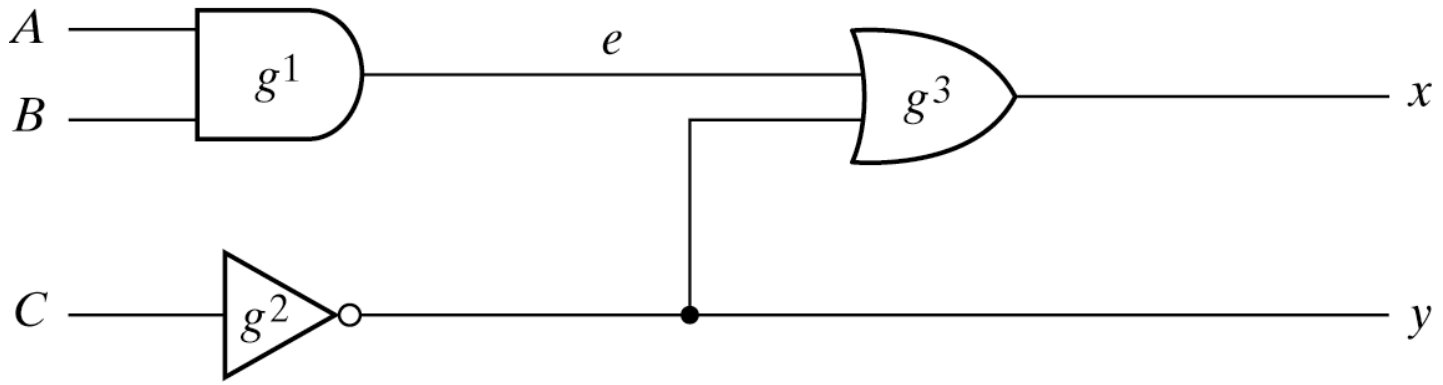

Fig. 3-37 Circuit to Demonstrate HDL

All examples and code from Mano “Digital Design” 3rd Ed.

Example: Simple Circuit HDL

```
module smpl_circuit(A,B,C,x,y);
   input A,B,C;
   output x,y;
   wire e;
   and g1(e,A,B);
   not g2(y, C);
   or  g3(x,e,y);
endmodule
```

- The module starts with **`module`** keyword and finishes with **`endmodule`**.
- Internal signals are named with **`wire`**.
- Comments follow **`//`**
- **`input`** and **`output`** are ports. These are placed at the start of the module definition.
- Each statement ends with a semicolon, except **`endmodule`**.

Circuit to code

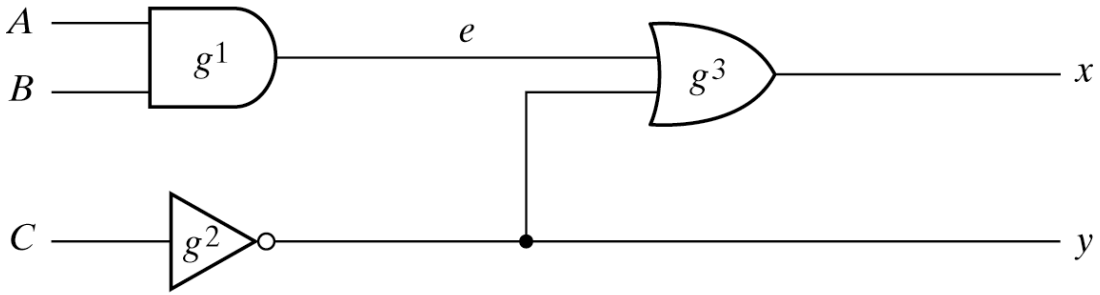

Fig. 3-37 Circuit to Demonstrate HDL

```
module smpl_circuit(A,B,C,x,y);
   input A,B,C;
   output x,y;
   wire e;
   and g1(e,A,B);
   not g2(y, C);
   or  g3(x,e,y);
endmodule
```

Adding Delays

- To simulate a circuits real world behaviour it is important that propogation delays are included.
- The units of time for the simulation can be specified with **`timescale.`**
  - Default is 1ns with precision of 100ps
- Component delays are specified as #(delay)

Simple Circuit with Delay

```
module circuit_with_delay (A,B,C,x,y);
   input A,B,C;
   output x,y;
   wire e;
   and #(30) g1(e,A,B);
   or  #(20) g3(x,e,y);
   not #(10) g2(y,C);
endmodule
```

Input signals

- In order to simulate a circuit the input signals need to be known so as to generate an output signal.
- The input signals are often called the circuit *stimulus.*
- An HDL module is written to provide the circuit stimulus. This is known as a *testbench.*

Testbench

- The *testbench* module includes the module to be tested.
- There are no input or output ports for the testbench.
- The inputs to the test circuit are defined with **`reg`** and the outputs with **`wire`**.
- The input values are specified with the keyword **`initial`**
- A sequence of values can be specified between **`begin`** and **`end`**.

Signal Notation

- In Verilog signals are generalised to support multi-bit values (e.g. for buses)
  - The notation

```
A =  1'b0;
```

  - means signal A is one bit with value zero.
- The end of the simulation is specified with **`$finish.`**

Stimulus module for simple circuit

```
module stimcrct;
reg A,B,C;
wire x,y;
circuit_with_delay cwd(A,B,C,x,y);
initial
   begin
        A = 1'b0; B = 1'b0; C = 1'b0;
     #100
        A = 1'b1; B = 1'b1; C = 1'b1;
     #100  $finish;
   end
endmodule
```

Timing Diagram

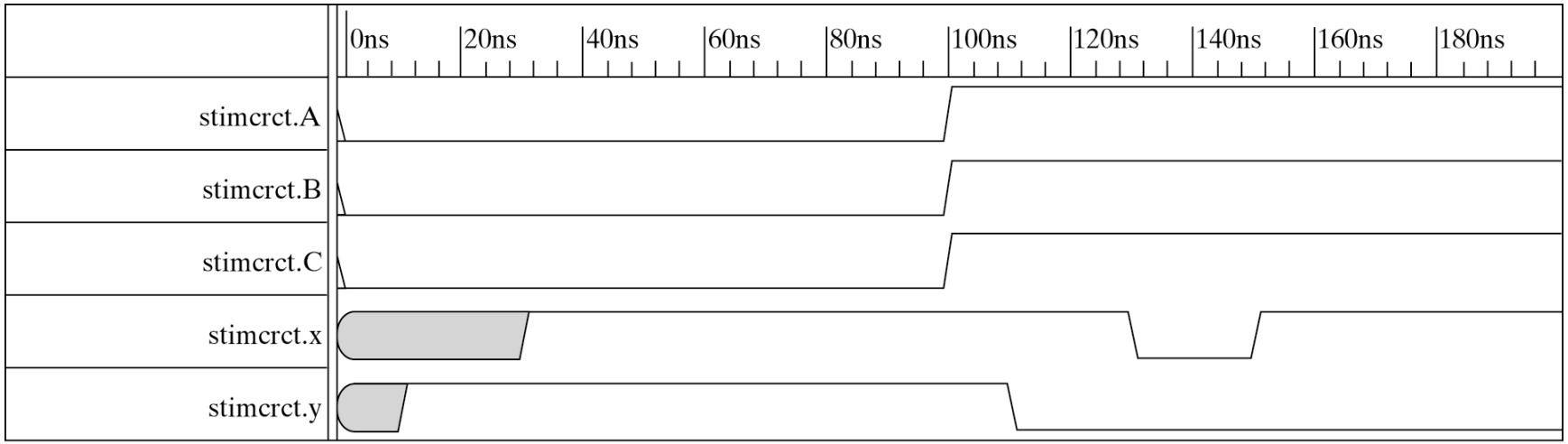

Fig. 3-38 Simulation Output of HDL Example 3-3

Gate Level Modelling

- The simple circuit used so far is an example of *gate-level modelling.*
- The module is a text description of the circuit layout.
- Verilog has all the standard gates
  - `and, nand`
  - `or,  nor`
  - `xor, xnor`
  - `not, buf`

Dataflow modelling

- Another level of abstraction is to model dataflow.
- In dataflow models, signals are continuously assigned values using the **`assign`** keyword.
- **`assign`** can be used with Boolean expressions.
  - Verilog uses **`&`** (and), **`|`** (or), **`^`** (xor) and **`~`** (not)
- Logic expressions and binary arithmetic are also possible.

Simple Circuit Boolean Expression

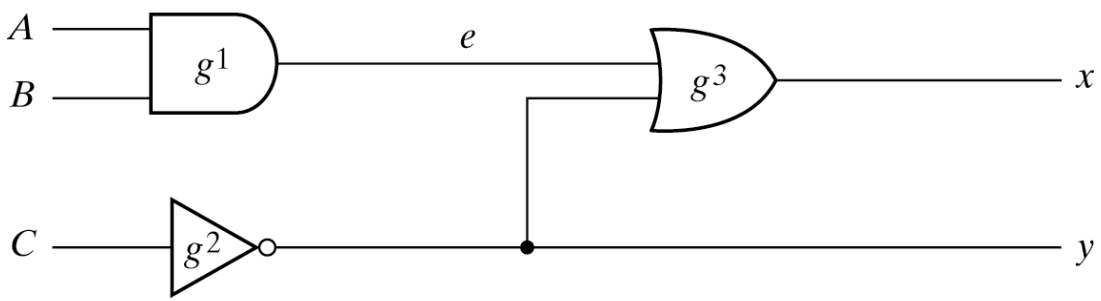

Fig. 3-37 Circuit to Demonstrate HDL

```
//Circuit specified with Boolean equations
module circuit_bln (x,y,A,B,C);
   input A,B,C;
   output x,y;
   assign x = A | (B & ~C);
   assign y = ~C ;
endmodule
```

Behavioural Modelling

- Represents circuits at functional and algorithmic level.
- Use proceedural statements similar in concept to proceedural programming languages (e.g. C, Java),
- Behavioural modelling is mostly used to represent sequential circuits.

Behavioural Modelling

- Behavioural models place proceedural statements in a block after the **`always`** keyword.
- The **`always`** keyword takes a list of variables. The block of statements is executed whenever one of the variables changes.
- The target variables are of type **`reg`**. This type retains its value until a new value is assigned.

Behavioral description of 2-input mux

```
module mux2x1_bh(A,B,select,OUT);
    input A,B,select;
    output OUT;
    reg OUT;
    always @ (select or A or B)
          if (select == 1) OUT = A;
          else OUT = B;
endmodule
```

HDL Summary

- Hardware Description Languages allow fast design and verification of digital circuits.
- Accurate simulation and testing requires delays and inputs to be specified.
- There are three different levels of abstraction for modelling circuits.

Why This Course?

- Because it is FUN intellectually!
- Because HDL prog. become increasingly more critical


Aerospace/Defense


Industrial/Scientific/Medical


Broadcast


Wired communication


Consumer


Wireless communication


HPC and storage


Automobile

Class goal

- Learn about basic concept and techniques of hardware/sotware co-design, …

- Hands-on class projects
  - Complete FPGA design flow to implement a “real” embedded computing system
  - Improve your HDL programming skills
  - Improve your Software programming skill
  - Learning by doing

Administrative issues

- Fill out the student info sheet
  - Name, status, reason of taking this class, expectations, prior knowledge, …
- Pre-requisites
  - EEL 3342: Digital Logic Design
  - Course self-contained, but logic design and computer architecture knowledge helpful (EEL 4768: Computer Architecture)
  - Willingness to work hard
- Information distribution
  - Webpage: www.eecs.ucf.edu/~mingjie/EEL4783_2012/index.html

Lecture schedule

See Webpage:

www.eecs.ucf.edu/~mingjie/EEL4783_2012

Final issues

- Please fill out the student info sheet before leaving
- Come by my office hours (right after class)
- Any questions or concerns?