

# EE 4755—Digital Design Using Hardware Description Languages

## Midterm Exam Review

### When / Where

Friday, 27 October 2023, 11:30-12:20 CDT

### Conditions

Bring one sheet of notes (can use both sides), 216 mm  $\times$  280 mm.

### Format

Several problems, short-answer questions.

## Resources

Lecture slides and examples used in class: <https://www.ece.lsu.edu/koppel/v/ln.html>

References for Verilog and logic design.

Problem Collections: <https://www.ece.lsu.edu/koppel/v/guides.html>

Solved tests and homework: <https://www.ece.lsu.edu/koppel/v/prev.html>

## Topics for Exam

Everything before sequential logic.

Material in lecture slides and homework.

## Study Recommendations

Study this semester's homework assignments. Similar problems may appear on the exam.

Study previous semesters' homework.

Solve Old Problems—memorizing solutions **is not the same** as solving.

Following and understanding solutions **is not the same as** solving.

Use the solutions for brief hints and to check your own solutions.

## Course Material Areas

### *Verilog*

The System Verilog language, including structural and behavioral code.

### *Synthesis*

How hardware is inferred, mapped, and optimized from Verilog.

### *Digital Design*

The functioning of the circuits covered in class.

How to design digital circuits.

### *Tools*

Understand what simulation and synthesis tools do.

# Verilog Topics

## Objects

See <https://www.ece.lsu.edu/v/2023/1020-types.v.html>.

Object Types: *variable* v *net* objects.

Key difference:...

... variables are assigned, nets are driven (connected to something).

## Data Types

Four-State Integer Types

Two-State Integer Types

Floating-Point Types

String Type

## Integer Data Types

Four-State Integer Types: `logic`, `integer`, `time`.

Two-State Integer Types: `int`, `bit`, `byte`, `shortint`, `longint`.

Integer qualifiers: `signed`, `unsigned`.

## Real Data Types

Real Types: `real`, `shortreal`.

Type punning: `$realtobits`, `$bitstoreal`, etc..



## Arrays

See <https://www.ece.lsu.edu/v/2023/1020-types.v.html>.

Packed v. Unpacked Arrays

```
uwire [7:0] e_pluribus_unum;    // Packed
uwire plain_array [7:0];        // Unpacked
```

Element and bit numbering:

```
uwire [7:0] color;             // Bit 0 is LSB.
uwire [0:7] colour;            // Bit 0 is MSB.
```

Static, Dynamic, and Associative arrays.

Used in testbenches and other non-synthesizable code.

```
logic [7:0] array1[];          // Dynamically sized.
logic [7:0] array2[string];    // Associative. (string is a data type)

initial begin
    array1 = new [ num_elts ];
    array1[ 5 ] = 123; // Hope that num_elts >= 6.
    array2[ "5" ] = 1;
    array2[ "five" ] = 2;
    array2[ "six" ] = 3;
end
```

## Modules

Port and parameter declaration.

Module and primitive instantiation.

Object declarations.

Continuous `assign`.

Procedural code.

Generate statements.

## Procedural Code

Execution of `initial`, `always`, and `always_comb`.

Delays (*e.g.*, `#5`).

Event controls (*e.g.*, `@( posedge clk)`).

## Elaboration and Generate Statements

<https://www.ece.lsu.edu/v/2023/1025-gen-elab.v.html>

Please Pay Attention

Make sure you really understand the differences listed below, especially for generate statements.

Elaboration-time constants.

Difference between a module parameter and a port.

Generate Statements

Difference between generate `if` and procedural `if`.

Difference between generate `for` and procedural `for`.

## Emphases, Key Skills

### Verilog—Key Skills

Given a design in one form, write design in another:

Explicit Structural

Implicit Structural

Synthesizable Behavioral

Logic Diagram

Use generate statements to interconnect modules.

Use generate statements in recursive construction of trees.

## Synthesis Key Skills

Given Verilog code:

Show inferred hardware (before optimization).

Show expected optimizations.

## Logic Design Skills

Given a design, be able to:

Compute Cost

Compute Delay



# Synthesis Topics

## Synthesis Topics

Understand what is done during inference, optimization, technology mapping.

<https://www.ece.lsu.edu/v/2023/1014-syn-general.v.html>.

Inference of combinational logic.

<https://www.ece.lsu.edu/v/2023/1015-syn-comb-str.v.html>

<https://www.ece.lsu.edu/v/2023/1045-syn-comb-behav.v.html>

Optimization of combinational logic.

# Digital Design Topics

## Digital Design Topics

### Common Components

Multiplexor

Binary Full Adder, Ripple Adder

Integer Equality and Magnitude Comparison

### Common Component Skills

Show how to implement using basic gates.

Know how to optimize for special cases (a constant input, etc.).

## Cost and Delay Estimation

### Simple Cost Model

Cost of  $n$ -input AND and OR gates are  $[n - 1] u_c$ .

Inverters (NOT gates) are free!

Application to linear and tree structures.

## Simple Delay Model

**Important:** Delay is based on **critical path**.

Delays

Delay of  $n$ -input AND and OR gates are  $\lceil \lg n \rceil u_t$ .

Inverters (NOT gates) have zero delay!