CME 341: Introduction to Verilog

Brian Berscheid

Department of Electrical and Computer Engineering
University of Saskatchewan

Today's agenda

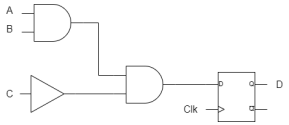
1. What is Verilog and why is it of interest?
2. What is an FPGA and why is it of interest?
3. Verilog Basics: Modularity

What is Verilog and why is it of interest?

Prerequisite review: EE 232 (Digital Electronics)

I hope / expect you learned the following:

- Basics of logic gates (AND, OR, NOT, etc.)
- Boolean logic, hand optimization strategies (algebraic, K-maps)
- Synchronous logic (flip-flops / registers)
- Problem statement $\leftrightarrow$ truth tables $\leftrightarrow$ logic circuit by hand



Problem: The 'by-hand' approach doesn't scale!

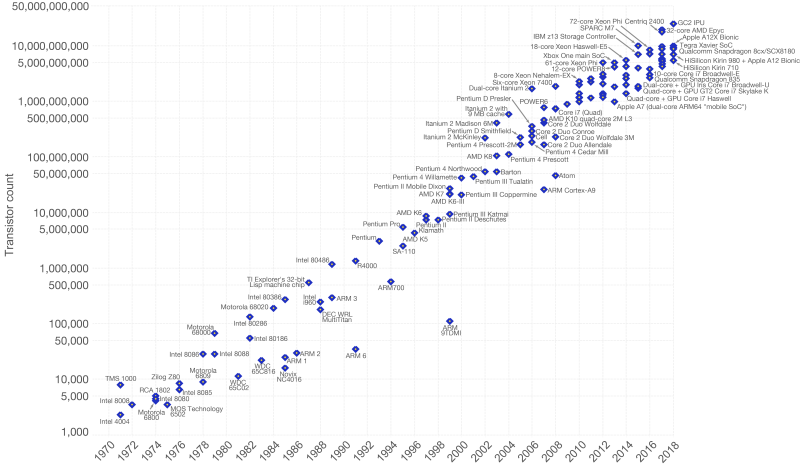
- Beyond about 4-6 variables, constructing the circuit by hand is too slow and error prone.
- Moore's law:
  - ▷ Transistor count in ICs doubles roughtly every two years
  - ▷ Digital logic capabilities increase and costs decrease accordingly
  - ▷ Has held true since 1970s; source of huge societal transformations
- Corollary: designing digital logic circuits becomes more challenging as time goes on!

If G.M. had kept up with technology like the computer industry has, we would all be driving twenty-five dollar cars that got one-thousand miles to the gallon.

– Bill Gates

Moore's Law – The number of transistors on integrated circuit chips (1971-2018)

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important as other aspects of technological progress – such as processing speed or the price of electronic products – are linked to Moore's law.



Data source: Wikipedia (https://en.wikipedia.org/wiki/Transistor_count)
The data visualization is available at OurWorldinData.org. There you find more visualizations and research on this topic.

A solution: Hardware Description Languages: (HDLs)

- Humans tend to work more effectively at higher levels of abstraction
- Computers are great at low level optimization
- Define a text-based language that can describe a hardware circuit (structure or behaviour) which can be compiled into a circuit by a computer
- Parallel to software development: C/Java/Python/etc vs assembly languages
  - ▷ Speed up the development process
  - ▷ Reduce chance of simple mistakes
  - ▷ Easier to understand, review, and debug

Verilog vs VHDL

- Numerous HDLs exist; by far the most common are:
  - ▷ Verilog HDL
  - ▷ VHDL

- Note that Verilog HDL and VHDL are different languages!

- This class will focus on Verilog HDL
  - ▷ Note that many versions of the Verilog language exist (Verilog-95, Verilog-2001, etc). Newer versions are known as SystemVerilog.
  - ▷ In CME 341, we will mainly be learning the basics of Verilog, but may touch on a few SystemVerilog features.

The bottom line...

Verilog is critical to the development of digital logic circuits!

There is strong demand for engineers with Verilog skills these days!

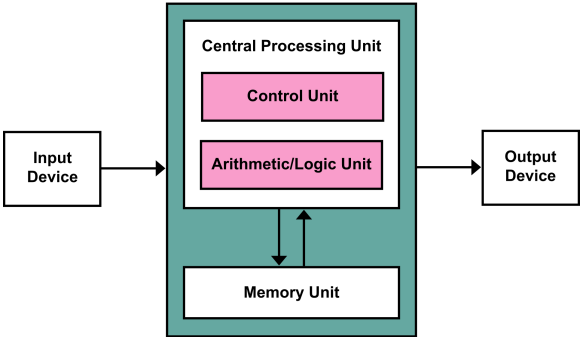
What is an FPGA and why is it of interest?

Common platforms for digital logic

- General purpose processor
- Application specific integrated circuit (ASIC)
- Field programmable gate array (FPGA)

General purpose processor

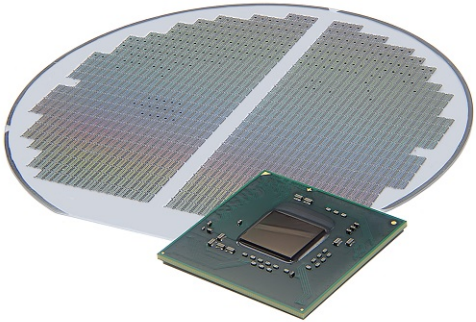
- Fixed, general purpose hardware structure
  - ▷ Program instructions select operations to perform at runtime
  - ▷ Single IC can perform any calculation; economies of scale
  - ▷ Fixed hardware structure can't be optimized for a particular application
- Some computations may be slow or inefficient! (ex. 128-bit mult w/ 64-bit ALU)
- Purely serial operation: one instruction at a time



https://en.wikipedia.org/wiki/Von_Neumann_architecture

ASIC

- Purely custom designed IC
- Includes exactly the circuitry required for the job at hand (no more, no less)
- Can construct as much parallelism as desired
- Manufacturing is very complex and involves a huge upfront investment → only feasible for high-volume products

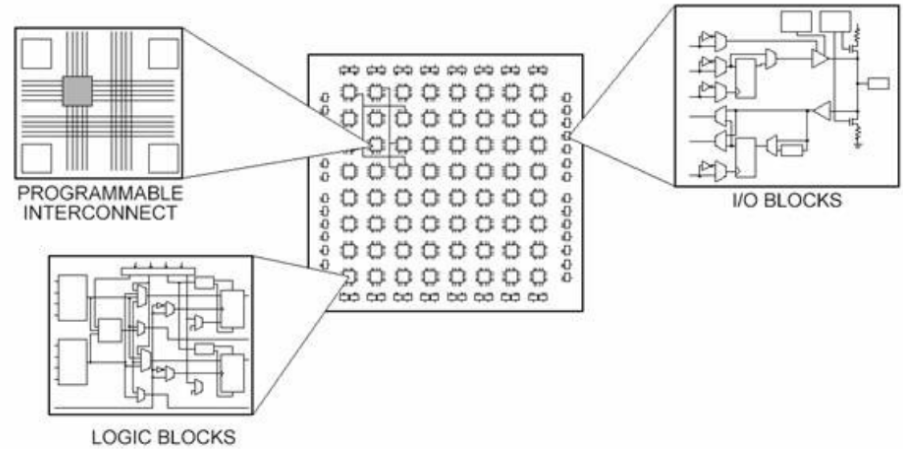


https://anysilicon.com/what-is-an-asic-and-how-it-is-being-made/

FPGA

- Simple model: essentially a huge grid of configurable logic blocks and interconnects
- Elements can be connected together at runtime through a configuration file to choose the "hardware structure"
- Combines advantages of general purpose processor and ASIC
  - ▷ Single generic hardware structure that is suited to many/all applications
  - ▷ Can optimize runtime logic circuit to current application (parallelism, data sizes, etc).
- Downside: generally more expensive per unit than ASIC and has higher power consumption

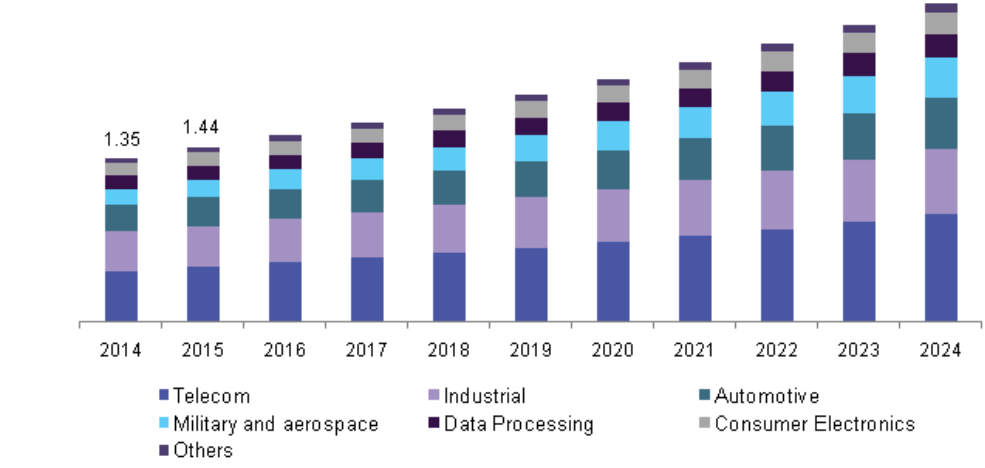
FPGA block diagram



https://www.ni.com/en-ca/innovations/white-papers/08/fpga-fundamentals.html

FPGA trends and applications

US FPGA market by application (USD billion)



https://medium.com/ict-market-research-reports/field-programmable-gate-array-fpga-market-717e6681a70

The bottom line...

FPGAs skills are in demand in the job market!

This class will teach you the basics!

Verilog Basics: Modularity

Modularity

- Verilog circuits are designed in a hierarchical way
- Each element in the hierarchy is known as a "module"
- All Verilog code resides within a module
- Modules typically contain inputs and/or outputs, with logic circuitry in between
- Modules can be instantiated within one another
- The highest level of hierarchy in circuit is known as the "top-level" module. Its inputs and outputs connect to pins on the FPGA.

Key Elements of a Module

- Starts with module declaration and port list
- Module needs to have a unique name
- Ends with the endmodule statement
- One module per file - filename should be module_name.sv
  - ▷ This file is sometimes called the "prototype"

```
module module_name ( port_list );

// Module code goes here...

endmodule    // Note: no semicolon
```

Module Port Lists

- Two main styles exist: Verilog-95-style and Verilog-2001-style
  - ▷ In Verilog-95-style, only list the signal names in the port list
    MUST declare direction (and optionally type) of ports inside main module
  - ▷ In Verilog-2001-style, include signal name, direction, and type in port list

Verilog-95-style

```
module abc (port_a, port_b);

input port_a;
output port_b;

wire port_a;
wire port_b;
...
endmodule
```

Verilog-2001-style

```
module abc (
  input  wire  port_a,
  output wire  port_b);
...
endmodule
```

Module Instantiations

- A module can be instantiated inside another module
- The instantiated module needs a unique instance name
- The module prototype is like a blueprint or template that is used by Verilog to create objects (instances)
- Need to connect signals (generally wires) to the ports of the instance
  - ▷ Signals should to be declared before they are connected
  - ▷ Much more on signal types later...
- The hierarchy can be nested as deeply or as shallowly as you like

Instantiation Example

Can you sketch a block diagram corresponding to this code?

```
module my_top_module ();
// This module has no input or output ports

wire port_a_wire;  // can give these any name you choose
wire port_b_wire;

abc abc_inst (
  .port_a (port_a_wire),
  .port_b (port_b_wire)
);

endmodule
```

Thank you!
Have a great day!