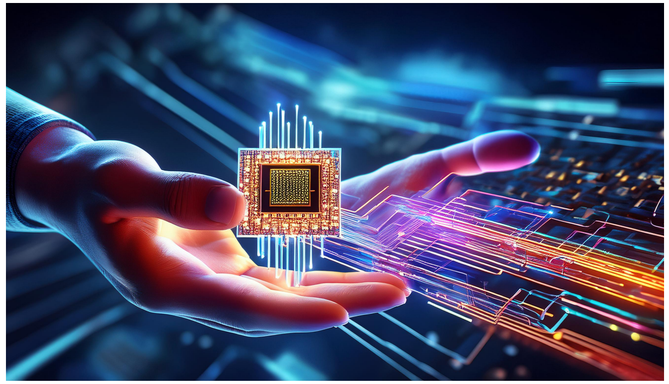
1/5/26 09:54

CSCI 250
Introduction to Computer Organisation
Lecture 1: Beyond Integer Arithmetics IV



Jetic Gū

Overview

- Focus: Course Introduction
- Architecture: Logical Circuits
- Textbook: LW Chapter 7
- Core Ideas:

1. More Verilog: Concurrent Statements

Why FPGA?

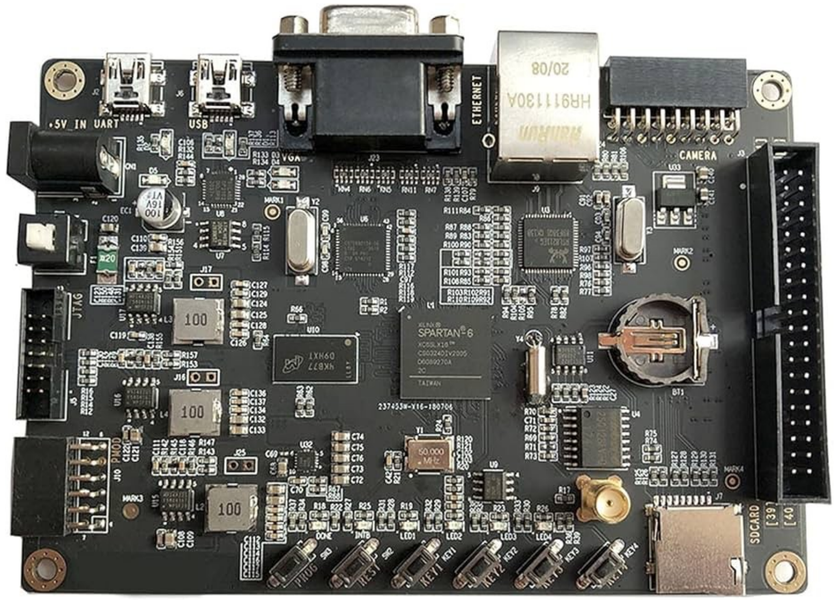
So far, in VHDL

- You've used signal to declare new variables
- You've use logical/arithmetics similar to Register Transferring Operations
- You've built components in Verilog
- Let's provide you with more detailed explanation of how Verilog works

Review

FPGA

- Field Programmable Gate Arrays
- Embedded system (development)
- Programmable logical circuit
  - Contains Arrays of Logical gates (a lot of LUTs, like truth tables)
  - Logical gates are connected with reconfigurable routing networks (LUTs)
  - Other components: RAM, etc…



Concept

Why FPGA

- Real circuits with physical gates are GOOD
- But you need to make a new one every time you change your design, that's very very very expensive
- Impractical for quick implementation and testing
- FPGAs are more expensive, but reconfigurable in the field (hence the name), with minimal punishment in performance (delays, heat, energy consumption, etc.)

1. Here's a youtube clip: https://www.youtube.com/watch?v=G2uYU7aD698

Concept

Use cases: Parallel Computing

- Computer CPUs are traditionally NOT parallel
  - Solution1: **more cores**
    You can only have so many cores (currently hundreds in a single chip)
    Con: expensive, wasteful, energy hungry, hard to maintain
  - Solution 2: **coprocessors**
    For multi-media, you have decoders; for rendering and matrix arithmetics, you have GPUs (thousands of arithmetic units in a single chip)
    Con: very very application specific
  - Solution 3: **FPGA**
    Programmable, highly parallel, as sophisticated as CPUs yet reconfigurable
    Con: not a lot of people knows how to use it?

P1
FPGA

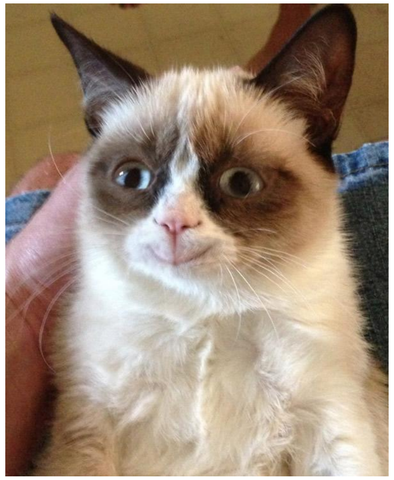
Use cases: Hardware Emulation

- Everything dies, including microchips
  - Vintage computers and gaming consoles
  - ECUs in cars
  - Controllers for hardware instruments
- Some of these can be **remanufactured**, but that's very expensive
- Some of these can be **emulated**, but you'll take a hit on performance and bugs can be unavoidable (accounting for physical gating delays using software is difficult)
- FPGAs are relatively inexpensive in comparison, and can have identical performance (or better) without software emulation issues

Technical

In Short

💰(HDL) > 💰(Python+C+WebDev)



More about Verilog (Combinational)

Comments

- C/C++
```
// this is a comment
/* This is a multi-line comment */
```
- Python
```
# this is a comment
```
- Verilog (same as C/C++)
```
// this is a comment
/* This is a multi-line comment */
```

Numbers

- `integer` Literals (32bit signed)
  - e.g. -1, 2, 42
- `real` Literals (64bit float)
  - e.g. 3.1415926535
- Other literals (Unsigned 32bit, for signed add `s` like `'sb`)
  - `'b`: unsigned binary, e.g. `'b1010100`
  - `'d`: unsigned decimal, e.g. `'d1995`
  - `'h`: unsigned hexadecimal, e.g. `'hacd17`

Numbers

- Sized numbers: `[bit_size]'[base_format][number]`
  - e.g. `'3b100`
  - e.g. `'12habc`
  - e.g. `'10d250`
  - Use underscores to separate long binary or hex numbers every 4 bits for readability
    - e.g. `'8b1001_1101`

Types of Objects

- `wire`: inputs, outputs, intermediates (for connecting hardware components)
  - These are like individual physical wires in a circuit
- `reg`: procedural stuff (for sequential circuits' storage)
  - Like registers
  - Doesn't always synthesise to reg/flip-flops: depends on how it's used
- `integer, real, realtime, time`: used in simulation only, cannot be synthesised and loaded onto an FPGA board

Types of Circuits

- Combinational
  - Use `assign` when fixing/transferring values
  - Every statement is evaluated/executed in parallel
- Sequential / Procedural
  - Keywords like `always`, `begin`/`end`, `initial`
  - Control blocks: usually used within `always`, not always sequential
  - We'll cross the bridge when we get there. For now, please don't use these for combinational designs