# Hardware Description Languages

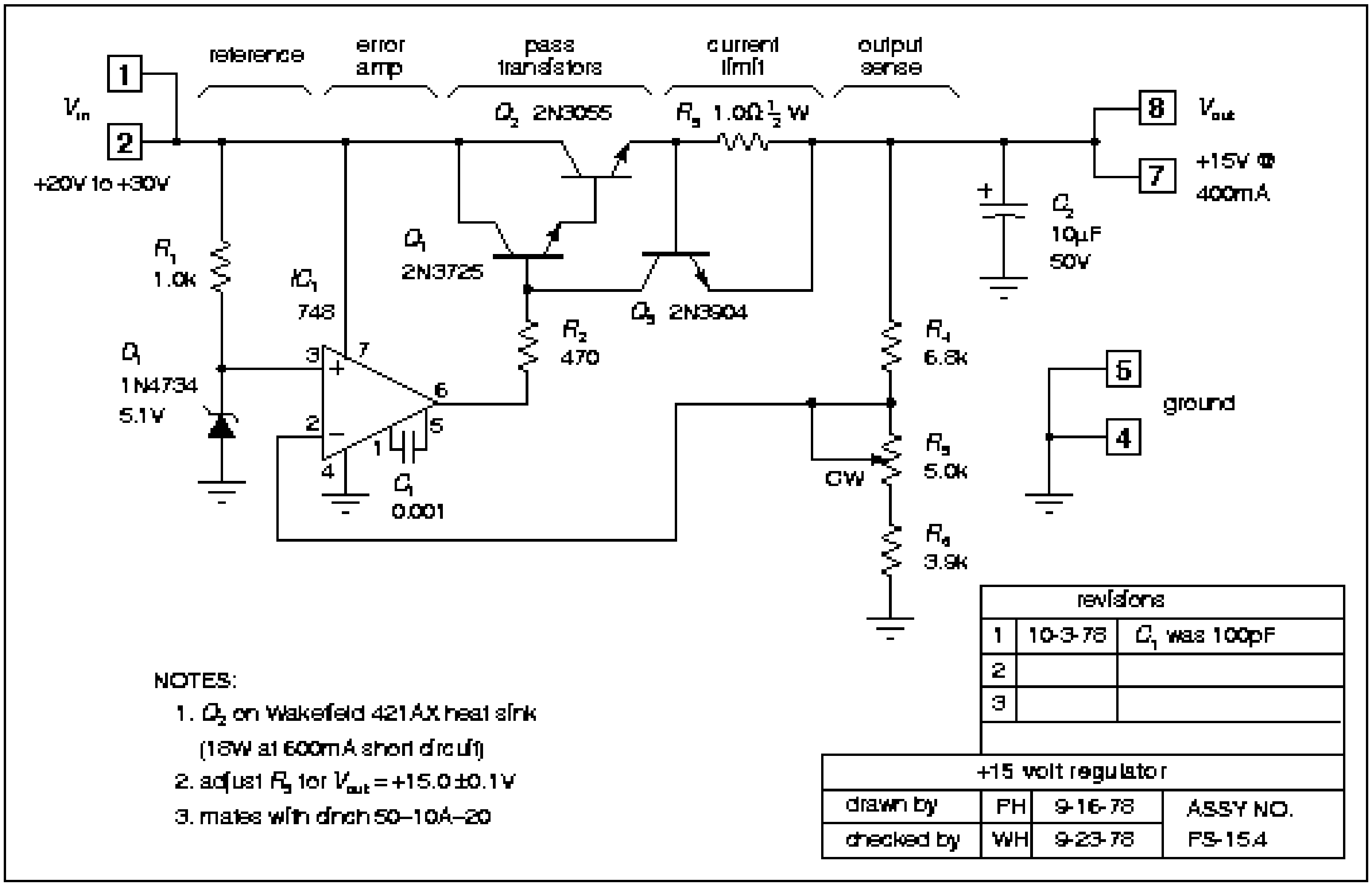
**Meir Guttman**
e-mail: meir@guttman.co.il

# What Engineers (used to) Do…(*)

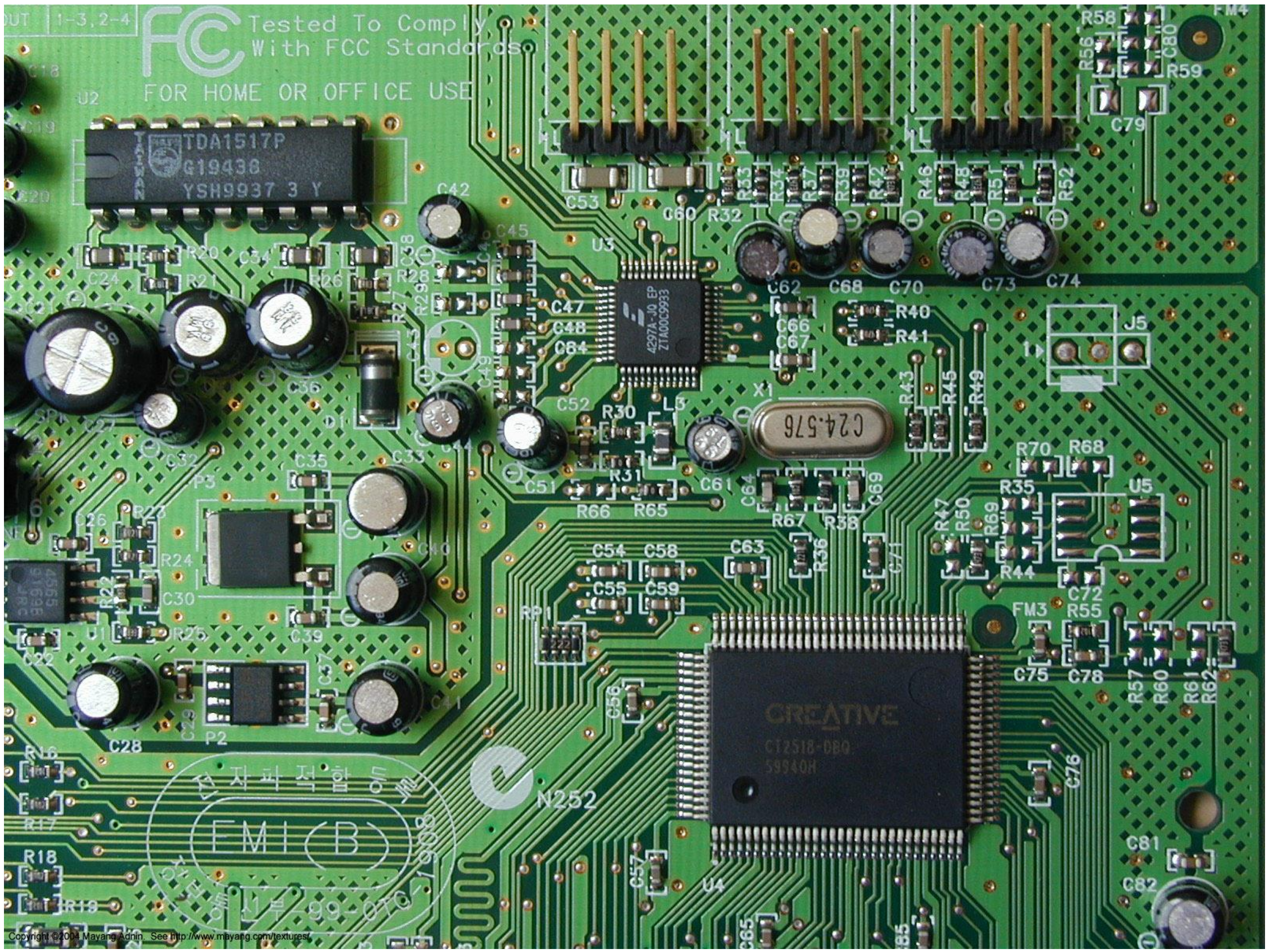
- ☆ Design:
  - Pick components
  - Connect them with wires
- ☆ Implement
  - Design the Printed Circuit Board (PCB)
  - Program a Numerically Controlled (NC) Wire-Wrap wiring machine
- ☆ Build
  - Manufacture the PCB or execute the NC wire-wrap program
  - Solder in components
- ☆ Test the design on a lab bench, hooked up to lab equipment such as Signal generators and Oscilloscopes
- ☆ Find design faults (“bugs”)
- ☆ Repeat…

(*) Logic and analog designs

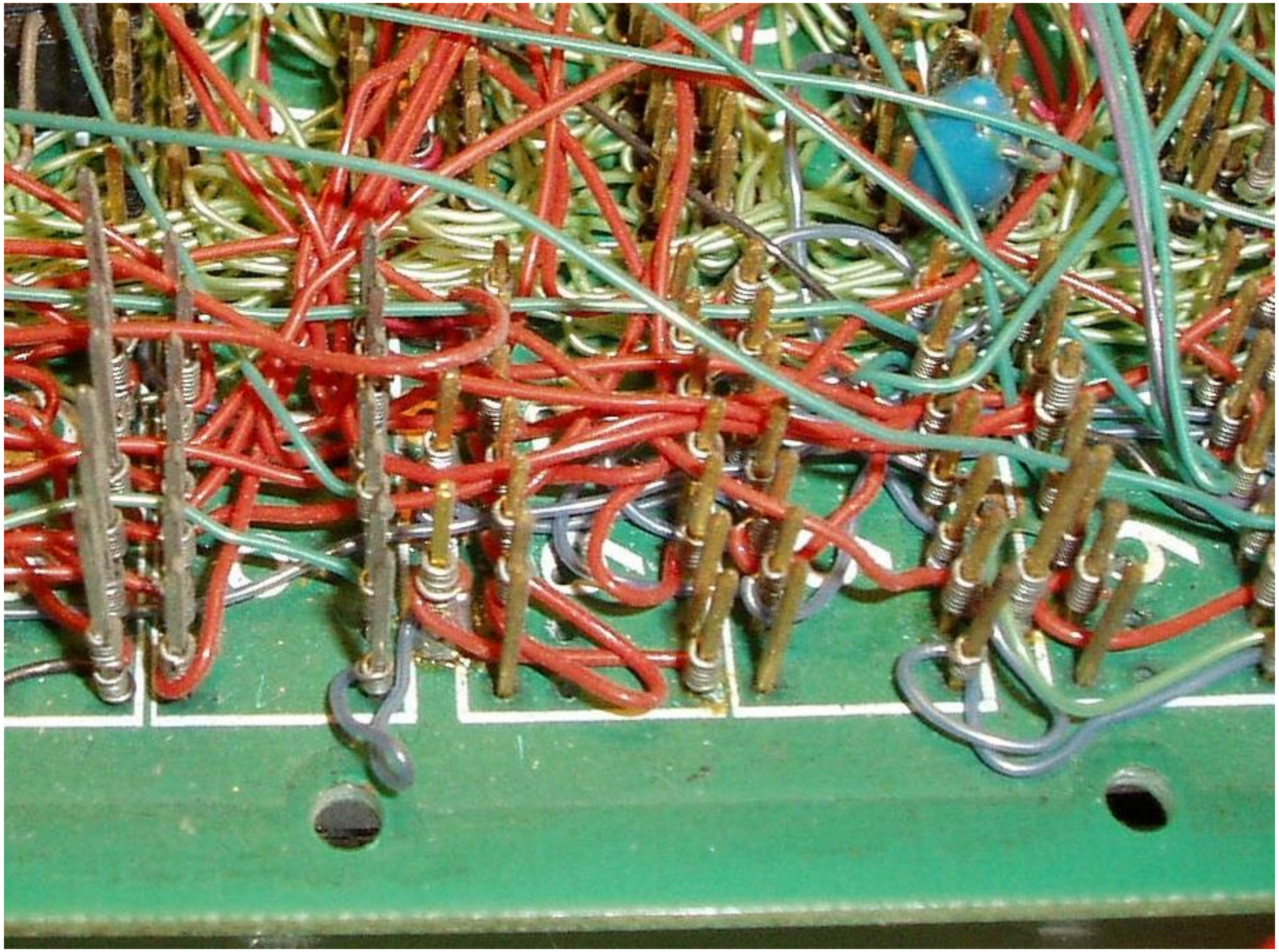
# The "Schematic Diagram"



# A Printed Circuit Board



# Wire-Wrap Board



# Petty Challenges

- Is the schematic diagram what the engineer designed?
- Is the printed circuit-board a true implementation of the design?
- I just discovered a bug, I need to re-wire/replace a component, now what???

$$\lim_{Complexity \to \infty} Handling = ???$$

# Magnum Challenges

- ✰ OK, you want to design an Integrated Circuit (IC):
  - It might be *a billion gates* affair
  - The design team is comprised of hundreds of people
  - The tooling to manufacture a prototype cost millions of dollars
- ✰ How do you describe it?
- ✰ How do you test such a design?
- ✰ How do you correct design-flaws in it?
- ✰ So, to borrow from the Apollo-13 movie: ***"Failure is not an option!"***

# The Advent of the PLD (Programmable Logic device)

- ✰ A generic device
- ✰ It can be “programmed” to implement a logic circuit
- ✰ It has two modes:
  - Program
  - Operation
- ✰ Programming methods
  - Burning internal fuses (the original PLD technology, one-time only!)
  - Charging buried “gates” of transistors (the same technology used by “Flash” memory, can be reprogrammed)
  - Writing into buried memory that controls wiring switches – used in FPGA
- ✰ The last one can re-program parts of the circuit even during the “operation” phase

# Field-Programmable Gate-Arrays (FPGA)

- ☆ Reaching now (march 2014) (not all together):
  - Up to 4.4 million logic cells
  - Up to 1,500 pins
  - Up to 5,000 Gbits/Sec I/O
  - Up to 5,000 DSP cores

# Application-Specific Integrated Circuits

- An ASIC is:
  - A *pre-fabricated* integrated circuit
  - Comprised of the very basic elements such as:
    - *NAND, NOR, XOR gates*
    - *flip-flops*
    - *full-adders*
    - *Counters*
    - *even CPU and DSP cores*
    - *… etc.,*
- An ASIC is made “application-specific” by adding one and usually more “metal” layers to interconnect the I/O ports of these basic building blocks to make a custom circuit.

# Enter HDLs

- ✰ First originating in DARPA's "Very-High-Speed Integrated Circuits" (VHSIC) program
- ✰ The language was named "VHSIC-Hardware-Description language" or VHDL. The HDL and name remains, the project long gone…
- ✰ Later another HDL emerged - Verilog

# Multiple levels of abstraction

- ☆ Referred to as “architectures”
- ☆ “Behavioral”– high-level functional description, even programmed in pure software languages such as ‘C’.
- ☆ “Synthesizable” – “behavioral” too, only a more restricted subset (to be revisited)
- ☆ “Structural” – connecting I/O of “components”
- ☆ In a big design, different parts can use different levels
- ☆ As we move down in the level of abstraction:
  - Simulation is more realistic
  - Real-world problems emerge
  - Simulation times soar
- ☆ So typically a given module designer will simulate his outer circuits at a high-level, using lower levels for his/her part

# Simulation

- A digital circuit is simulated as an “Event-Driven” simulation:
  - We assert a change of state to an input, e.g. from ‘0’ to ‘1’…
  - … this in turn *schedules* a change of state in one or more internal and/or output signals, usually after a “propagation delay”.
  - This can further cascade for some time, scheduling event changes at different times in the future…
  - All scheduled signal-state-changes are inserted into a time-ordered “event queue”.
  - Each event is evaluated in a simulated time series…
  - … until the simulated circuit and its outputs reach a stable state
  - Was the (internal state and) output the required/expected one?
  - If not, hurray! A bug was discovered!
  - Redesign and repeat…

# Simulation (cont.)

- ✩ A unit of simulation in an HDL is a “process”
- ✩ A  process is either:
  - An explicit one, declared by a ‘`begin process ... end process`’
  - An implicit one, e.g. `T_clock <= '1';`
- ✩ All processes are simulated ***concurrently!***
- ✩ Cause-and-effect are taken care of, don’t worry! ☺
- ✩ Other than within a process, ***HDL simulation execution is not by their source code sequence!***
- ✩ Ergo, there is no place for a complaint in the form of “… but this statement is *before* that one!” This can be very bewildering…

# Test-Benches

- A simulated circuit is driven and its outputs are checked by a "Test-Bench"
- A test-bench is simply another HDL module

# Hardware Compilers

- ✰ Take a *behavioral* description of a circuit …
- ✰ … depicted in a *synthesizable HDL* subset…
- ✰ … and creates from it a so-called “gate-level” circuit…
- ✰ … comprised of basic hardware building blocks.
- ✰ Typical basic building blocks:
  - Flip-flops
  - Simple NAND, NOR, XOR gates
  - PLD and FPGA elements
  - Cell library of Application-Specific Integrated Circuit (ASIC)
  - IC “silicon-library” elements (anything from gates, to USB-3 interface and more)
- ✰ These building blocks do not require further design efforts, just “pick and place”

# Hardware "Fitting"

- ✰ All these building blocks are found, one way or another, in a given PLD, ASIC or Custom-IC library.
- ✰ These logical entities must be either:
  - Assigned physical entities in a PLD or an ASIC
  - Placed and interconnected on the silicon substrate in a custom IC
- ✰ Once all these physical elements are assigned/placed and interconnected, they must be translated into either:
  - A downloadable file to a Programmable Logic Device
  - Metallic layers to interconnect ASIC cells
  - A set of IC manufacturing masks – referred to as "Tape-Out"

# What Engineers (still) Do

- ✰ Design:
  - Pick component models
  - Connect them with "wires" ("signals")
- ✰ Simulate
  - Apply (simulated) stimulus
  - Check the *observable* behavior
- ✰ Compile/Synthesize the design
- ✰ … the rest is the same as before, only most bugs were hopefully already eliminated

**The key: Hardware Description Languages**

# Summary

☆ The advent of Hardware Description Languages made possible:

- Hierarchical design of complex hardware
- Test-benching it by simulation rather than by building “bread-boards”
- Simulation at various and simultaneous levels of abstraction:
  - *High-level behavioral HDL*
  - *Synthesizable HDL*
  - *Gate-Level HDL*
- Translating the design into manufacturing tooling
- Re-using major design modules
- Testing a project using PLDs before committing to an ASIC or a fully-custom IC