

Pushing the limits of DIY electronics

Bridging the gap between DIY and professional electronics

Hunz
hunz@mailbox.org

14. August 2015

More sophisticated DIY electronics - why?

- ▶ professional electronics get more and more sophisticated
- ▶ we need to keep up
- ▶ you can do really cool things yourself
- ▶ Examples:
 - ▶ hi-speed FPGA boards (with $>1\text{GByte/s}$ RAM throughput)
e.g. as SDRs, logic analyzers, etc.
 - ▶ custom networking equipment, e.g. ARM-based routers with GBit ethernet interfaces

Challenges for DIY projects

- ▶ more advanced ICs often don't have leads any more
→ saves precious PCB space, more IOs per space
 - ▶ Examples: QFN packages & Ball Grid Arrays (BGAs)
 - ▶ BGAs: solder balls underneath package, cannot be soldered w/ soldering iron
- Boards with 4+ layers needed due to high number of signals
- Other soldering technique needed (often reflow soldering)
- high mechanical precision required for assembly
- ▶ Faster busses & interfaces (DDRx busses, PCI Express, HDMI, etc.)
 - ▶ layout more demanding
- better software (or more time) & more know-how needed

In this talk

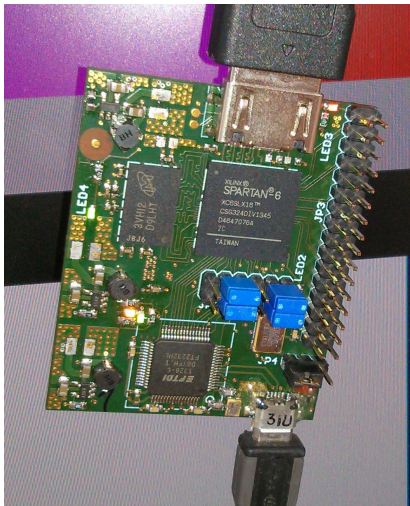
1. General hints
2. Reflow soldering with solder paste & board assembly
3. hi-speed busses & interfaces (characteristic impedances, length matching)
4. multilayer boards (special stackups required for hi-speed)
5. design software (Eagle & KiCAD, etc.)

General hints

- ▶ make a tiny test-board for risky parts like your first DDRx design before using it in a complex project
- ▶ using outer layers allows fixing things later with copper wire
- ▶ use Electroless Nickle Immersion Gold (ENIG) finish for fine-pitch & BGA boards
(extremely even surface - contrary to HAL)
- ▶ higher soldering temperature needed for Pb-free: 237°C, max. 260 (narrow margin)
- ▶ some people do BGAs w/o paste, just w/ flux - wouldn't recommend it as I had problems with that once
- ▶ mind the popcorn effect: keep ICs in original pouch with desiccant & indicator paper - reseal pouches with adhesive tape

Example: Testboard for FPGA + DDR2-RAM

- ▶ tiny test-PCB
- ▶ FPGA+RAM
- ▶ voltage regulators
- ▶ FT2232H for JTAG
- ▶ HDMI connector
(directly connected to
FPGA IOs)



Soldering

basically 3 options:

- ▶ Reflow oven
 - ▶ modified pizza oven (more heat, temperature control)
 - ▶ cheap oven from china (modified)
 - ▶ Problem: black surfaces (IC packages, connectors) absorb more IR radiation from quartz lamps than metal surfaces
→ temperature on PCB varies with location
- ▶ hot air gun with hot plate
 - ▶ cheap hot air guns available
 - ▶ problem: hard to tell when BGA balls melt
 - ▶ problem: large copper areas won't get sufficiently hot w/o preheater
 - ▶ hot plates for cooking can be used as cheap preheaters
 - ▶ set them to (100-120 °C)
 - ▶ check PCB surface w/ IR thermometer

Soldering (2)

- ▶ Vapor phase
 - ▶ special liquid which boils at high temperature (230-240°C)
 - ▶ great soldering method
 - ▶ good heat transfer - PCB rests in vapor phase, not in liquid
 - ▶ overheating ICs is difficult
 - ▶ BUT: if it does get too hot, very bad things will happen (hydrofluoric acid & stuff, see <http://www.heise.de/make/meldung/Dampfphasen-Medium-in-Kleinmengen-1780946.html> (german))
 - ▶ liquid is very expensive but you won't need much per run
 - ▶ you need a high pot (e.g. pot for asparagus)
 - ▶ several DIY projects, haven't tried it myself yet

Soldering: Reflow oven

- ▶ T962 from china
- ▶ small holding capacity, but high heating power (often a problem w/ pizza ovens)
- ▶ LPC2134 arm controller - not read-protected!
- ▶ did a dump of the firmware, reverse-engineered it
- ▶ wrote my own firmware - temperature controller implemented on a PC
- ▶ connection to oven using 3.3V RS232
- ▶ only few commands: heat, read temperature, run fan
- ▶ host-code done in Lua - easy to adapt or add new temperature profiles
- ▶ tiny firmware runs completely from RAM
- ▶ code: <https://github.com/znuh/t962>
some documentation will follow

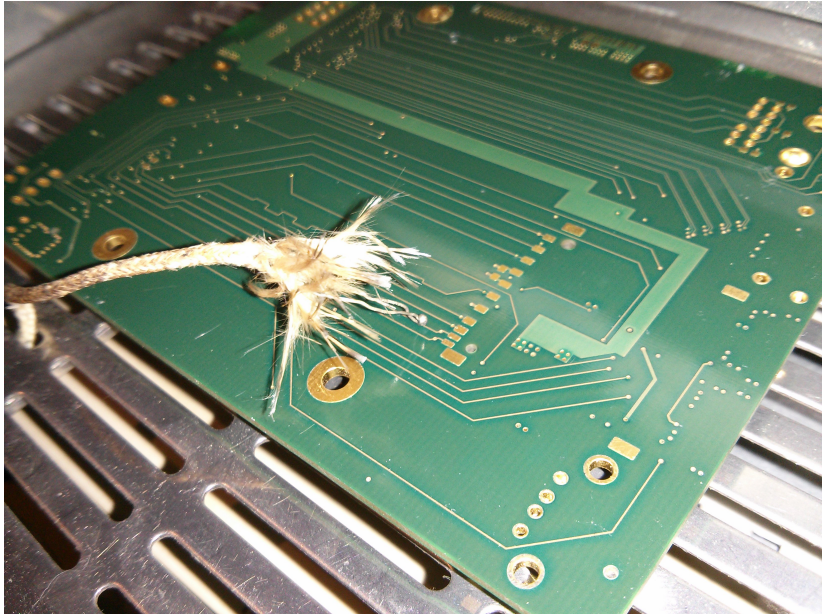
Soldering: T962 oven

- ▶ masking tape should be replaced w/ kapton tape (heat resistance)
- ▶ for my alternative firmware 3.3V RS232 Rx&Tx pinheaders
- ▶ external reset & boot buttons (for bootloader invocation)
- ▶ **add pullup resistors for triacs** - otherwise fan & heaters will be on while in bootloader!
- ▶ spacers for PCB (e.g. screw nuts)
- ▶ measure temperature directly on PCB (K-type sensors are well suited)
- ▶ do some test-runs first
- ▶ more docs (in german): <https://www.mikrocontroller.net/topic/310148#3649461>

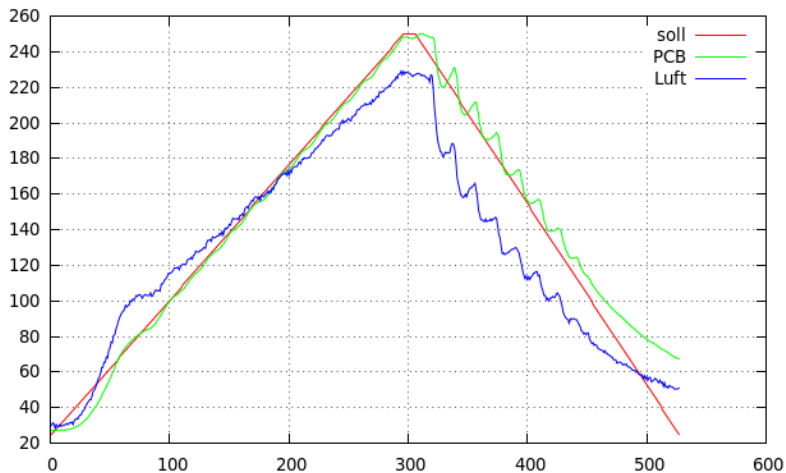
Soldering: T962 oven (2)



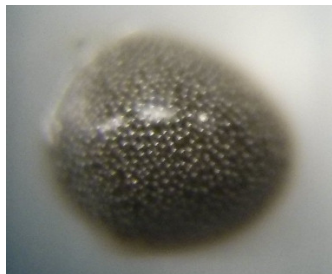
Soldering: T962 oven (3)



Soldering: T962 oven (4)



Solder paste

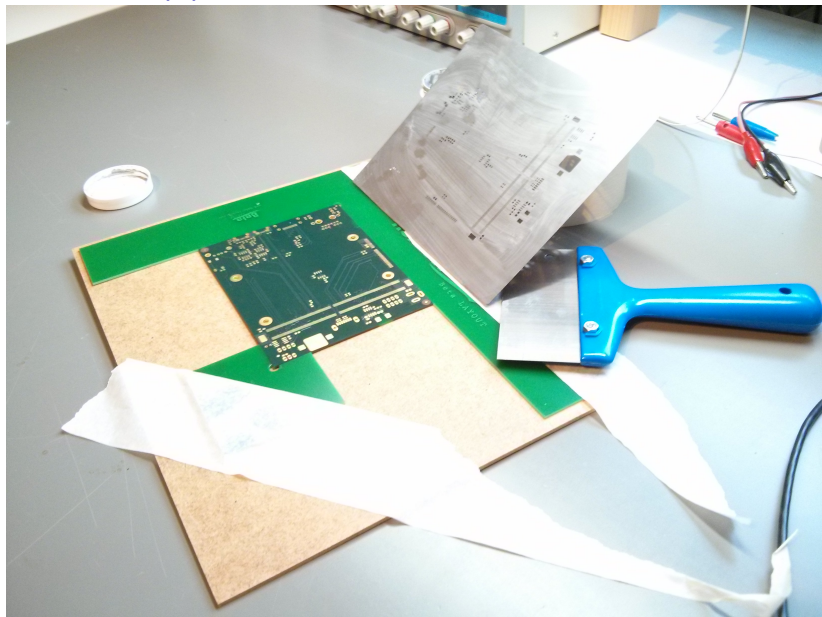


- ▶ solder paste: tiny solder balls embedded in flux
- ▶ for fine-pitch & BGAs you need a stencil
- ▶ paste is forced through openings of stencil
- ▶ some PCB houses offer free stencils for your boards (e.g. pcb-pool), others don't
- ▶ sometimes a single-layer dummy-PCB + free stencil from another PCB house is cheaper than a non-free stencil from the house you get your PCBs from

Solder paste (2)



Solder paste (3)



Solder paste (4)

- ▶ you need to fix the stencil firmly to the PCB so that you can fold up the stencil after printing
- ▶ there are fixing-kits for this, you can also make sth. yourself
- ▶ stir paste well before use, keep in refrigerator - paste is usually good for about 1 year
- ▶ if a tiny paste droplet covers two adjacent pads it's not always a problem - tiny amounts will flow to where they should be during solder process
- ▶ cleaning a stencil works well with lighter fluid
- ▶ if you mess up the printing you can clean the PCB and start over
- ▶ wear a lab coat - paste will get everywhere and getting it out of your clothes is a challenge

Placing & aligning parts

- ▶ place most challenging parts (BGAs, QFNs) first, easy parts later
- ▶ if you smeared the paste too much to fix clean the PCB & parts, start over
- ▶ for BGAs and QFNs place a rectangle around the chip in the silkscreen
- ▶ helps a lot with the alignment
- ▶ use a tweezer for picking & placing the chips
- ▶ vacuum tweezers: only recommended with electric pump
- ▶ place chip softly on solder paste
- ▶ align chip carefully, inspect w/ magnifying glass
- ▶ push chip slightly down

Placement helper for eagle

- ▶ hacked a quick components placement helper for eagle board
- ▶ needs Lua, LuaXML, lgi (Lua GObject Introspection) with Gtk and Cairo
- ▶ only tested on Linux
- ▶ lgi is (Windows) / might be tough to install on other OSes
- ▶ code: <https://github.com/znuh/pcbtools>
- ▶ usage: **lua gui.lua board.brd**
- ▶ easy to use
- ▶ basically translates Eagle XML elements to Cairo calls
- ▶ eats RAM for breakfast (1 cairo surface per layer)

Placement helper for eagle

The screenshot shows the Eagle PCB design software interface. The main window is titled "brdviewer" and displays a complex PCB layout with various components and traces. On the left side, there is a panel with two tabs: "Layers" and "Elements". The "Elements" tab is active, showing a list of components with their IDs, top/bottom layer counts, and package names. The component "C0402" is highlighted in blue. Below the list is a "clear highlight" button. At the bottom of the window, there is a status bar with a "scale" slider, a "rotate" button set to 90°, and checkboxes for "mirror X" and "mirror Y". The status bar also displays "0 / 17 C0402 2.2uF".

#	#Top	#Bot	Pkg
17	0	17	C0402
9	0	9	R0603
13	5	8	C0603
7	2	5	R0603
4	1	3	0603
4	1	3	R0603
4	1	3	C0805
3	0	3	C0603
3	1	2	C0603
2	0	2	R0402
2	0	2	R0402
4	2	2	KUBUS3_PASSERMARK
1	0	1	R0402
1	0	1	C0805
1	0	1	SOIC8
1	0	1	8S2
1	0	1	SO-08M
1	0	1	R0603
1	0	1	R0603
1	0	1	PAD.03X.03
3	2	1	C0402

clear highlight

scale [slider] rotate 90° [checkbox mirror X] [checkbox mirror Y] 0 / 17 C0402 2.2uF

Hi-speed busses & interfaces

- ▶ signal integrity for high-speed signals is a challenge
- ▶ length-matching needed
 - ▶ **propagation speed/delay** depends on the **dielectric constant** ϵ_r of the insulator
 - ▶ rule of thumb for PCBs: 6ps/mm for top/bottom, 7ps/mm for inner layers
 - ▶ signals in inner layers are approx. 20% slower than on top/bottom!
 - ▶ not respected by eagle, vias not considered either
 - ▶ eagle measures full length of tracks, not just between ICs (termination resistors usually after ICs)
- ▶ transmission lines with **characteristic impedance** needed
 - ▶ special multilayer stackups
 - ▶ width of PCB tracks depend on multilayer stackup

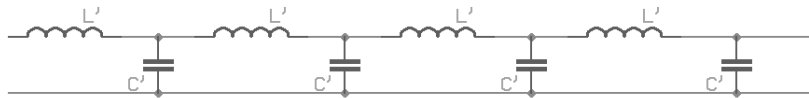
Length matching tool for eagle

- ▶ <https://github.com/znuh/pcbtools>
- ▶ work in progress, no documentation
- ▶ uses Lua, needs LuaXML
- ▶ edit delay_config.lua
- ▶ **lua -i bga_workshop.lua** - starts interactive Lua session
- ▶ **load_brd('myboard.brd')** loads eagle board
- ▶ **get_signal_delays('U1', 'IC4')** shows adjusted metric distance for signals between IC4 and U1
- ▶ **get_signal_delays('U1', 'IC4', 'time')** shows time delay for signals between IC4 and U1

Length matching tool for eagle: screenshot

```
> get_signal_delays("U1", "IC4", "time")
GND                : inf ps
DRAM_A0            : 126 ps
DRAM_A1            : 102 ps
DRAM_A2            : 108 ps
DRAM_A3            : 126 ps
DRAM_A4            : 102 ps
DRAM_A5            : 89 ps
DRAM_A6            : 101 ps
DRAM_A7            : 103 ps
DRAM_A8            : 96 ps
DRAM_A9            : 90 ps
DRAM_A10           : 86 ps
DRAM_A12           : 105 ps
DRAM_A13           : 116 ps
DRAM_BA1           : 89 ps
DRAM_BA2           : 91 ps
DRAM_BA0           : 98 ps
DRAM_N_CAS         : 117 ps
DRAM_CK_P          : 135 ps
DRAM_CK_N          : 135 ps
DRAM_CKE           : 91 ps
DRAM_LDM           : 90 ps
DRAM_DQ0           : 81 ps
```

Characteristic impedance Z_0



- ▶ **high frequency signals** travel through conductors like a wave
- ▶ transmission line model with capacitance C' & inductance L'
- ▶ these values depend on the **geometry of the transmission line**
- ▶ characteristic impedance $Z_0 = \sqrt{\frac{L'}{C'}}$ (with losses ignored)
- ▶ signal reflected at end of transmission line if not properly terminated
- ▶ reflected signals can travel back and forth destroying the signal integrity
- ▶ proper termination with resistor equivalent to characteristic impedance (remember 10Base2?)

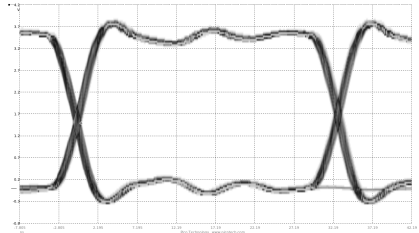
Transmission lines

- ▶ a few more details: termination & slew rate
- ▶ simulating transmission lines
- ▶ designing transmission lines for a given impedance
- ▶ multilayer stackups for transmission lines

Termination schemes

- ▶ mostly on-die termination for point-to-point signals like LVDS
- ▶ termination may be needed for multipoint connections
- ▶ stubs are very bad - avoid them, place termination after last IC
- ▶ DDR2/3 memory has builtin termination (ODT) for data busses
- ▶ clock and control/address lines need extra termination
- ▶ for control/address lines you might get away without any termination
- ▶ for synchronous signals reducing clock frequency can help (signals have more time to settle before clock edge)

Frequency vs. slew rate



- ▶ Simply reducing the frequency of a transmission doesn't improve the signal quality
- ▶ Why?
- ▶ Reducing the frequency doesn't reduce the **slew rate**
- ▶ **slew rate**: time for a signal to go from low to hi (t_{rise}) or vice versa (t_{fall})
- ▶ slew rate always has **higher frequency portions** than base frequency of transmission (see spectrum of a square wave)
- ▶ short/fast slew rates are needed for high frequencies (otherwise signal doesn't reach hi/low state in time)
- ▶ more details: <http://www.mikrocontroller.net/articles/Wellenwiderstand> (german)

Finding the slew rate

- ▶ sometimes t_{rise} and t_{fall} given in datasheet
- ▶ time from 10% of hi level to 90% of hi level
- ▶ professional tools use **IBIS** data: Input/Output Buffer Information Specification
- ▶ these files are textfiles, contain waveform data
- ▶ e.g. waveform for output buffer driving a 50Ω transmission line
- ▶ can be used to simulate & analyze signal integrity with given transmission line & termination
- ▶ easy to read and plot (e.g. w/ gnuplot)

Simulating transmission lines

- ▶ SPICE is usually used for electronics simulation
- ▶ SPICE is just the engine, no user interface
- ▶ easy to use, free as in beer user interface: LTspice
<http://www.linear.com/designtools/software/>
- ▶ download without registration - thank you Linear! <3
- ▶ Windows & OSX versions available, windows version works very well with wine
- ▶ you can find tutorials on the web

Two open-source projects for using IBIS with SPICE:

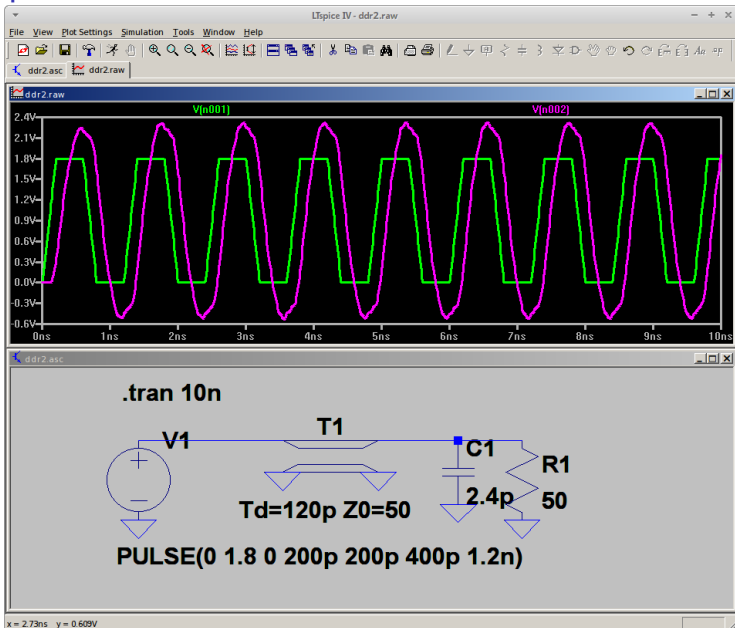
- ▶ <https://github.com/russdill/darter>
- ▶ <http://www.thedigitalmachine.net/eispice.html>

Haven't tried them yet

Simulating transmission lines with LTspice

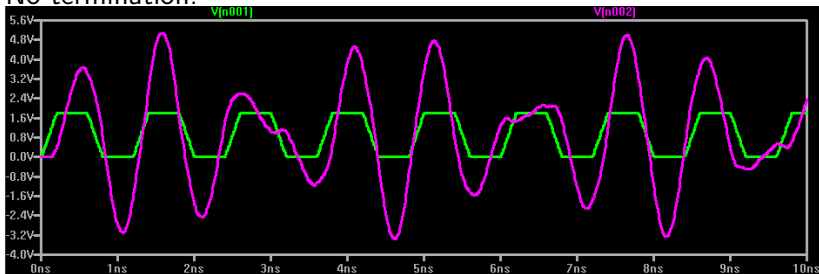
- ▶ use a PULSE voltage source
 - ▶ insert t_{rise} and t_{fall}
 - ▶ OR: use a PWL file (time/value text file)
 - ▶ can be easily generated from IBIS file
- ▶ add a transmission line (tline)
 - ▶ Td: electrical length
simulating the longest line of a bus is usually sufficient
 - ▶ Z0: characteristic impedance
- ▶ add input buffer and termination
 - ▶ capacitance of input buffer important - see datasheet
 - ▶ often on-die termination with Z_0 available for hi-speed signals
 - ▶ for unterminated inputs: usually $1M\Omega$

LTspice demo

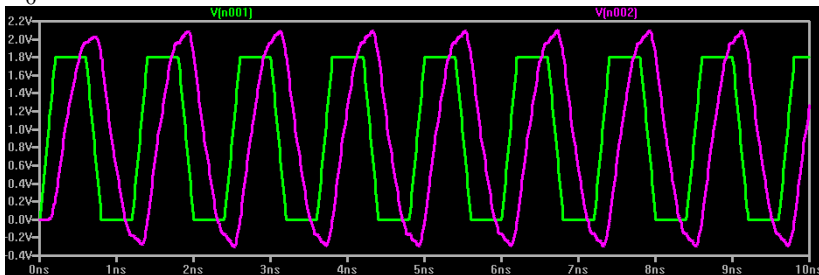


Z_0 mismatch

No termination:

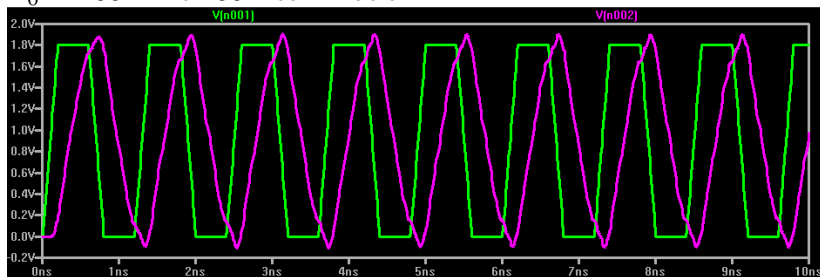


$Z_0 = 75\Omega$ with 50Ω termination:



Z_0 mismatch (2)

$Z_0 = 100\Omega$ with 50Ω termination:



- ▶ Z_0 doesn't need to be spot-on
- ▶ signal gets gradually worse as mismatch grows
- ▶ specifications often allow $\pm 20\%$
- ▶ we try to hit Z_0 as good as possible

Designing transmission lines for a given Z_0

- ▶ always at least one low-impedance reference plane needed
can be GND or Vdd - GND is better (less noise)



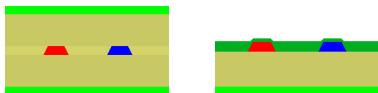
- ▶ a: (asymmetric/offset) **stripline**: for inner layers - 2 reference planes
asymmetric/offset: unequal distance to reference planes
- ▶ b: (coated) **microstrip**: for outer layers
coated: w/ solder mask above the conductor
- ▶ c: **coplanar waveguide** (CPW): good for analog signals, normally not used for busses (space requirements)
optional additional ground plane

Designing transmission lines for a given Z_0 (2)



- ▶ a: (asymmetric/offset) **stripline**: for inner layers - 2 reference planes
asymmetric/offset: unequal distance to reference planes
- ▶ b: (coated) **microstrip**: for outer layers
coated: w/ solder mask above the conductor
- ▶ c: **coplanar waveguide** (CPW): good for analog signals, normally not used for busses (space requirements)
optional additional ground plane
- ▶ Z_0 depends mostly on:
 - ▶ width w of conductor: **increasing** $w \rightarrow$ lower Z_0
 - ▶ height h (h_1/h_2) of conductor above reference plane(s):
decreasing $h \rightarrow$ lower Z_0
 - ▶ dielectric constant ϵ_r of insulators: higher $\epsilon_r \rightarrow$ lower Z_0

Differential transmission lines



- ▶ differential lines often used for hi-speed signals (Ethernet, LVDS, PCI Express, SATA, etc.)
- ▶ more robust to noise
- ▶ driven with inverse polarity (P: positive, N: negative)
- ▶ receiver utilizes difference between P&N
- ▶ noise couples into both lines, doesn't change difference
- ▶ differential impedance Z_{diff} : usually $2 * Z_0$
- ▶ more details: <http://www.polarinstruments.com/support/cits/AP157.html>
- ▶ Z_{diff} depends on Z_0 of individual lines
lower $Z_0 \rightarrow$ lower Z_{diff}
- ▶ distance s between conductors: lower $s \rightarrow$ lower Z_{diff}

Some notes re: transmission lines

- ▶ trapezoid shape tracks produced due to etch-back
- ▶ etch-back usually equal to copper height
- ▶ affects differential pairs more than single-ended
- ▶ ϵ_r for FR-4 PCBs frequency-dependent - 4.2 usually is a good value
- ▶ microstrips see a mix of air soldermask and FR-4 ϵ_r
- ▶ soldermask height above PCB and conductors depends on PCB house
- ▶ example values: 42um above PCB, 15um above conductor
source: http://www.we-online.de/web/de/index.php/show/media/04_leiterplatte/2013_1/webinare_1/signalintegritaet/Webinar_Signal_final.pdf

Calculating w , h and Z_0

- ▶ there are approximation equations
- ▶ various javascript implementations
- ▶ e.g.: <http://www.leiton.de/leiton-tools-impedance-capacity-calculator.html>
- ▶ KiCAD has a builtin calculator (`pcb_calculator`)
- ▶ more precise: field solver
- ▶ open source 2D solver: atlc
<http://atlc.sourceforge.net/>
- ▶ input data: windows BMP files
- ▶ no GUI, console tool - takes a while depending on BMP size
- ▶ mdtlc: free (beer) windows GUI with builtin atlc backend (old atlc version though) <http://mdtlc.sourceforge.net/>

- ▶ transmission line images from previous slides are atlc input images
- ▶ calculates signal propagation speed, single-ended and differential impedance
- ▶ bitmaps can be generated with console tools
- ▶ or with a simple graphics library like GD
- ▶ red, blue and green for P, N and reference conductors
- ▶ other colors for insulators - custom ϵ_r values can be mapped to custom colors
- ▶ ratio between h, w, s, etc. important - actual scale doesn't matter as long as it's fine enough
- ▶ e.g. 1px for 5um, larger images are more accurate but take more CPU-time

some of my tools

- ▶ work in progress, crappy, started a rewrite
- ▶ Lua code, no documentation yet
- ▶ <https://github.com/znuh/pcbtools>
- ▶ **config.lua**: multilayer stackup definitions, required impedances - **edit these**
- ▶ then run **lua -i impedance.lua** (interactive Lua session)
- ▶ will give you quick results using approximation equations
- ▶ generates a visualization of stackup as stackup.png
- ▶ **run_atlc(name, layer_id, w, s)** can be used to generate a BMP and run atlc (s only for differential signals)
- ▶ needs lua-gd and convert from imagemagick, BMPs generated in atlc/ subdir
- ▶ units in mm, layer_id needs to be a signal-layer, not a reference layer
- ▶ example: `run_atlc("pcie",2,0.25,0.25)`

Multilayer stackups

- ▶ height h between layers depends on multilayer stackup
- ▶ we need to choose a multilayer stackup which gives us good h values
- ▶ custom stackups are quite expensive (manufacturer has to produce a full panel just for you)
→ we need to choose a PCB house with a suitable pool stackup
- ▶ ensure that PCB manufacturer won't use another stackup for your pool order - talk to your PCB house before design & production!
- ▶ **controlled impedance**: manufacturer guarantees certain $Z_0 \pm x$ - expensive, not in pool, we can do without

Multilayer stackups (2)

- ▶ you usually want to use top/bottom as signal layers so you can patch signals later if necessary
- ▶ that means adjacent layers should be reference layers if you want to use microstrips
- ▶ solid copper planes in layer stack need to be symmetric (copper balance)
- ▶ Vdd planes close to GND are good because they form a good capacitor for HF
- ▶ with less than 8 layers h often is very high
- ▶ this results in rather large track widths w for $Z_0 = 50\Omega$
- ▶ might lead to space issues during routing
- ▶ we need to make trade-offs (e.g. slight Z_0 -mismatch or non-adjacent Vdd/GND layers)

Multilayer stackups - examples



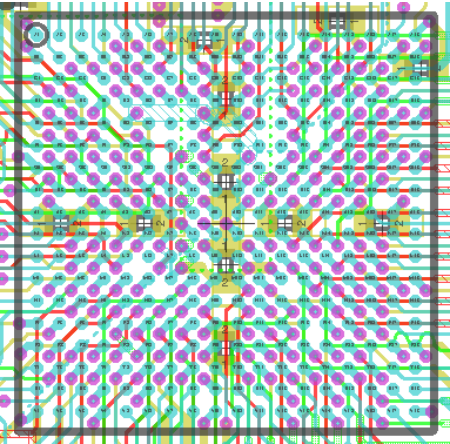
left to right:

pcb-pool ML6, WE-direkt pool ML6, eurocircuits pool ML8

PCB manufacturers - some examples (mostly Europe/Germany)

- ▶ Eurocircuits: <http://www.eurocircuits.com/> - pool available for up to 8 layers, detailed stackup view, guaranteed stackup
- ▶ Leiton: <http://www.leiton.de/> - pool for up to 8 layers
- ▶ PCB-pool: <http://www.pcb-pool.com/> - pool for up to 6 layers, free stencil and ENIG, you need to ask for the designated pool stackup
- ▶ WE direkt: <http://www.wedirekt.de/index.php/pcb> - pool for up to 8 layers, small additional fee for guaranteed pool stackup

Dogbone breakout for BGAs

- ▶ divide pads in 4 quadrants
 - ▶ place 0.2mm vias between pads
 - ▶ place 0402 decoupling caps in via-free zone
 - ▶ works well with pitch $\geq 0.8\text{mm}$
- 
- ▶ ensure to cover vias with solder stop (in eagle: Masks / Limit: set to $>$ via size, check tStop layer)
 - ▶ minor clearance violations usually no problem (e.g. 0.1125mm instead of 0.125mm)

Design software

- ▶ Eagle: <http://www.cadsoftusa.com/eagle-pcb-design-software/about-eagle/>
- ▶ hobbyist version for non-commercial projects with ≤ 6 layers
- ▶ XML boardfiles easy to read, understand & modify

- ▶ KiCAD: <http://www.kicad-pcb.org/>
- ▶ free & open-source
- ▶ make sure to use devel version, not stable (lots of new features) - save often, make backups!
- ▶ layout engine very powerful, push&shove, etc. better suited for more complex projects
- ▶ builtin python interpreter
- ▶ boardfiles: textfiles, easy to read, understand & modify
- ▶ builtin 3d viewer

Conclusion

- ▶ lack of expensive professional software can be compensated for with free/open source software, custom tools, knowledge and a high motivation
- ▶ reasonably priced pool PCBs can be used for very advanced projects
- ▶ reflow soldering can be done without expensive equipment too
 - needs some experiments & patience

further strategy:

- ▶ write & improve software
- ▶ build & provide reflow equipment in hackerspaces
- ▶ collaborate

more questions? DECT: hunz (i'm in the Königlich Bayerisches Amtsvillage - the thing with the bavarian flags)