
Horizon-EDA dokumentace programu

Vydání 1.0 CZ

Lukas K.

20.11.2019

Obsah:

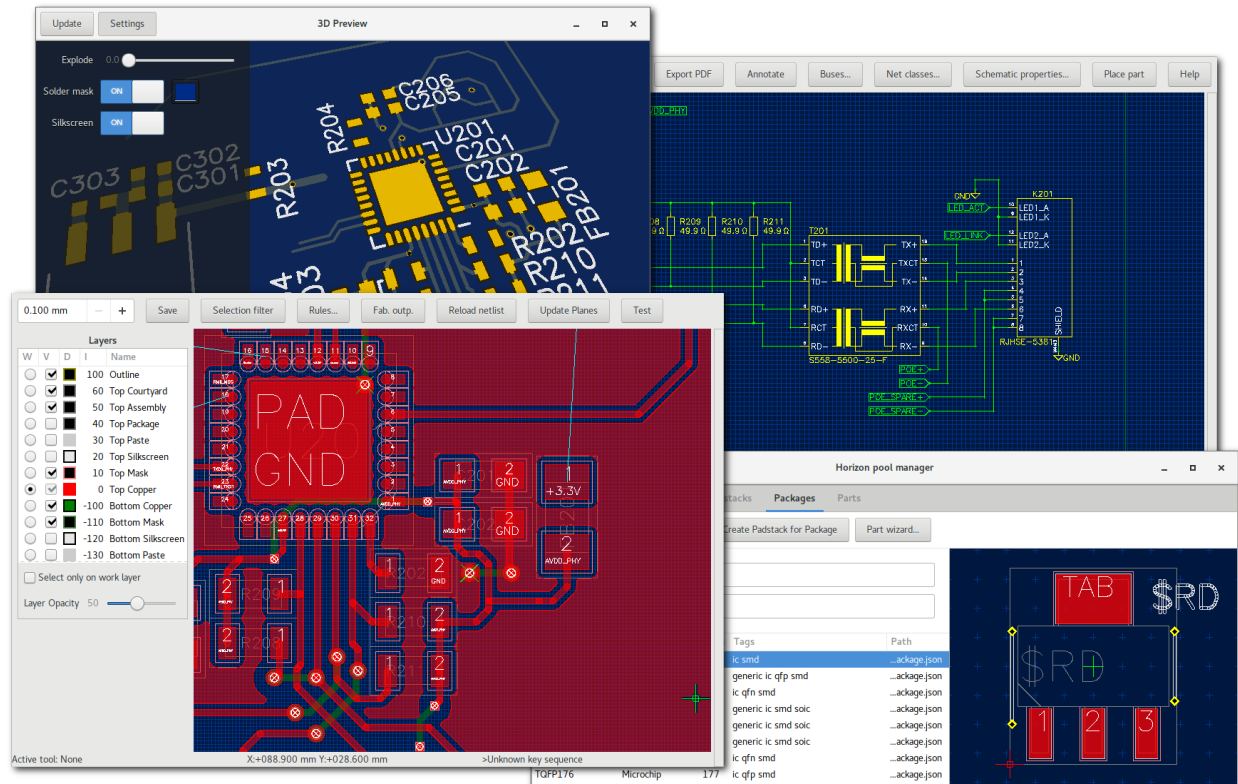
1	Why another EDA package	3
2	Přehled funkcí	5
3	Non-Goals	17
4	Installation	19
5	Setting up a pool	21
6	Create a new Project	23
7	Draw a Schematic	27
8	Create a Board	33
9	Example project	39
10	Tools	41
11	Spacebar Menu	43
12	Grid	45
13	Numeric Entries	47
14	Drawing	49
15	Selection	53
16	Moving and the interactive manipulator	55
17	Layers	57
18	Editor schémat	59
19	Editor desky plošných spojů DPS (PCB)	61
20	Kopírování rozmístění a umístění	63

21 Zpětné úpravy (Backanotation)	65
22 Pravidla	67
23 Why a Pool?	69
24 Fond knihovny součástek, pouzder, pájecích míst a schematických značek (Pool)	73
25 Správce fondu (Pool manager)	77
26 Vytvoření pouzdra součástky	81
27 Contribute to the Pool	83
28 Sestavení (kompilace) programu na operačním systému Windows	85
29 Sestavení (kompilace) programu na operačním systému Linux	87
30 Parametry programu	89
31 Popis funkce	93
32 Použití příkazového řádku (CLI Command Line Interface)	95
33 Python module	97

Tato dokumentace je zatím rozpracovaná

Horizon EDA je aplikace pro návrh elektroniky **Electronics Design Application** je to podobný nástroj jako *KiCAD_* nebo *LibrePCB_*. Umožňuje kreslit schémata, řešit rozvržení plošných spojů a komfortně spravovat fond součástek.

Podívejte se na dokumentaci programu Horizon EDA *Hlavní výhody* nebo přímo *Zkuste* nebo začněte *číst od začátku*.



Why another EDA package

So you may be wondering why I started horizon EDA back in 2016, given that KiCad was a thing at that time.

Let's get started with a quote from Tom Hausherr: [PCB Design Perfection Starts in the CAD Library](#)

[...] PCB design perfection starts in the CAD library.

This also applies to EDA software as in a schematic / PCB design tool can only be as good its library structure. Since the definition of library items such as symbols packages is the very foundation of any EDA software, changing these definitions is next-to-impossible without significant changes to almost all other parts of the application. Having a certain library structure in place also thus guides any further development on that EDA application.

Having used KiCad for small and medium-sized projects, my biggest pain points were the library lacking a concept of orderable parts without duplicating the symbol for each part and the schematic editor not knowing about nets. KiCad's board editor, while being quite good as a layout tool was lacking expressive design rules. Especially the first and second point didn't seem easy to alleviate without major changes that would have involved lots of discussion since these would be breaking changes.

That made me start thinking how I'd design an EDA tool that meets my wishes and is easy enough to implement from scratch as a one man show and provides a clean-slate playground for experimentation.

At the very core of these thoughts was to keep schematic and netlist representation of a design separate to allow for non-schematic based workflows such as interconnectivity tables. That lead to the decision to define pins and their direction in what's called a Unit and not in the symbol as it's common among many other EDA packages. This also makes it possible to have multiple symbols representing the same thing (such as a resistor) without any effect on the netlist. Apart from name and direction, a pin as it's defined in a Unit can also have multiple alternate names to specify multiple pin functions as they're commonly available on MCUs and FPGAs.

To define the netlist representation of an actual part, units are referenced by what's called an Entity. This reference is called Gate. For simple parts, an entity references just a single Unit that includes all pins. For some parts it makes sense to have more gates.

Parts that include multiple instances of the same functionality such as quad opamps will then reference the opamp unit 4 times as well as a unit for power supply.

On the board side of things, a packages are defined as in pretty much every EDA package out there - pads and graphical items such as silkscreen, reference designator and assembly outline. Pads however are defined by a padstack describing copper, soldermask and other layers in terms of shapes and polygons. This greatly facilitates odd-shaped pads as they

can be drawn as-is without resorting to hacks such as using multiple pads to make up one actual pad. To avoid having a custom-drawn padstack for every pad size, padstacks are usually accompanied by a short script written in a custom stack-based language to adjust their size as well as other properties such as corner radius or solder mask expansion.

The pads of a package are mapped to the pins defined in the units referenced by an entity in what's called a Part. In order to mapped to something orderable, Parts have fields for the manufacturer name and the manufacturer's part number (MPN) among other details such as datasheet link or description.

All aforementioned references between items (such as entities referencing units) are by UUIDs. In Horizon EDA, all items get assigned an immutable UUID at time of their creation. This UUID is then used by other items to reference this item.

To wrap up this introduction:

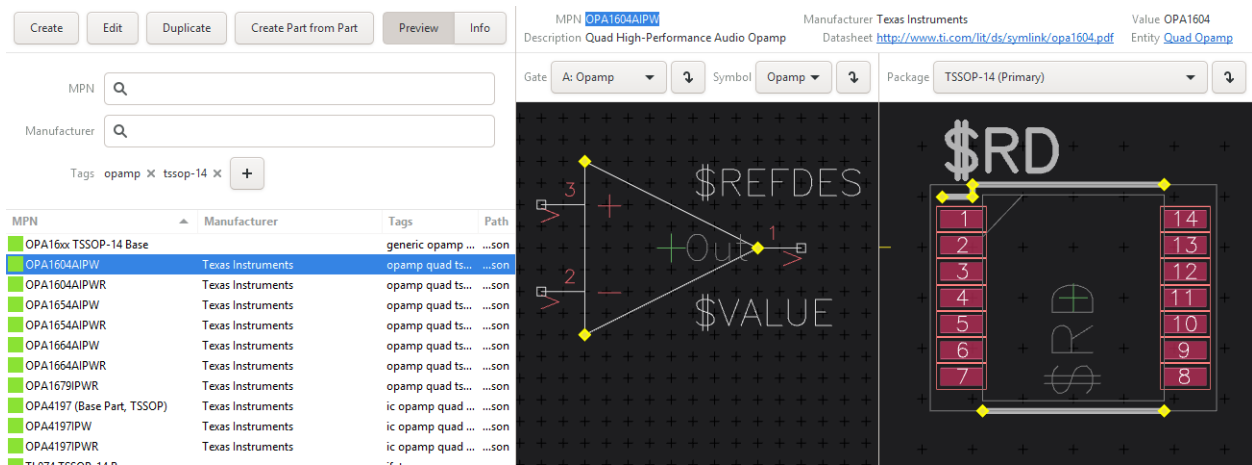
My biggest weakness is that i will eventually turn any arbitrary electronics project into an excuse to write EDA software (and vice versa).

(Based on <https://twitter.com/mycoliza/status/824809235632447492>)

Next: *Feature Overview*

2.1 Interaktivní a jednoduchý management součástí

Snadná správa součástí, pouzder součástek a symbolů schemat pomocí správce fondu knihovny součástí (Pool manager):



Přiřazení vývodů k pájecím ploškám v editoru součástí (Part editor):

Save

Part Editor

×

MPN

LTC2400CS8

Inherit

Value

Inherit

Manufacturer

Linear Technology

Inherit

Tags

adc ic

Inherited:

Inherit

Entity

LTC2400 / Linear Technology

Package

SOIC-8 /

Base part

none

Parametric data

Copy from base

{}

Gate	Pin	Mapped
Main	~CS	✓
Main	Fo	✓
Main	GND	✓
Main	SCK	✓
Main	SDO	✓
Main	Vcc	✓
Main	Vin	✓
Main	Vref	✓

0 pins not mapped

Map

Unmap

Pad	Gate	Pin
1	Main	Vcc
2	Main	Vref
3	Main	Vin
4	Main	GND
5	Main	~CS
6	Main	SDO
7	Main	SCK
8	Main	Fo

0 pads not mapped

2.2 Snadné vytváření součástí

Jednoduše přidejte vývody tak, jak jsou uvedeny v katalogovém listu součástky:

Pin	Name	Direction	Alt. Names	Gate
1	RESET	Input		Main
2	D	Input		Main
3	CLK	Input		Main
4	~CLK	Input		Main
5	Vee	Power Input		Main
6	~Q	Output		Main
7	Q	Output		Main
8	Vcc	Power Input		Main

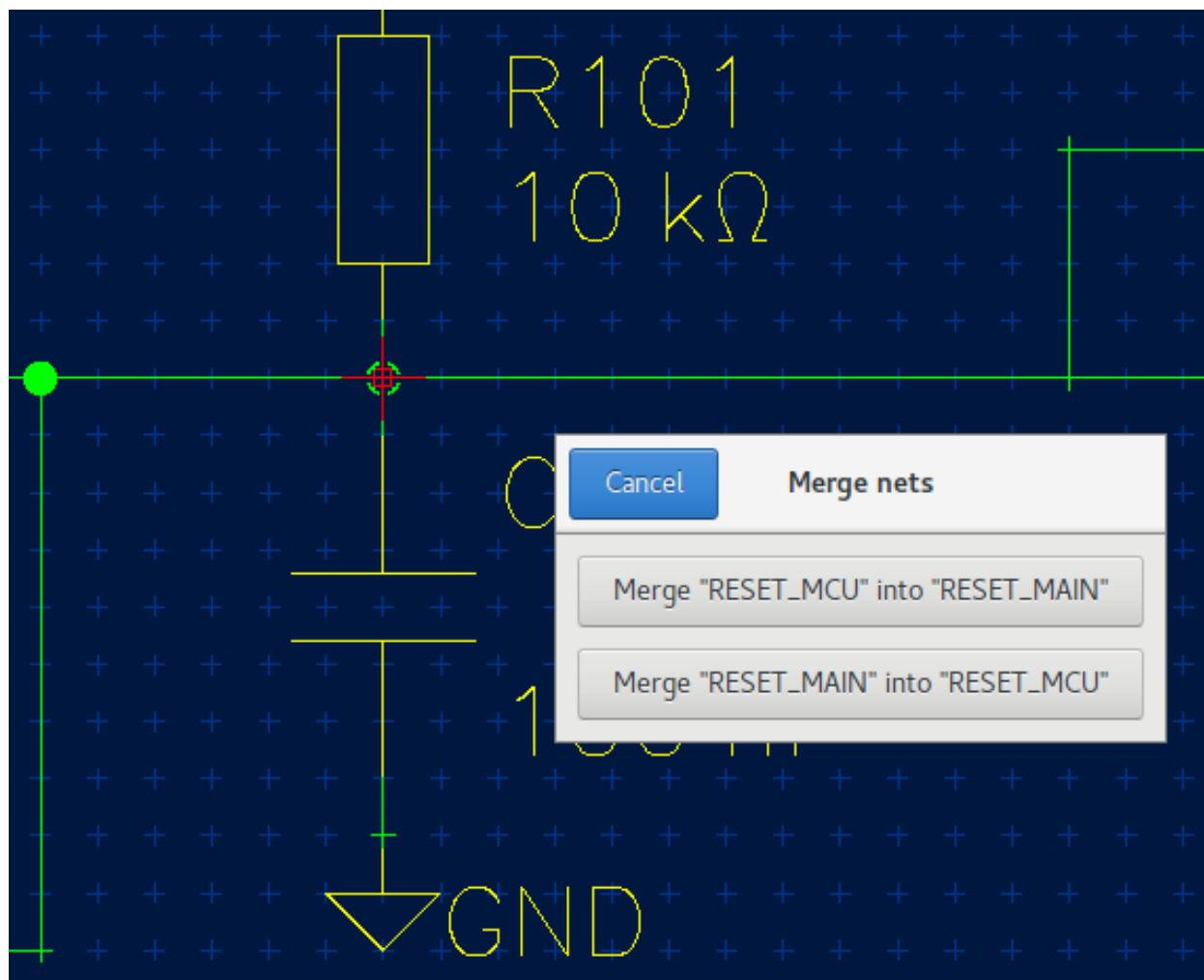
Horizon-EDA se dodává také se šablonami skriptů pro import průmyslového standardu ve formátech jako je IBIS, který vám ušetří zdoluhavou práci při opisování toho, co je v katalogovém listu.

2.3 Loves beginners and power users alike

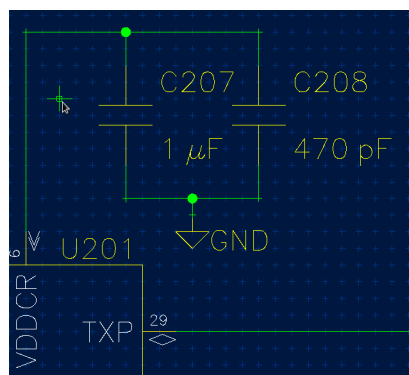
Just press the spacebar and get a list of all the actions you can perform. These actions can be bound to customizable single key shortcuts or to vim-like multi key sequences.

2.4 Schematický editor, který ví co děláte

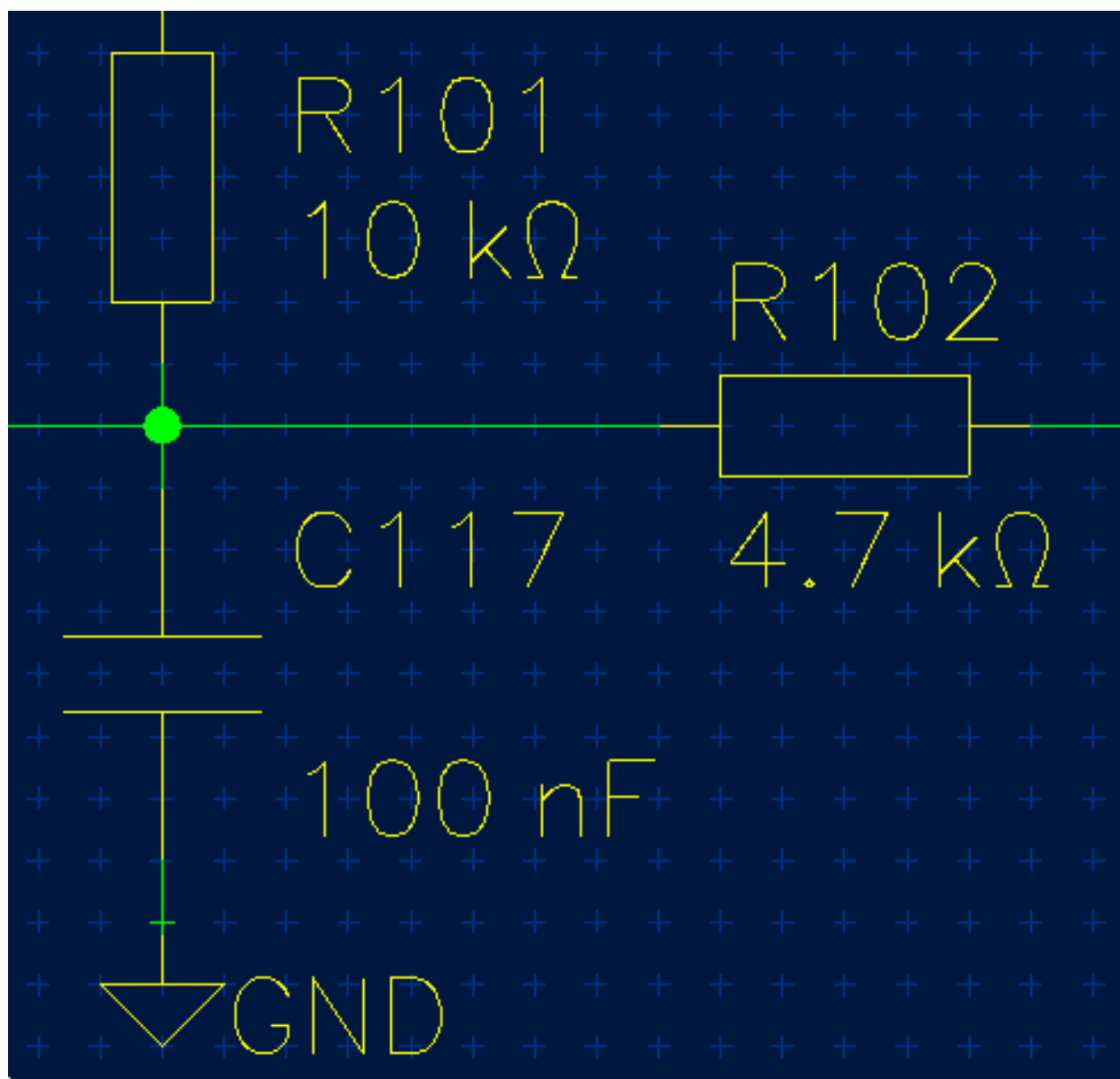
Kreslení schemat není jen o linkách a štítcích. Schéma v editoru Horizon EDA ví o jednotlivých spojkách a zeptá se vás na jejich sloučení:



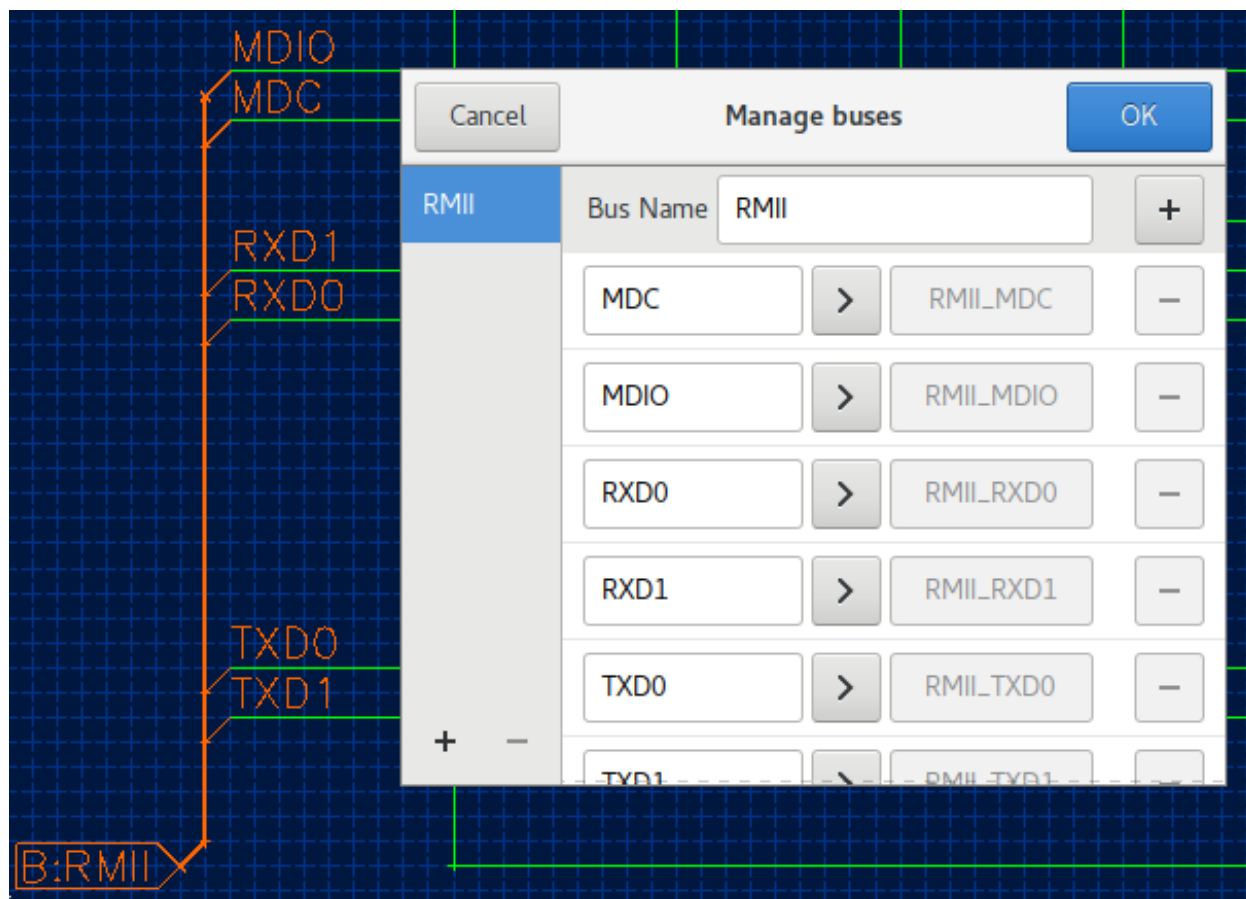
Umístí uzly tam, kde by měly být:



Také automaticky přeorientuje texty, takže se tím vyhnete těžko čitelnému referenčnímu označení:

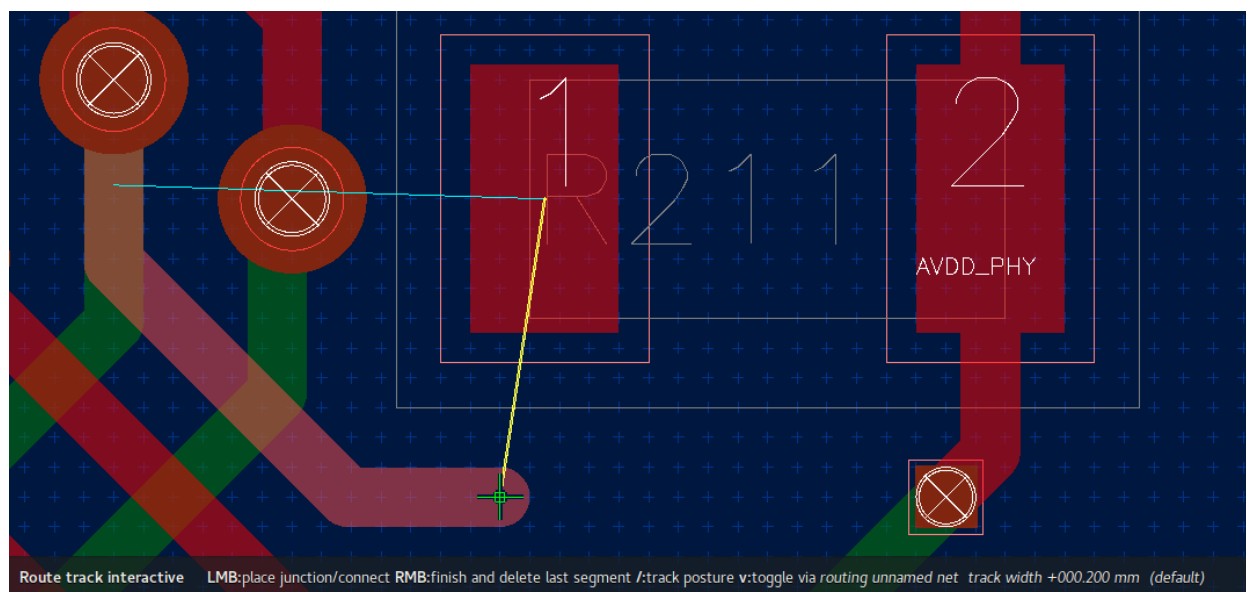


Ani sběrnice nejsou Horizonu cizí:



2.5 Interaktivní trasér spojů s přímou kontrolou pravidel (DRC)

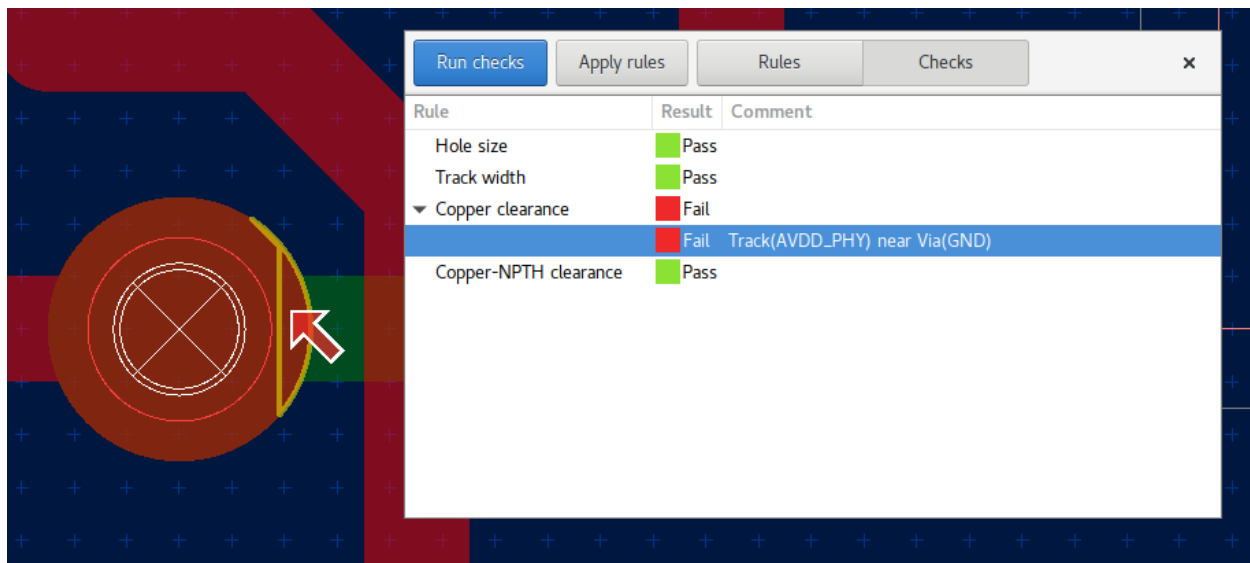
Pomocí interaktivního traséru (routeru) původně vyvinutého pro KiCad, se vytváření spojů stane hračkou. Samozřejmě respektuje vaše pravidla návrhu.



2.6 Silná pravidla

S výkonnými a flexibilními pravidly může horizont kontrolovat a upravovat váš návrh splňující nastavená kritéria:

Pokud něco nesplňuje vaše požadavky, horizont přesně řekne, co to je a na kterém místě:



2.7 Průmyslové standardy výroby

Až bude váš návrh připraven k výrobě, jednoduše exportujte ve formátu průmyslového standardu Gerber a NC-vrtání ve formátu RS-274X:

Generate
Fabrication outputs

Gerber Layers

☒ Outline Filename

☒ Top Paste Filename

☒ Top Silkscreen Filename

☒ Top Mask Filename

☒ Top Copper Filename

☒ Bottom Copper Filename

☒ Bottom Mask Filename

☒ Bottom Silkscreen Filename

☒ Bottom Paste Filename

NC-Drill

Mode

Drill filename

Gerber Settings

Outline width

Output Settings

Prefix

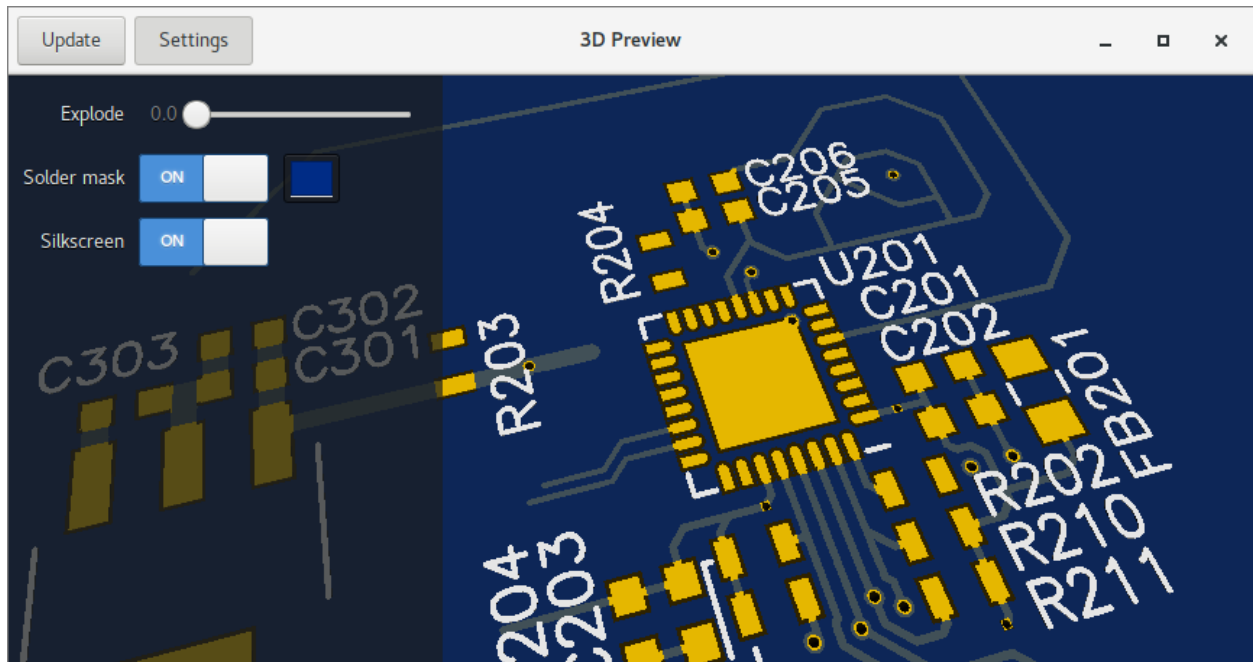
Directory

Gerber and NC-drill files will get prefixed and placed in the directory specified above.

Wrote layer Bottom Paste to gerber file /tmp/gerber/apollo-main.gbp
Wrote layer Bottom Silkscreen to gerber file /tmp/gerber/apollo-main.gbo
Wrote layer Bottom Mask to gerber file /tmp/gerber/apollo-main.gbs
Wrote layer Bottom Copper to gerber file /tmp/gerber/apollo-main.gbl
Wrote layer Top Copper to gerber file /tmp/gerber/apollo-main.gtl
Wrote layer Top Mask to gerber file /tmp/gerber/apollo-main.gts

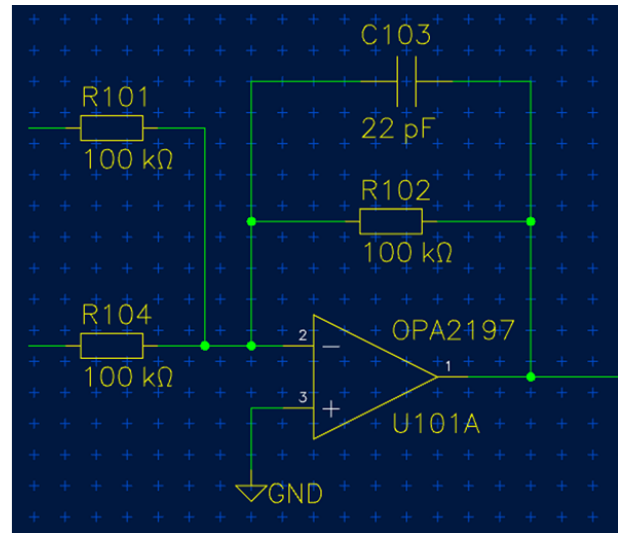
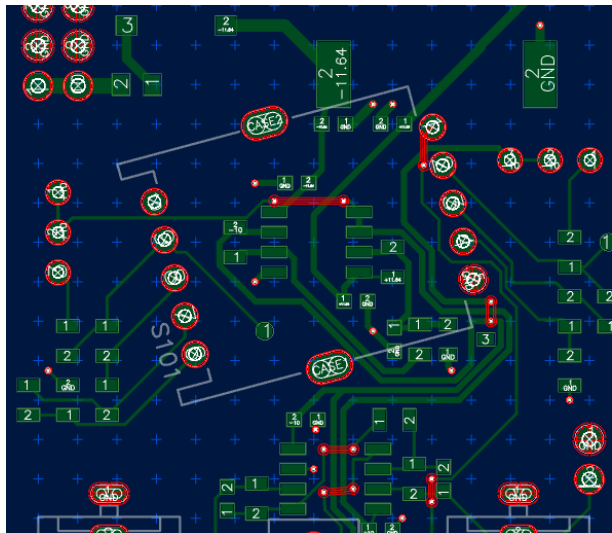
2.8 3D pohled

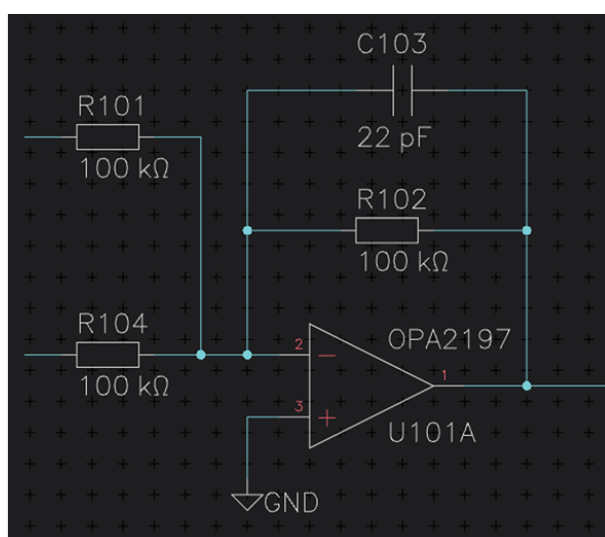
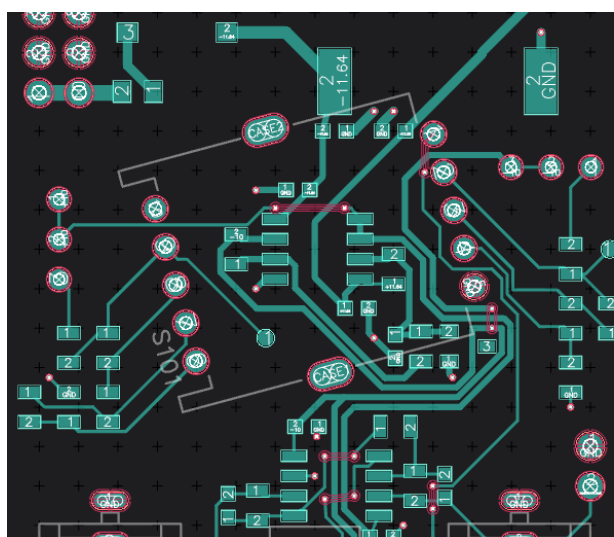
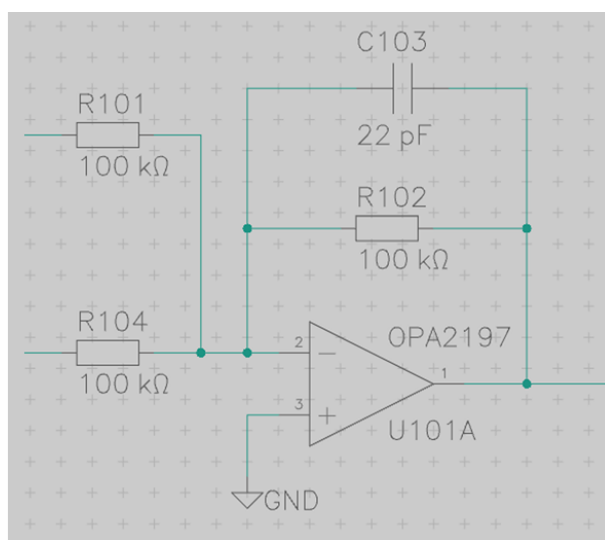
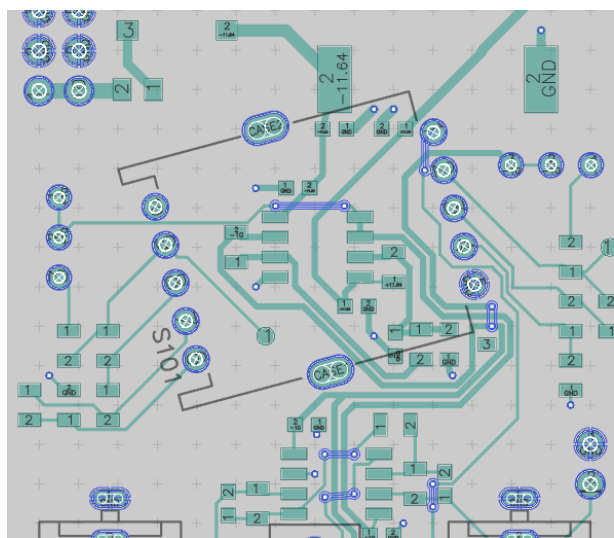
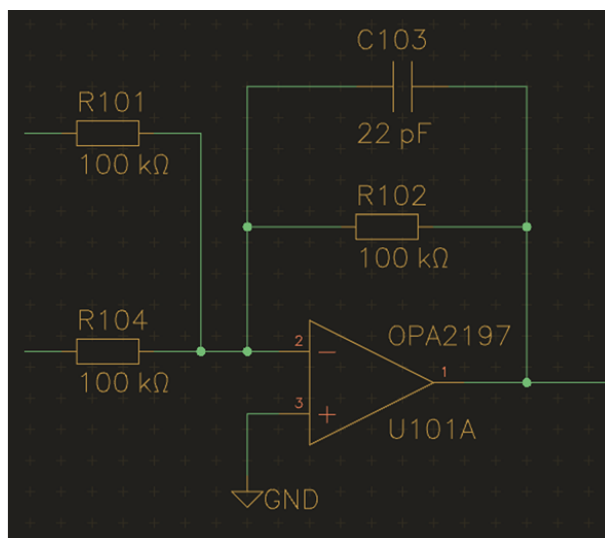
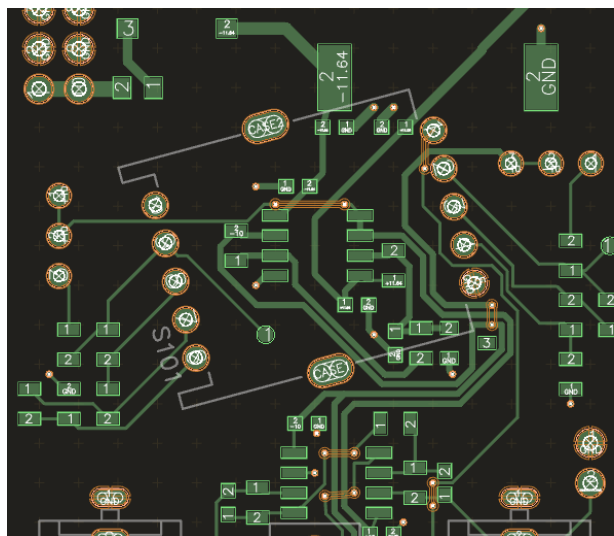
Podívejte se na svou desku, jako byste ji drželi ve svých rukou. Za použití funkce rozpadu (Explode), můžete vrstvy zobrazit oddělené.

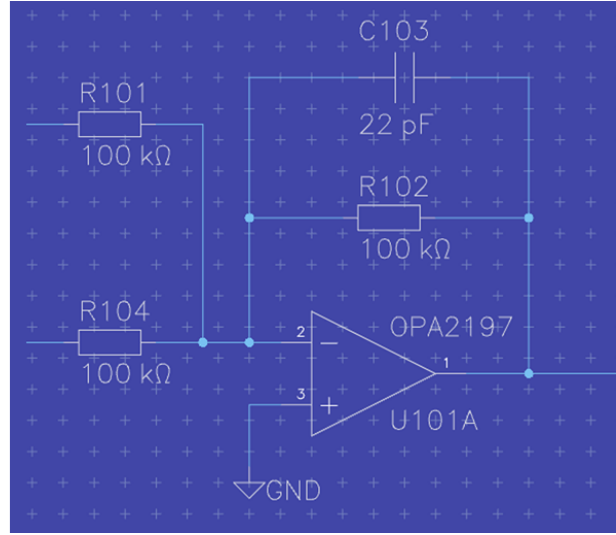
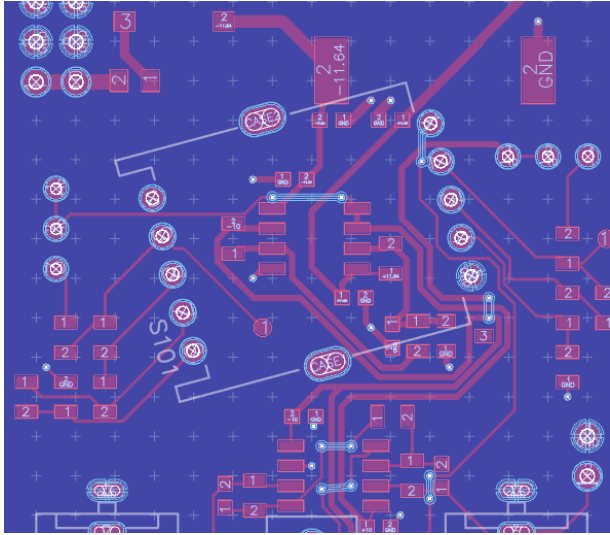


2.9 Customizable color schemes for both schematic and board editors

Decoration affects people, and people are different – do your own thing or select from the existing color schemes.







2.10 Je toho mnohem víc

- Akcelerované zobrazování (OpenGL)
- Neomezená možnost příkazů zpět / opakování
- Kopírování / vkládání, a to i mezi instancemi
- Vyplněné plochy spojů
- Spoje diferenciálního páru
- Libovolné tvary pájecích míst

To limit the project's focus, some things are explicitly out of scope.

3.1 Autorouter

Writing a good autorouter is a lot of work that's more well spent on other aspects of the application as experience has show that an autorouter is rarely useful for small to medium-sized boards.

3.2 Simulation

Schematic design for PCBs and schematic design for SPICE-type simulation are very different as in that schematics for simulation will often simplify aspects of the real world such as replacing an ADC input with it's equivalent circuit. On a personal side I'm perfectly happy with LTSpice in terms of user interface and SPICE core and don't see much scope for new developments in that space.

3.3 Raytracer

Other EDA applications recently gained a custom raytracer for rendering pretty 3D visualisations of circuit boards. For horizon that's out of scope as the OpenGL-based 3D view is pretty enough for checking for the board for issues such as forgotten solder mask and getting an idea of what it'll look like assembled. Any more pretty visualisation is best taken care of by exporting to 3D modelling software such as blender.

3.4 On File formats

Many people complain that there's no commonly agreed on standard format for schematics and boards in the industry. The file formats for these are application files formats, meaning that they'll need to support each and every knob

and button the application has. Adopting another application's file format for horizon EDA would therefore result in horizon EDA being a bad clone of the other application.

JSON has been chosen as a serialization format as it directly maps to common data structures such as maps and arrays (opposed to XML) and is easily manipulated in almost every environment.

Next: *Installation*

So you wanna give horizon a test drive? Great! Here's how.

4.1 Windows

Grab the latest build from [AppVeyor CI](#) and unzip it somewhere. Note that these are 64bit binaries. In case a build is still running or someone broke the build, you can download past builds from [the build history](#) (click on artifacts to get the zip)

4.2 Linux

See *Sestavení (kompilace) programu na operačním systému Linux* for instructions on how to build horizon on linux.

Next: *Setup a pool*

Setting up a pool

There are two ways of setting up a pool. If you are familiar with the version control system git and want to be in direct control of your repository you can use the git workflow below, if not, just use the pool manager.

5.1 Git

If you're familiar with git, just clone clone [horizon-pool](#) somewhere and you're good to go. You're supposed to use git to keep your local copy up to date and submit new parts.

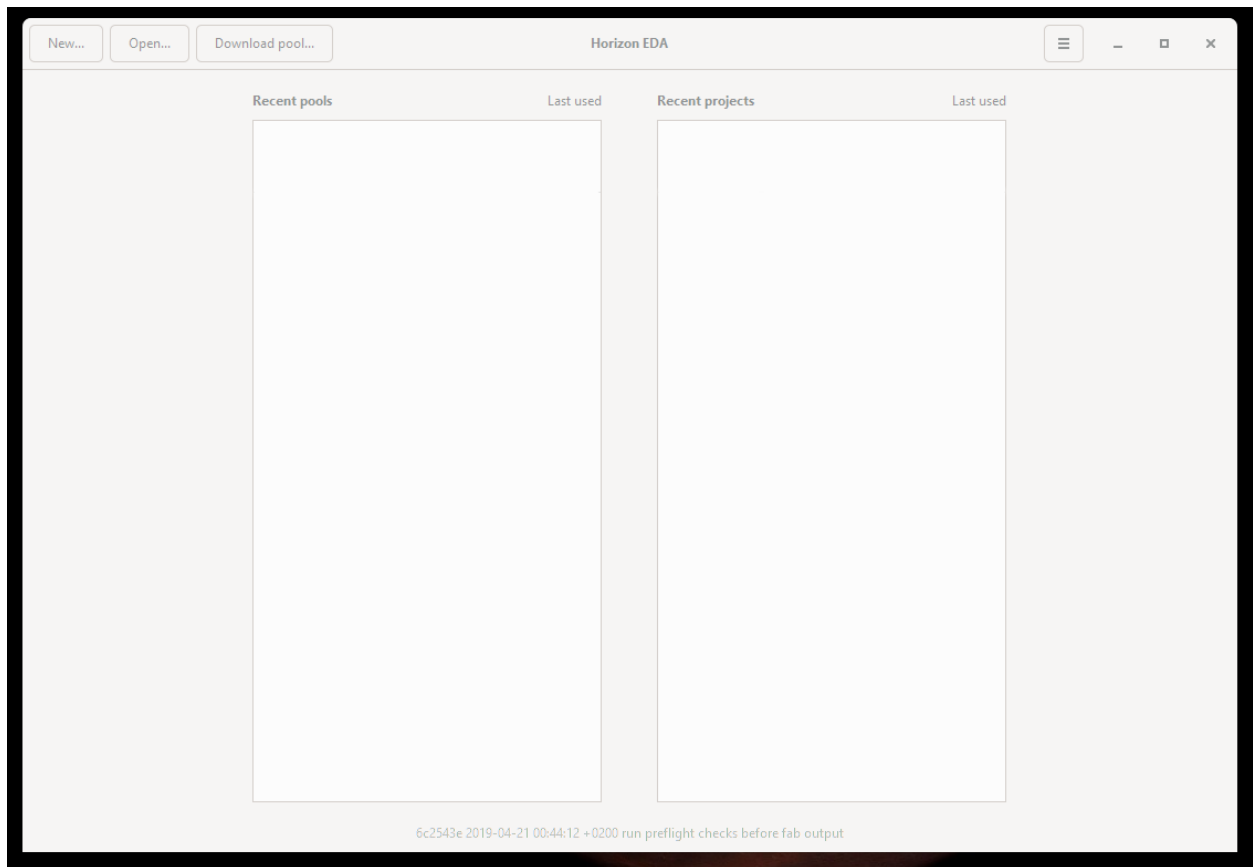
5.2 Pool manager

Don't know how to git? No problem! Double-click `horizon-eda.exe` or launch `./horizon-eda` from your shell and click on ,Download...' to download the pool. The default pool `horizon-eda/horizon-pool` is the one you want to use. The pool manager will assist you in keeping your pool up-to-date, see the „Remote“ tab. It will also assist you by creating a fork, branches, commits and pull requests on your behalf so you can contribute to the pool without any git knowledge.

Next: *Create a new Project*

Create a new Project

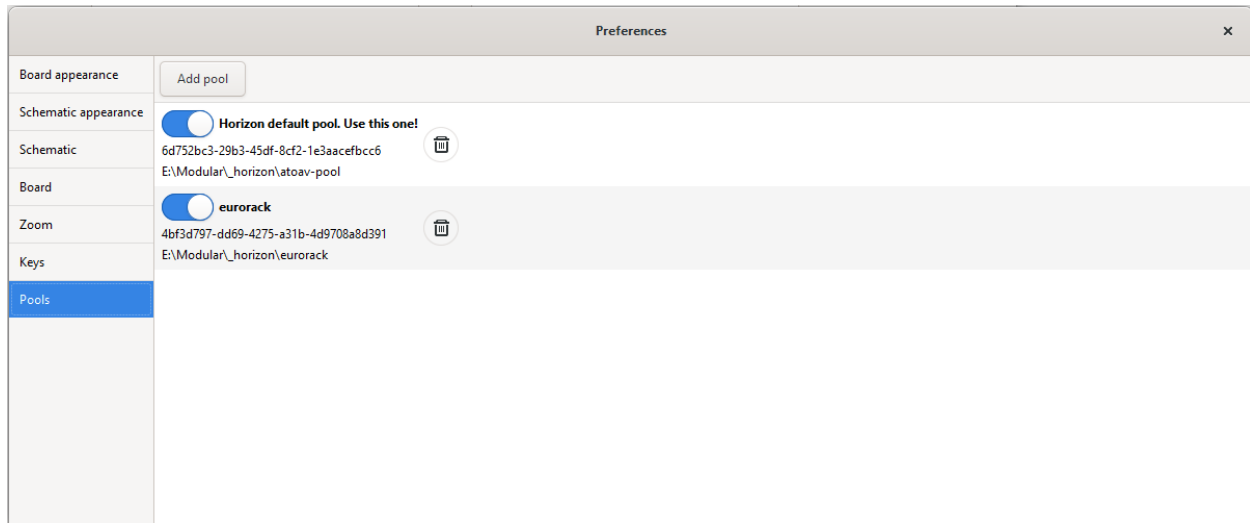
When you start Horizon EDA you should see a window like this one



Make sure you have a pool set up and added

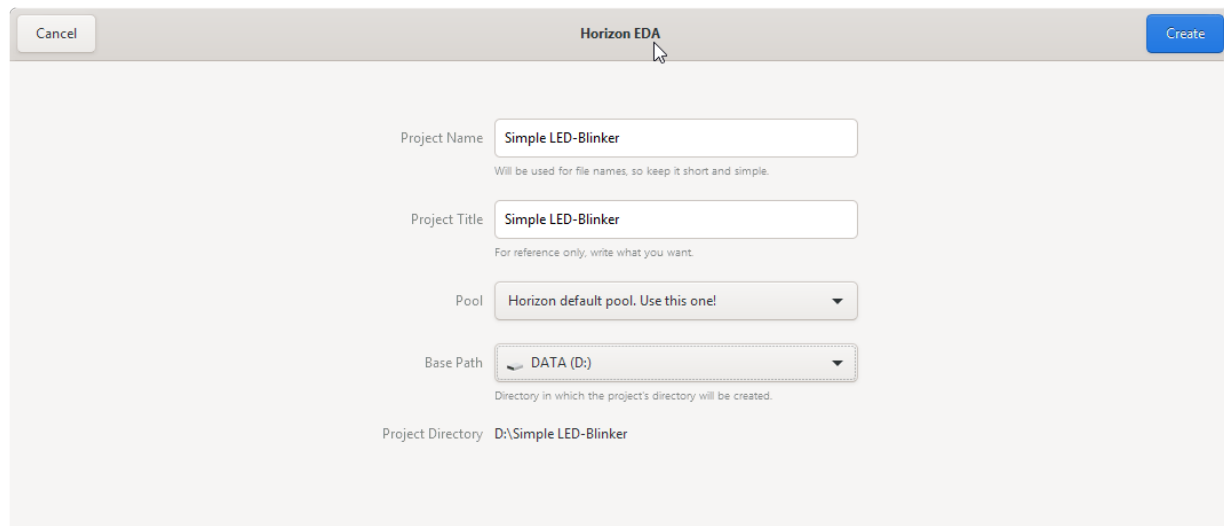
Click the menu icon in the top right corner and open the preferences dialog. In the Pool section selected the pool

you want to use for your project. If you don't have a pool here you can open a `pools pool.json` by clicking on „Open...” in the main window. When you're done the preferences dialog should look like this:

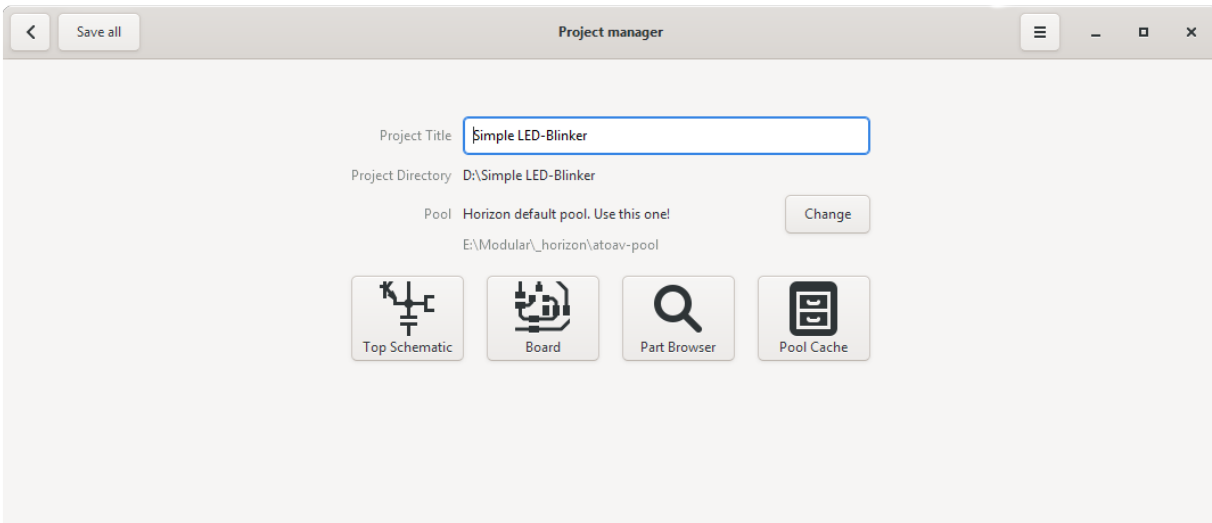


Now, hit „New...” and then „Project” to create a new project.

In the Project Dialogue Window you can select a project name and a location where you want to store the project folder. Additionally you can change the active pool:



In the Project manager window that opens now, you can create a Schematic, a Board, Browse for Parts or Manage the Pool Cache.

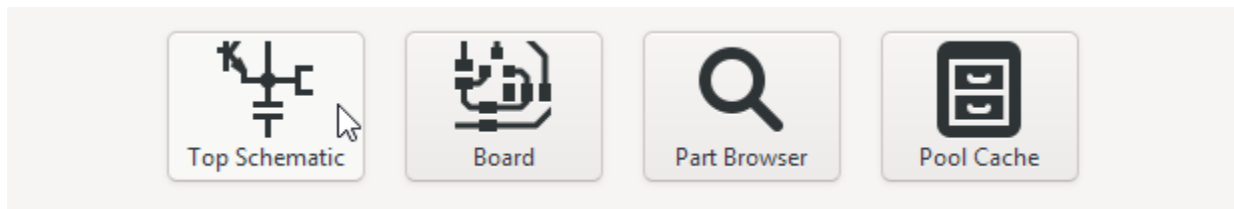


The Pool Cache stores a copy of all parts that you used in your project and helps you to protect your projects from outside changes (e.g. by updates in the Pool). When you *want* to update the parts, or remove unused parts you can do so in the Pool Cache Window.

Next: *Draw a Schematic*

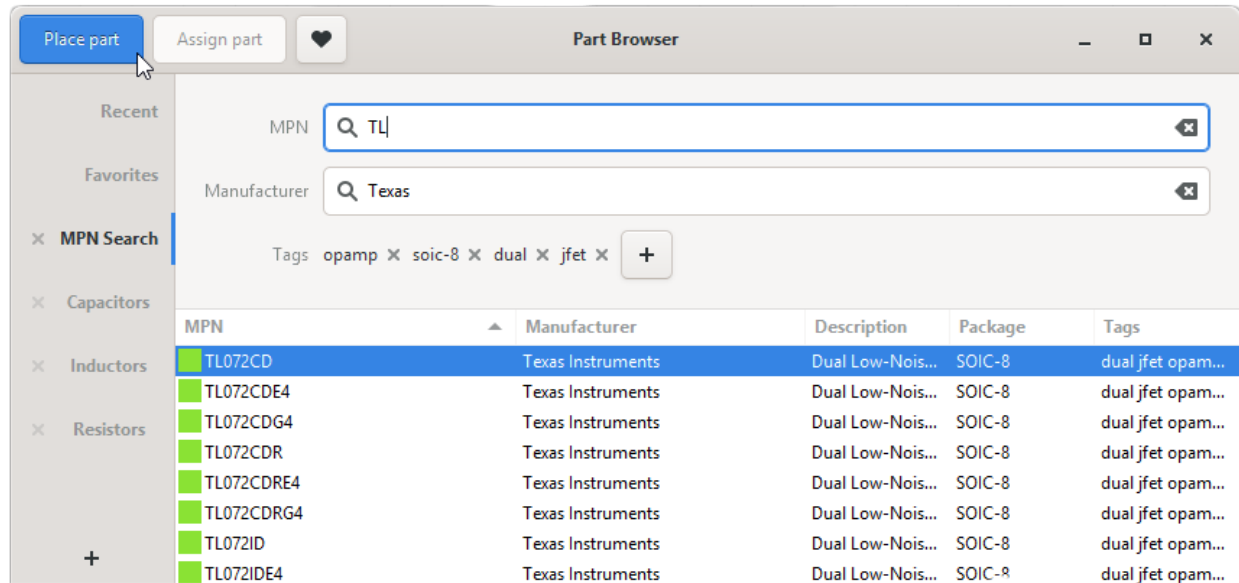
Draw a Schematic

In the Project Manager click on the Top Schematic button, to open the Schematic editor:



7.1 Placing Parts

You can then start placing parts by clicking the „Place Part“ button in the top bar, or by simply typing `p p` (remember it as „Place Part“). This opens the Part Browser, where you can search your whole pool for the Parts you need and place them on the schematic.



7.2 Connecting Parts

Once you placed parts you can activate the „Draw net line“ tool by typing `n` and drawing the connections. You can also simply drag out a pin and begin connecting pins:

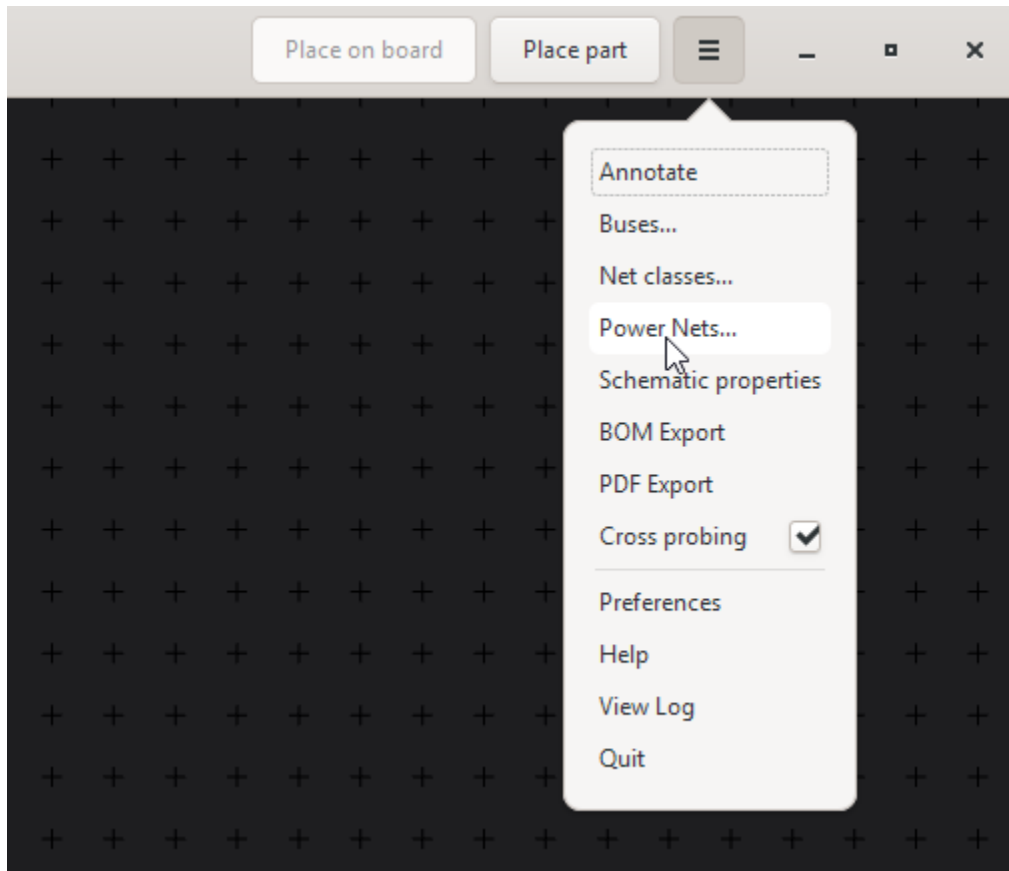
7.3 Basic Movement

You can move parts and nets by selecting them and typing `m` or by using the `←/↑/↓/→` arrow keys. Rotate with `r` and mirror with `e`

If you forget any of these keys, just press `Spacebar` to open the *Spacebar Menu* and search for the command you are looking for.

7.4 Power Nets

Because Circuits make much more sense if you use power nets you can create new power nets by clicking on the entry in the application menu:

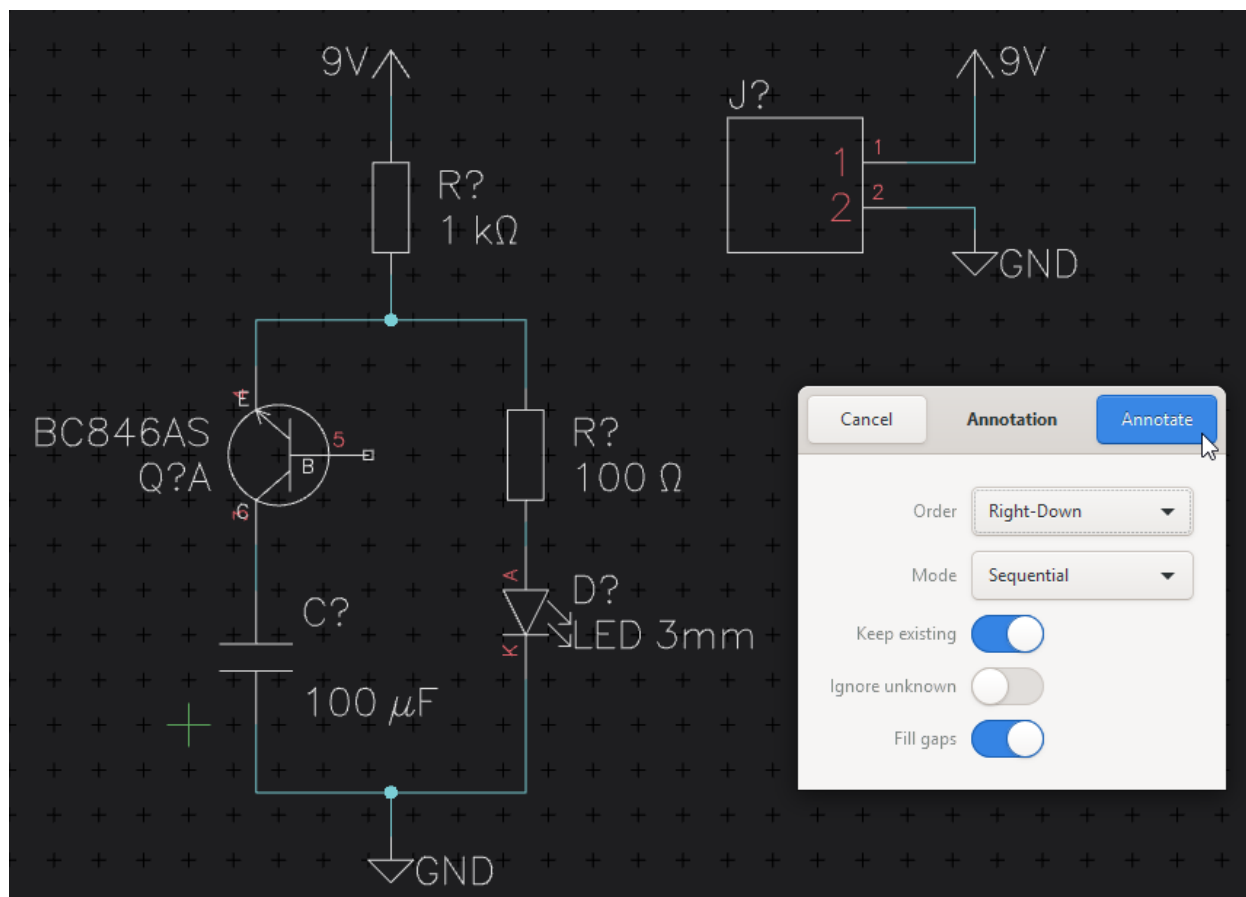


In the window that pops up press the „Add power net“ button, give the net a name and select a symbol style.

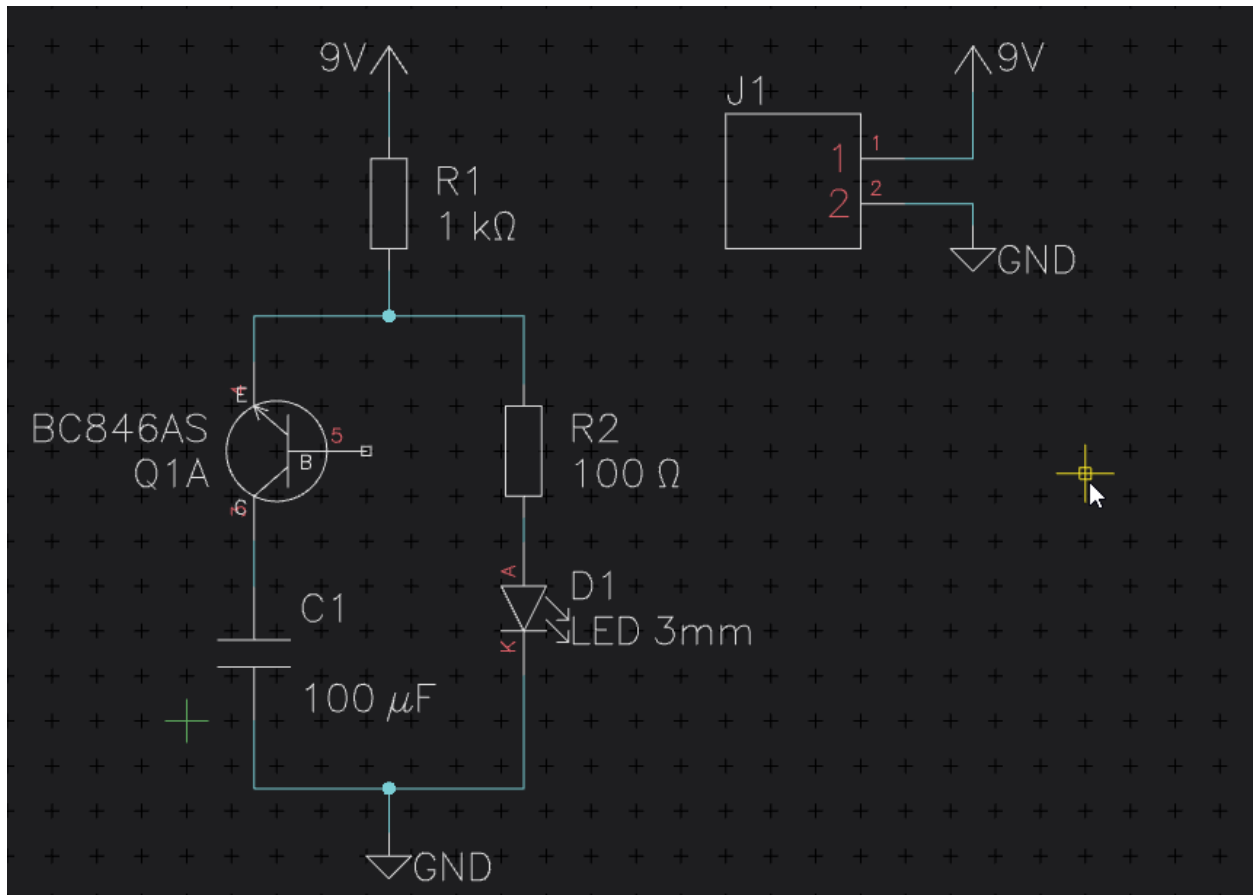
Place power nets with the „Place power symbol“ Tool (type p o).

7.5 Annotation

Run the Annotation Tool to give all un-annotated parts (signified by the „?“ in their reference designator) a proper number. So we go from this:



to this:

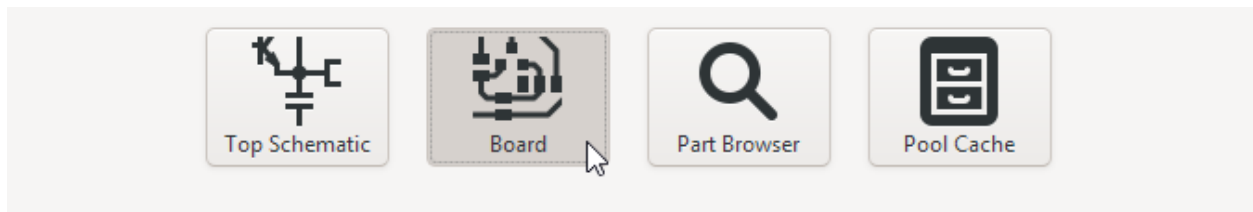


Once you are done, click on „Save“ so you can start to layout the schematic you just created.

Next: [Create a Board](#)

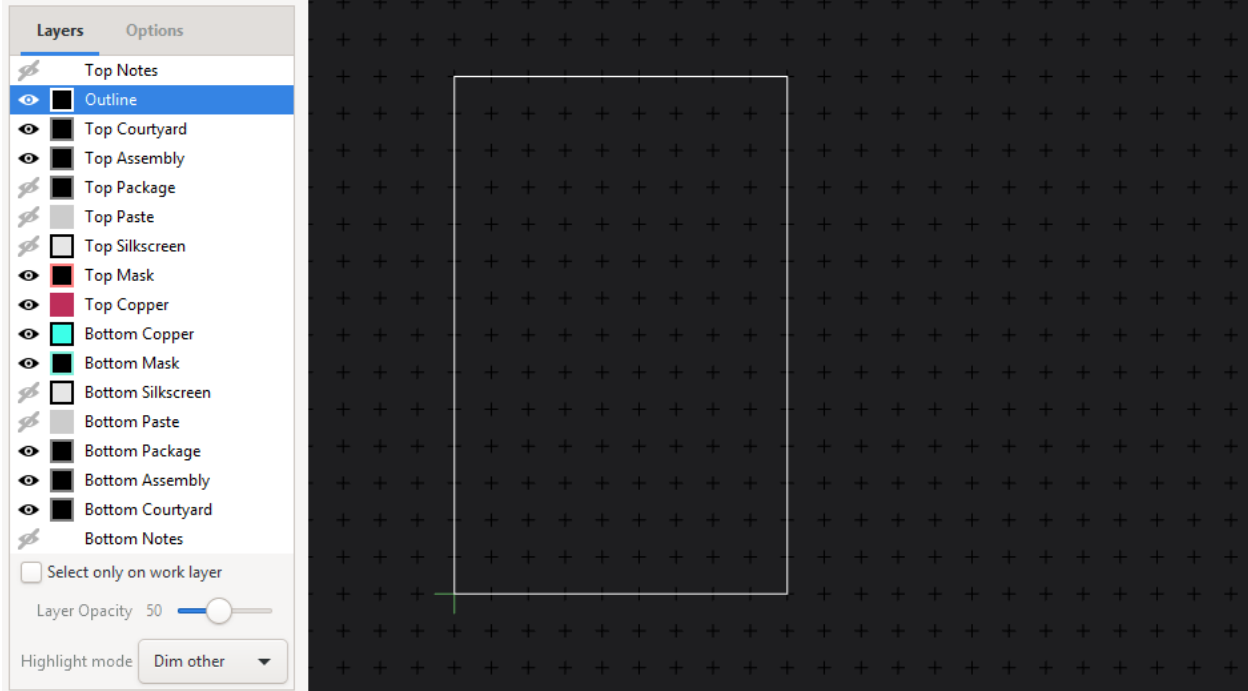
Create a Board

Open the board editor from the Project manager.



8.1 Draw a Board Outline

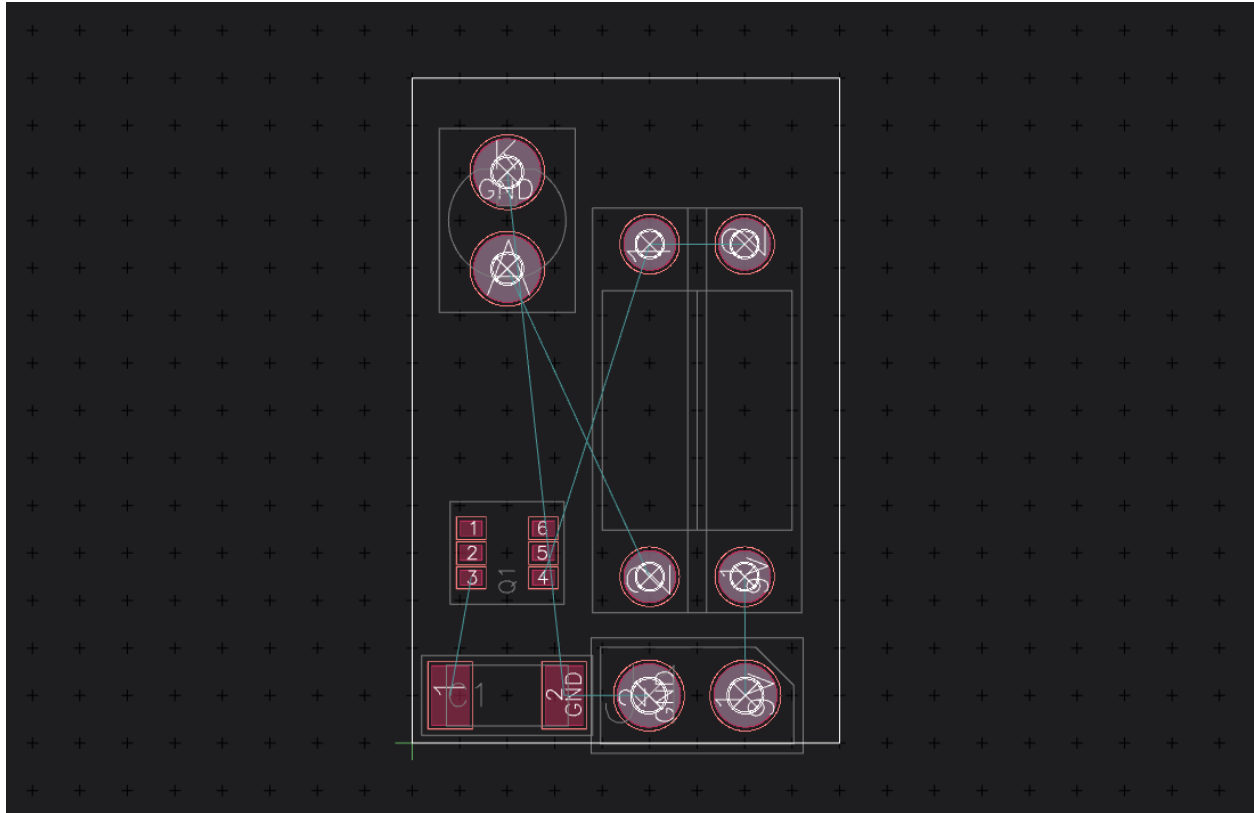
Draw a outline by first selecting the „Outline“ Layer in the Layers Panel and then selecting the „Draw polygon rectangle“ Tool by typing `d Y` (or use the Spacebar menu):



8.2 Place the Packages

Place packages on the board using by typing `p p` (this time it stands for „Place Package“). If you still have the Schematic Editor open you can also select a Symbol and use the „Place on Board“ Tool by clicking it's button in the top bar.

If you select Parts in the Schematic Editor they will also be highlighted in the Board Editor. If you change something in the Schematic Editor bring the changes from the schematic to the board, by saving the schematic and clicking ‚reload netlist‘ in the board editor.

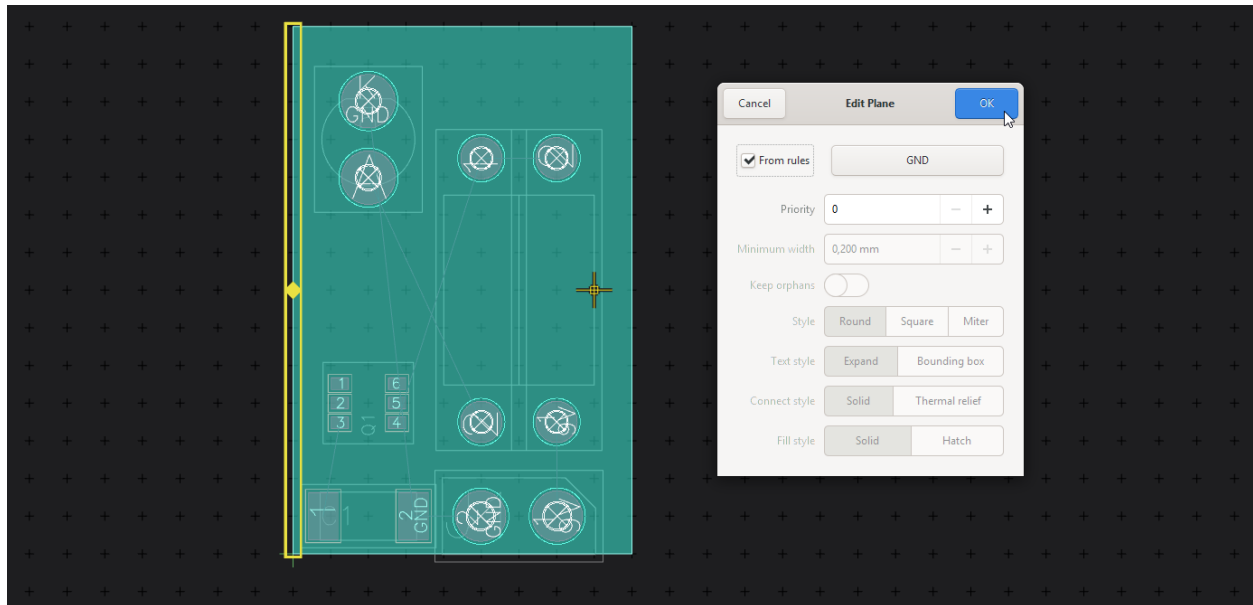


If you want to place a part on the bottom of the board, press **e** to flip it to the bottom side. If you are working a lot on the bottom you can use the „View bottom“ tool to flip the whole board around.

8.3 Add Planes

If you want to use a Ground Plane now would be a good time to add it:

1. Select the layer you want to have the plane on (Top- or Bottom-Copper)
2. Draw a polygon rectangle by typing **p** **y**
3. Right-Click one of the Edges of the newly added polygon and select „Add Plane“
4. In the Window that pops up select the GND net (if it is not there you didn't add the power net in the Schematic)



If the solid filled color of the plane annoys you, you can switch the display style to something else by clicking onto the colored square left of the layer:

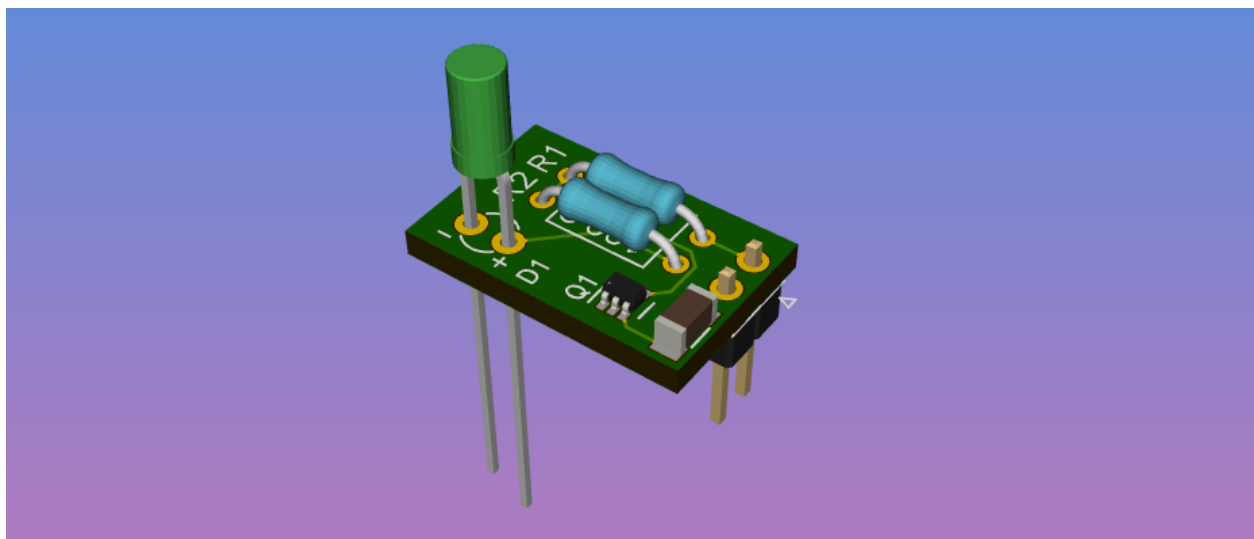
8.4 Route the Nets

To route tracks between the parts you connected in the Schematic, type **x** and drag from any pin that starts an airwire:

The track will always be routed on the layer you have selected, you can quickly select the Top Copper Layer by pressing **1** and the Bottom Copper Layer by pressing **2**.

If you want to change sides in the middle of a track you can place a via by pressing **v** and then the number of the layer you want to continue.

Once you are done, check out the 3D view of your part, and export Gerbers using the menu point „Fabrication Output“ in the application menu.



Next: *Look at a Example Project*

KAPITOLA 9

Example project

Instead of starting your own project, you can also download the [design files for an X-Band transmitter](#). To open it, point the project manager to the `ddstx.hprj` file. Make sure that you extract all the files contained in that repository.

Next: *Basic Editor Usage: Tools*

In order to provide a unified user experience and enable code reuse, the editors for symbols, schematic, padstack, package and board are all based on the interactive manipulator.

10.1 How to select the right tool

To edit the things on screen, use the tools. Tools can be started in multiple ways:

- By typing in the tool's key sequence. Available key sequences are listed by clicking the help button in the top right corner or typing ?.
- By using the *Spacebar Menu*. Just start typing for the tool you're looking for or browse through the list.
- By right clicking an object. This will bring up context menu listing all the tools relevant to what's selected.
- By pressing one of the dedicated buttons in the top bar.

After having started a tool, it receives all keyboard and mouse input, till the tool is finished or you end it by pressing `ESC`. Look at the bar that appeared at the bottom of the canvas to see what pressing keys will do in this particular tool.

Next: *Basic Navigation: Spacebar Menu*

Spacebar Menu

Horizon's Editors rely on fast and intuitive single key shortcuts and key sequences alike. While this is very powerful for those who use it daily, it could be cumbersome for those who just started out or have long periods passing between each of their electronics projects.

In order to make the transition between beginner and power user an easy one, Horizon EDA has a Spacebar Menu, that (hence the name) pops up after you pressed `Spacebar`. In this menu all Tools you can select within the Editor's Canvas are listed. To help you getting faster the shortcuts and key sequences are listed alongside the tools. So if you find yourself in the position of having to go to the space bar menu over and over again, you can easily speed up things, by using the short cuts.

You can easily customize these shortcuts in the preferences window.

Next: *Grid*

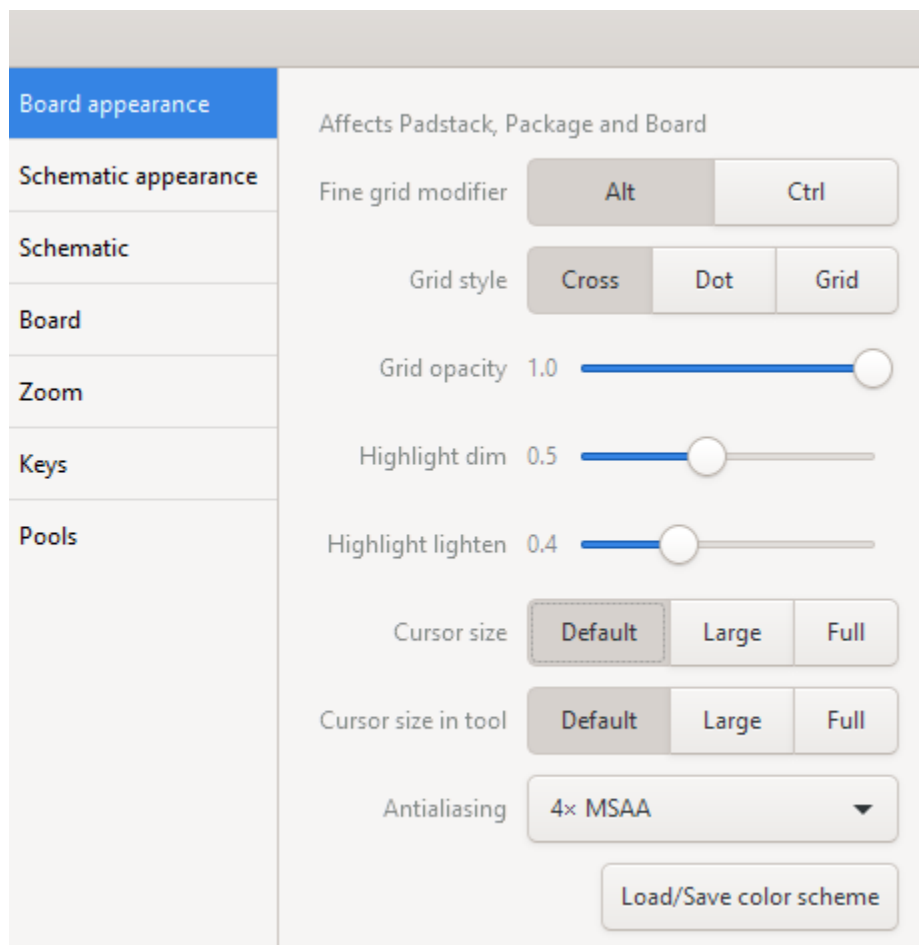
Grid

All movements in Horizon happen on the grid. By holding `Alt` you get a finer grid (by a factor of 10).

You can change the size of the grid by changing the value in the numeric field on the top left of the window. Note that depending on a zoom level your grid may be resized by factors of two.

You can enter basic math operations into *any* numeric field in horizon, this makes it easy to divide a value by two, multiplying something by a certain factor, or adding a value to a coordinate to create a fixed offset. For more details, see *Numeric entries*.

You can change the appearance of the grids and the cursor in the Preferences:

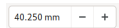


Next: *Drawing*

Numeric Entries

All numeric entries in horizon support basic two-operand math as well as some other goodies.

They look like these:



Both point and comma are recognized as a decimal separator regardless of locale settings. By default all numbers are treated as millimeters. Suffixing a number with an `i` will treat it as inches.

Additionally, two-operand infix math is supported, so you can do this:

Addition: $1+2$

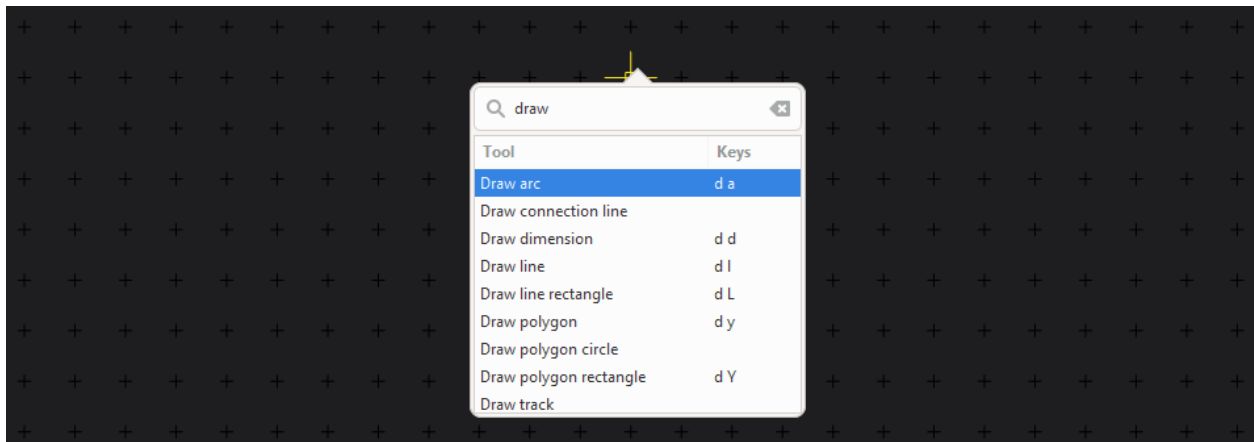
Subtraction: $1-2$

Multiplication: $3*2$

Division: $3/2$

Average $\frac{a+b}{2}$: $3|2$

The different Editors in Horizon EDA share a set of different Drawing tools. Here you can see the ones available in the Board Editor:



The drawing tools are mainly split in three categories:

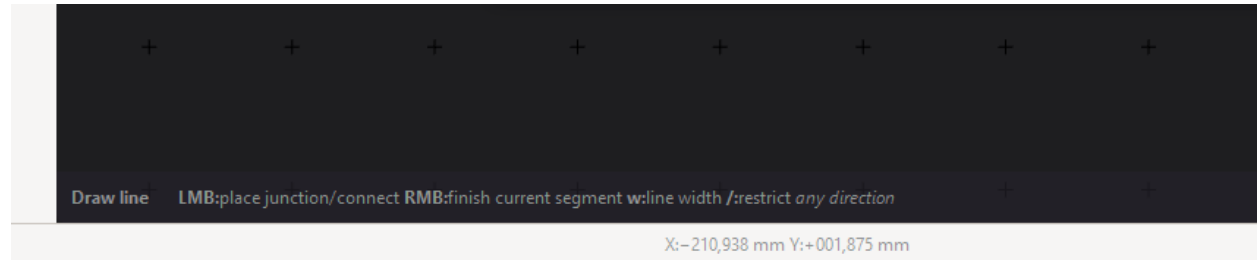
- lines
- polygons
- special (like draw track, draw dimension etc.)

Lines are usually used for everything visual (e.g. Silkscreens) while polygons are used for all things where it matters that the thing you draw results in a closed shape (pads, board outlines, package assembly and courtyard layers, etc.)

14.1 Draw Line

To draw a line simply select the „Draw line“ Action in the spacebar menu or type the key sequence `d l` (think: „draw line“) – once you click anywhere, you start the first point of the line right at the place where your manipulator was. You might notice that the point snaps to the grid. If you want a finer grid hold the `Alt` key down while placing points. Left mouse button places more and more points, while the right mouse button (or pressing `ESC`) will finish the lines.

You might also have noticed the Action Bar at the left bottom of the editor window:



The Action bar will show additional keys you can press to change the behaviour of the tool. In this case you could press `w` to change the width of the stroke or `/` to restrict the movement of the manipulator to one direction.

14.2 Draw Line Rectangle

To save your time, there is also a „Draw line rectangle“ Action, which can also be invoked by typing `d L`. Per default you first set the rectangles center point and then one of the corner points. By pressing `c` you can change this behaviour and set two diagonally opposed corner points instead.

14.3 Draw Arc

The „Draw arc“ tool is straightforward it draws line arcs by setting three points (in this order): start point, end point and the center point. You can also use the key sequence `d a` to start the tool. If you want your arc to flip direction, press `e` before putting down the center point.

14.4 Draw Polygon

When drawing polygons with `d y` you can set a series of points by clicking, until you either press `ESC` or use the right mouse button. You can make the next edge of the polygon an arc by pressing `a`. Just like with the „Draw Arc“ tool set the endpoint first and the center point after. Before setting the center point you can flip the arc direction with `e` and finally you set the end point of the arc.

14.5 Draw Polygon Rectangle

Similar to the „Draw line rectangle“ Tool there is a „Draw polygon rectangle“ tool. Invoke it by typing `d Y`. Just like with the according line Tool you can switch between the different draw modes (Center/Corner) by pressing `c`.

There are some differences though: you can set a corner radius by pressing `r` and entering a value and you can choose a decoration by pressing `d`. These decorations are used to mark the pin 1 on a Package's assembly layer. You can cycle

through different decoration positions by pressing `p` and set the size of the decoration by pressing `s` and entering a value.

14.6 Draw Polygon Circle

For ease of use there is also a „Draw polygon circle“ Tool. With the first click you set the circles center point and by setting the second point you set the radius. You can also enter a radius by pressing `r` and entering a value.

14.7 Draw Dimension

Sometimes it can be useful to add dimension information to certain parts. You can do so by using the „Draw dimension“ ActToolion. Start it by typing `d d`, selecting the first and the second point and dragging it out. If the numeric value is on the wrong side, you can fix it by selecting the dimension and flipping it with the `e` key.

Dimensions can also be set to specified length by selecting the end that's supposed to move and activating the „Enter Datum“ tool (press Enter). You can then snap other items to the end points of the dimension.

Next: *Selection*

Selection throughout the Editors of Horizon EDA is very straightforward and there are no surprises here. No matter how basic it seems, the different modes and filters could save you quite some time.

15.1 Basics

Initially, the „hover select“ mode is active. It simply selects the smallest object under the cursor. Leftclick or drag with the clicked button to select objects permanently. Hit `Esc` or click into a blank space for returning to hover select mode.

If you want to select multiple things by clicking keep Hit `Ctrl` pressed while you click on things – this also works to deselect things you accidentally selected while dragging a selection box.

15.2 Selection Mode

You can change how dragging the left mouse button behaves in the lower left corner of the editor window. Available are three different modes (Box, Lasso, Paint) with four different selection characteristics (Auto, Include Origin, Touch Box, Include Box).

15.3 Selection Filter

If you have to select many things, it can sometimes be handy to only select certain classes of objects. This is what the selection filter is for. You can open it up via the spacebar menu or by pressing `Ctrl+i`

Double click on an item to select only that item. Click on the check mark button on the top left to select all items.

Next: *Moving and the interactive Manipulator*

Moving and the interactive manipulator

16.1 Move with Mouse

You can move parts and nets by selecting them and typing `m`, press `Alt` to move on a fine grid. The movement of the selected object has its origin (or pivot) point right at the spot where the interactive manipulator was, when you started the move tool.

This can be used to our advantage, because it allows us to move objects that have been placed on a finer grid without snapping them back to the coarse grid.

16.2 Move with Keyboard

By using the `←/↑/↓/→` arrow keys you can move the selected object one grid step into the desired direction. If you press `Alt` at the same time you will move the part on the fine grid (1/10 of the original grid). Because this is also a tool, you can cancel it by pressing `Esc` and finish it by pressing `Enter` or clicking onto a empty spot.

16.3 Move Exactly

The „Move exactly“ tool allows you to enter numbers to move an object by an exact number of units (e.g. 1 mm) into the desired direction. You can start the tool by pressing `M`

16.4 Flip

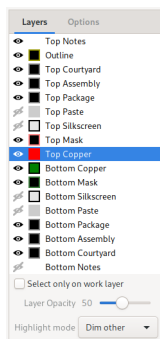
You can flip symbols or parts by pressing `e`. If you use this on the board editor, it is the same as flipping the part onto the other side

16.5 Rotate

You can rotate objects by 90° if you press `r` or select the „rotate“ tool. If you like to rotate a object by something else (e.g. 30° or 45°) use the „Rotate arbitrary“ tool. The pivot point of the rotation is always the location of the interactive manipulator, so make sure it is at the right spot

Layers

Board, package and padstack editor use the widget shown below to specify how layers are displayed.



The selected layer is called the „work layer“ and is always visible and drawn on top of all other layers. Use Page up/down to move the work layer to the next or previous layer. Pressing 1 selects the „Top Copper“ layer, 2 selects the „Bottom Copper“ layer, 3 . . . 0 select inner layers if present.

Clicking on the eye toggles a layer’s visibility. Keep in mind that the work layer is always visible even if it’s set to be invisible.

Clicking on the colored box cycles through a layer’s display modes:

- Solid color: Layer is drawn with outlines and fill, overlapping filled objects on the same layer will appear brighter.
- Solid color with black border: Layer is drawn just filled, overlapping objects look the same as non-overlapping ones.
- Black with colored border: Layer is drawn just with outlines.
- Striped: Same as first, but areas are striped rather than filled.

Chcete-li spustit editor schémat, klikněte v manažeru projektu na „Top Schematic“. Podobně jako jiné grafické editory, schématický editor v Horizon-EDA je založen na Interaktivním manipulátoru. Chcete-li umístit součástky, použijte příkaz „Place part“ nebo prohlížeč součástí v hlavním projektu. Ten je mnohem pohodlnější, protože může být trvale otevřený. Začněte kreslit propojení (vodiče) schematu stisknutím Klávesy „n“.


18.1 Spoje a segmenty spojů (Nets)

Na rozdíl od některých jiných schématických editorů Horizon-EDA si eviduje jednotlivá propojení a nejde jen o nakreslenou čáru (vodič), která se nakonec při generování seznamu spojů (netlistu) transformuje na vodič (spoj). Propojení může být reprezentováno jedním nebo více segmenty vodičů. Segment spoje je sada propojení vodičů, uzlů, vývodů atd. všechny propojené vodiči (čárami spoje). Editor sleduje, které vodiče patří do kterého segmentu a oznamuje, když se operace chystá sloučit dva segmenty.

Když se podíváte na spoj v editoru vlastností po výběru spoje, spoj v editoru vlastností je „celý“ spoj, nejen segment spoje. Proto přejmenování spoje nemění propojení. Pro připojení vývodu na segment jiného spoje, použijte nástroj pro přesunutí segmentu do jiného nebo nový spoj „Move net segment to other/new net“.

Štítek spoje (Net label) pouze zobrazuje název spoje, ke kterému je připojen, ale nemění ho. Upozorňuje vás na nekonzistenci v systému schématu, které by mohlo mít za následek nežádoucí propojení, editor schématu umísť varování na porušené položky.

18.2 Symboly napájení (Power Nets)

Nejjednodušší způsob vytvoření symbolu napájení ve schematu je použít příkaz pro správu napájení „Manage Power Nets“. Příkaz je k dispozici v menu hamburger . Poté použijte příkaz „place power symbol“, pro umístění symbolu napájení pro tento spoj. Symboly napájení „napájí“ jejich vodiče a připojené segmenty. Můžete si vybrat ze tří stylů napájecích symbolů ve výše uvedeném příkazu. Symboly antény a tečky lze umístit buď nahoru nebo dolů. Symbol GND může směřovat pouze dolů.

18.3 Sběrnice (Buses)

Pro seskupení souvisejících vodičů použijte sběrnice (Buses). Po vytvoření sběrnice přidejte vodiče. Můžete buď přiřadit existující vodiče nebo nově pojmenovat automaticky vytvořený vodič kliknutím na tlačítko se šipkou vedle ní.

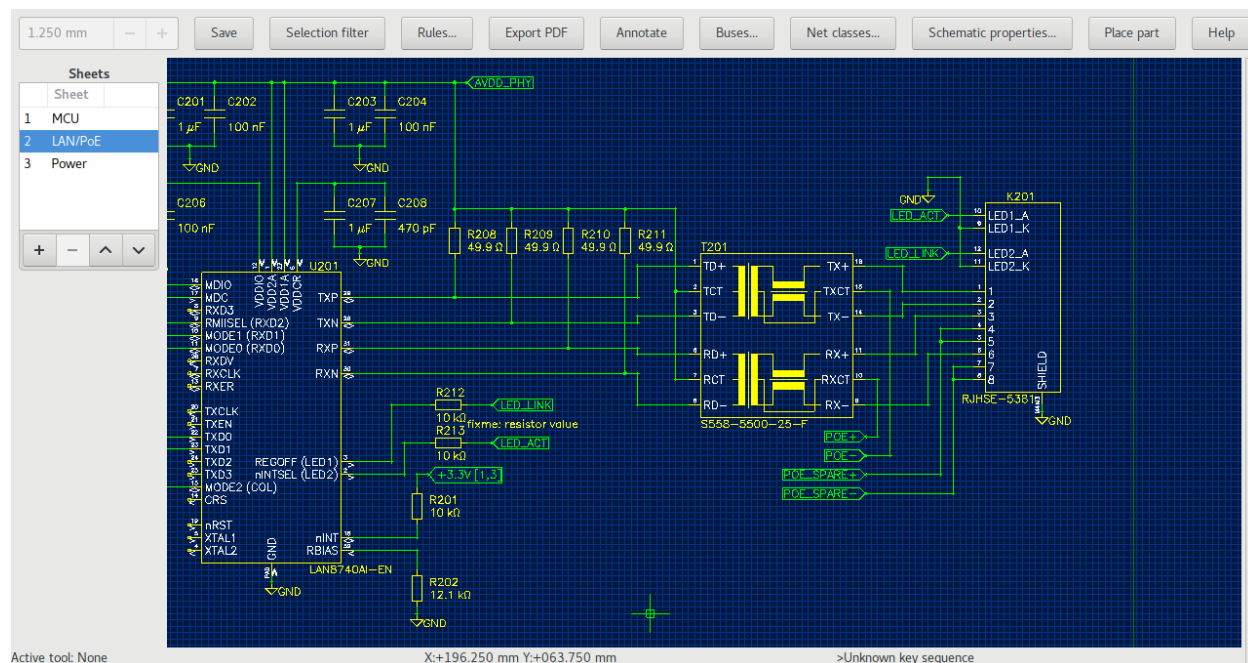
18.4 Diferenciální páry (Diff. pairs)

Chcete-li vytvořit diferenciální pár, vyberte dva vodiče, které chcete spárovat a spusťte příkaz „Set diff. pair“. Můžete také vybrat jeden vodič a budete požádáni o druhý. Chcete-li oddělit vodiče, použijte příkaz „Clear diff. pair“. Doporučuje se přiřadit oběma vodičům stejnou hladinu (Netclass) např. „100diff“, abyste jim mohli snadno nastavit stejná pravidla.

18.5 Přímě na desku (To board)

Chcete-li usnadnit umístění pouzder součástek na desku plošných spojů DPS (PCB), jednoduše vyberte požadované symboly a stiskněte „To board“. Tím se přepnete na editor desky dále spusťte nástroj „place package“ s pouzdry pro vybrané symboly. Možná bude nutné nejdříve znovu načíst seznam spojů (netlist) v Editoru desky, aby editor desky načtl nové komponenty.

18.6 Snímek obrazovky



Editor desky plošných spojů DPS (PCB)

Chcete-li spustit editor desky plošných spojů (dále jen desky), klikněte na „Board“ ve správci projektu. Stejně jako všechny ostatní grafické editory je i editor desky v Horizon-EDA založen na Interaktivním manipulátoru

Editor desky si udržuje interní kopii seznamu spojů (netlist). Chcete-li aktualizovat seznam spojů, stiskněte „Save“ (Uložit) v editoru schémat pro zápis seznamu spojů na disk, pak klikněte v editoru desky na „reload netlist“ pro aktualizaci seznamu spojů nebo znovu otevřete editor desky.

19.1 Vyplněné oblasti (Planes)

Pro přidání vyplněných oblastí nejprve nakreslete do hladiny spojů (copper) mnohoúhelník (polygon) požadovaného tvaru. Poté pomocí nástroje „Add plane“ přiřadíte oblast. Pro zobrazení vyplněných oblastí je nutné aktualizovat tyto oblasti pomocí příkazu „Update all Planes“. Oblasti s nižší prioritou se vyplní jako první.

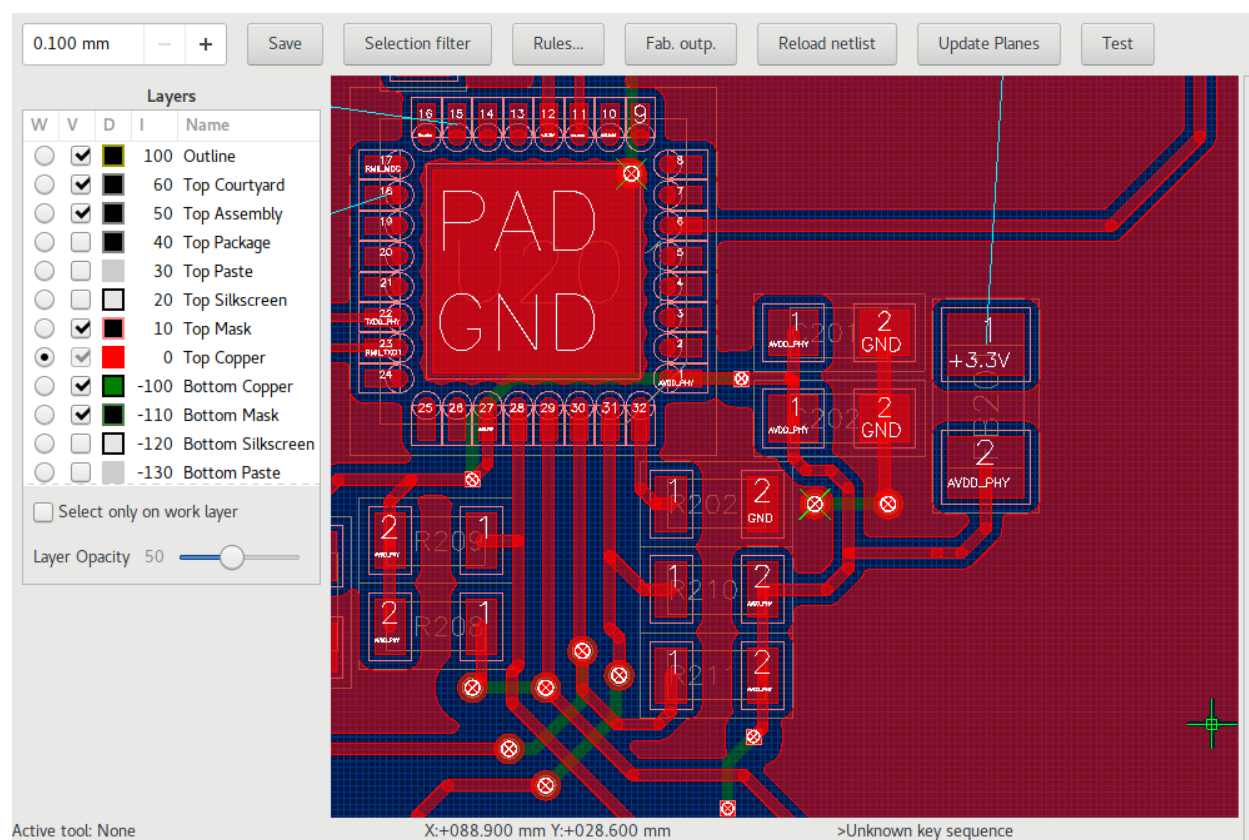
19.2 Prokovené otvory (Vias)

Před vložením prokoveného otvoru vytvořte výchozí pravidlo pro tyto otvory v okně pravidel „Rules“ a přiřadíte jim požadovaný tvar (padstack).

19.3 Diferenciální páry

Jedná se o speciální útvar trasy signálového páru se stanovenými parametry vzájemných elektrických vlastností. Chcete-li vytvořit diferenciální pár, podívejte se na *Diferenciální páry*. Před trasováním diferenciálního páru, vytvořte pravidlo určující šířku stopy a mezeru tohoto páru. Chcete-li vést trasu, použijte nástroj „Route diff“.

19.4 Snímek obrazovky



Kopírování rozmístění a umístění

20.1 Motivace

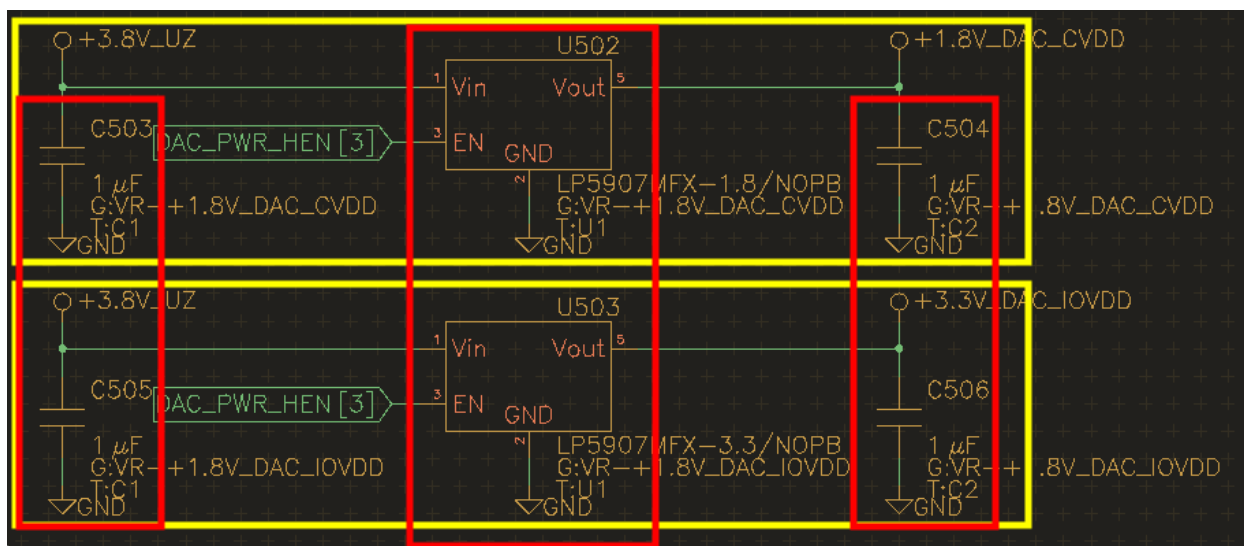
Návrh desky plošného spoje často obsahuje podobné, ale ne shodné celky jako jsou např. regulátory napětí. Nebylo by hezké, kdyby stačilo udělat rozmístění jednou a potom zkopírovat do ostatních kopií? Technologie Horzion-EDA vám to umožní v jednoduchém dvoustupňovém procesu.

20.2 Skupiny a značky

Aby tato funkce fungovala, musíte nejprve sdělit Horizonu-EDA, jak spolu komponenty souvisí. Toho je dosaženo přiřazením skupin a značek ke komponentám. Každá sekce, tj. Všem komponentám spojeným s jedním regulátorem napětí, přidělte jednu skupinu. Vyberte všechny symboly jedné sekce a pomocí příkazu „Set new group“ přiřadte všechny do nové skupiny. Pro zviditelnění skupin a značek na schematu použijte „Toggle group & tag visibility“. Skupina komponent a značka se poté zobrazí pod referenčním označením.

Abychom odlišili v Horizonu-EDA související komponenty v každé skupině, dostanou přiřazené shodné značky. Protože nově umístěná součást již bude mít shodnou značku a skupinu a značky zůstanou zachovány při kopírování / vkládání jiné kopie stejného okruhu budou pravděpodobně mít vhodné značky již nastaveny. Chcete-li změnit značku na součásti, použijte příkaz „Set tag“.

Až budete hotovi, schéma by mělo vypadat zhruba takto (obdélníky přidány pro názornost). Všechny komponenty uvnitř žlutého obdélníku patří do stejné skupiny, vše uvnitř červeného pole patří ke stejné značce.



Chcete-li se ujistit, že jste udělali přiřazení správně, můžete použít akci „Highlight group/tag“.

20.3 Deska

Umístěte a trasujte nějakou skupinu jako obvykle.

20.3.1 Kopírovat umístění

Pro každou skupinu umístěte pouzdro součástky tak, aby ostatní pouzdra součástek byly odkázány na požadované místo a umístěte okolo všechny ostatní pouzdra součástek. Poté vyberte všechny součástky skupiny, které chcete znovu použít a spusťte nástroj „Copy placement“. Klikněte na referenční pouzdro (kterékoli pájecí místo nebo středový bod) v již umístěné skupině a všechny vybrané součástky budou umístěny shodným způsobem.

20.3.2 Kopírování skladeb

Vyberte všechny spoje (ostatní objekty budou ignorovány), které chcete kopírovat v natrasované skupině, spusťte příkaz „Copy tracks“ a klikněte na libovolné pouzdro (libovolné pájecí místo nebo středový bod) v cílové skupině.

Zpětné úpravy (Backannotation)

Někdy můžete potřebovat propojit vývody součástek na základě jejich umístění na desce, jako například konektory nebo vývody FPGA. Nebylo by krásné, pokud by jste mohli definovat tato spojení přímo v editoru desky bez přechodu tam a zpět mezi editory desky a schémat pro každý spoj? S Horizonem-EDA je to možné!

21.1 Jak na to

Pomocí nástroje „Draw connection line“ (spustit přes mezerník) propojte jednotlivá pájecí místa (plošky), které požadujete propojit. Potom použijte funkci „Backannotate connection lines“ pro zpětnou změnu v editoru schémat. Nově vytvořená připojení se zobrazí jako prázdné štítky nového přerušného spoje. Po uložení schématu a aktualizaci seznamu spojů (Reload netlist) v editoru desky, budou spoje automaticky nahrazeny naznačenými spoji, které je nutno následně ručně propojit stopou mědi. Pozn. Pro správnou činnost této funkce je nutné mít spuštěn editor desky a editor schémat současně.

21.2 Omezení

Protože tato funkce není přidána příliš dlouho, některé možnosti nejsou zatím podporovány:

- Nelze propojit dva existující spoje
- Řetězové spoje (propojení dvou spojů s jedním vývodem) nepřinesou očekávaný výsledek

¹

¹ Přeložil / Translated by : Peta-T 16.11. 2019

Pravidla

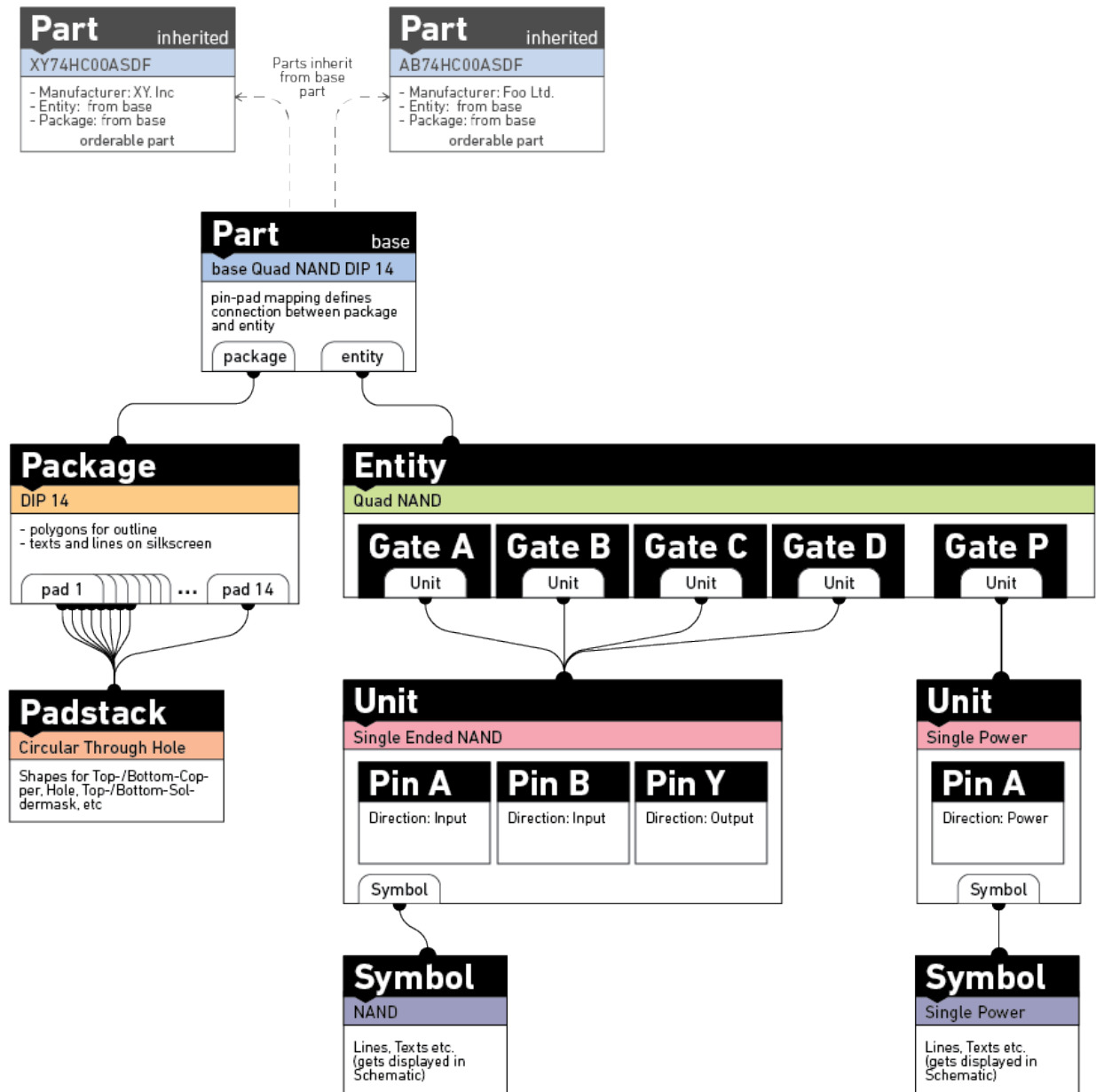
Horizon-EDA používá pravidla pro specifikaci vazeb pro DRC i pro vstup různých nástrojů, jako je interaktivní trasér. Pravidla jsou vyhodnocena shora dolů a použije se první pravidlo odpovídající všem kritériím. Je tedy na vás, abyste se ujistili, že pravidla jsou uspořádána od konkrétnějších na méně konkrétní. Vždy je dobré mít obecné pravidlo úplně dole.

Why a Pool?

So what's all this Pool stuff anyhow? Many EDA packages organize packages, symbols and the like in libraries. These are often messy and version-controlling these is difficult since many independent parts are put in a single file. Especially the latter often makes collaboration difficult.

With horizon, there are no libraries. Instead all the non-project elements (symbols, etc.) are stored in a pool. Similar to the „central library“ approach common among the more enterprisey EDA packages.

However the new thing here is, that a Part within this pool is *composed* of multiple other elements, that handle different aspects of the Parts nature:



For example you can define one „Quad NAND“ Entity and reuse it for each new NAND Part, without having to redefine the Gates time and time again. The Quad NAND Entity in turn is composed of multiple Single Opamp NAND Units and one Power Unit. If you now want to make a Dual NAND Entity, you can just reuse the already existing Units and this guarantees you consistency with the other NAND parts in your pool.

Each of these elements that make up a Part is stored in a single json file in the respective directory, i.e. /symbols, /entities, /units, /parts, etc. The exact location within these directories is irrelevant, as long the json file is stored in the correct directory: Symbols in /symbols, Units in /units and so on. Additionally it is important for the files to end in „.json“ so they can be picked up by the pool updater. To make searching for parts more convenient, the metadata of all json files is aggregated into a sqlite database. This is what the ‚Update pool‘ button in the Pool Manager is for.

Naturally a pool with a focus on composition is organized using tags instead of a hierarchical system since these often lead to (unnecessary) confusion over aspects like whether to group parts by manufacturer or other attributes.

Although you can create your own pool, you are strongly encouraged to use the pool over at <https://github.com/>

[horizon-eda/horizon-pool/](#). To add new parts to it, simply submit a merge request. See also: *Contribute to the Pool*

Fond knihovny součástek, pouzder, pájecích míst a schematických značek (Pool)

Co to vlastně je fond knihovny součástek (Pool)? Pro funkci programu je potřeba organizovat mnoho souborů pouzder součástek, symbolů a podobných v jakýchsi knihovnách. (Moje) zkušenost ukázala, že jsou často chaotické a správa verzí je obtížná, protože mnoho nezávislých částí definic je vloženo do jednoho souboru. Zejména toto ztěžuje spolupráci.

V Horizonu-EDA neexistují knihovny. Místo toho všechny neprojektové součásti (symboly atd.) jsou umístěny do jednoho fondu (repozitáře). Stejně jako přístup do „centrální knihovny“, která je běžnější spíše ve firemních programech pro tvorbu desek plošných spojů. I když si můžete vytvořit svůj vlastní fond, důrazně se doporučuje používat tento fond <https://github.com/carrotIndustries/horizon-pool/>. Chcete-li do něj přidat nové díly, jednoduše odešlete požadavek na přidání do veřejné knihovny pomocí příkazu „Merge“.

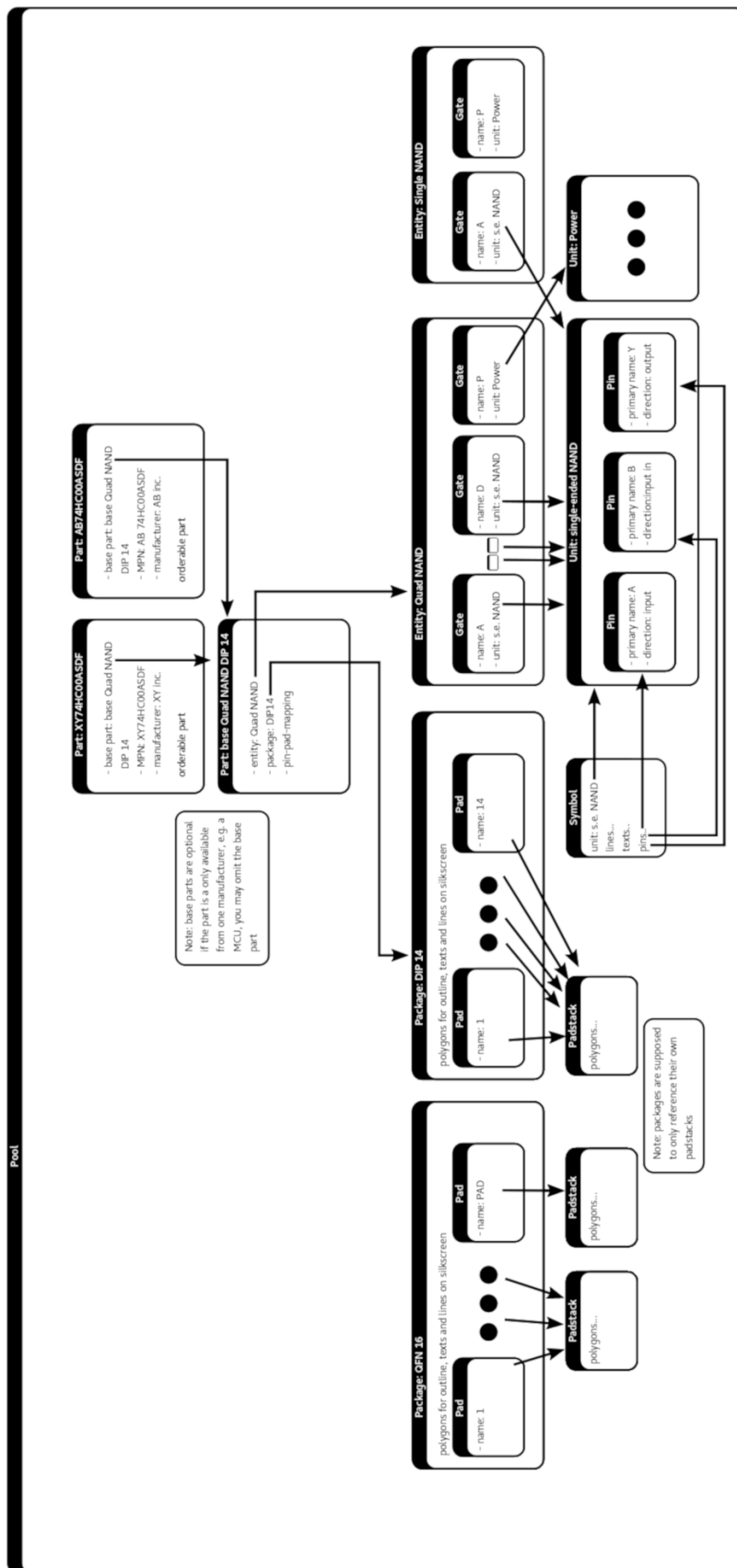
Nyní je fond organizován pomocí značek místo hierarchického systému, protože tyto často vedou k záměně nad aspekty, jako je tomu, zda skupiny dílů podle výrobce nebo jiné atributy.

Aby byl fond uspořádaný, přidávejte pouze díly, které můžete skutečně koupit s jejich odpovídajícími symboly, entity atd. Takže nepřidávejte součástku s názvem 7805, místo toho přidejte MC7805BDTRKG vyrobený společností ON Semiconductor.

Každá z níže uvedených položek je uložena v jednom souboru s příponou .json v příslušné hlavní složce, tj. /symbols, /parts, atd. Přesné umístění v této složce je nepodstatné, pokud je soubor json uložen ve správné hlavní složce. Je důležité, aby soubory měli příponu „.json“, aby byly nalezeny Správcem fondu. Chcete-li pohodlně vyhledávat díly je možné vložené vyhledávací údaje ze všech souborů json řadit pomocí databáze sqlite. To je umožněno tlačítkem ‚Update pool‘ ve Správci fondu.

24.1 Struktura fondu

Pro pochopení souvislostí jednotlivých souborů ve fondu je nutné pochopit jejich strukturu, která je na níže uvedeném obrázku, nebo v případě nečitelnosti je možno ji vidět [zde](#)



24.1.1 Součástky (Parts)

Na vrcholu struktury fondu je součástka (Part). Chcete-li se vyhnout zdvojení, součástka může zdědit svoji definici z jiné součástky. To je určeno pro použití ve skupině součástí, které se liší pouze v některých vlastnostech jako odpor nebo výstupní napětí pro pevné regulátory napětí. Každá součástka může být doplněna parametrickými daty, aby bylo vyhledávání snadnější. Tato funkce je zatím ve vývoji, protože není k dispozici žádné uživatelské rozhraní pro součástky založené na parametrických datech. Součástky také obsahují mapování Entity vývodů k pájecím obrazcům pouzder.

24.1.2 Pouzdra (Packages)

Pouzdro definuje průmět (otisk) půdorysného tvaru součásti na desku plošného spoje. Pokud výrobce součástky definuje doporučený tvar průmětu, použijte tento. Používejte pouze obecné pouzdra, pokud odpovídající neexistují. Podrobnosti o pouzdrech viz. *Vytvoření pouzdra součástky*.

24.1.3 Entity (Entities)

Entita je reprezentace součásti v seznamu spojů (netlist). Díly, které jsou logicky stejné jako např. různé typy konektorů USB proto mohou všechny sdílet stejnou entitu, např. „USB konektor se stíněním a ID“. Vlastní součástky se sestávají z jedné nebo více Jednotek (Units).

24.1.4 Jednotky (Units)

Jednotka ve skutečnosti definuje logické vývody součástky. Pouze pro součástky, které sestávají z jednoho celku „gate“ jako např. regulátory napětí, jejich entity jednoduše odkazují na jednu jednotku (Unit). Pro části sestávající z více celků „gate“ jako duální operační zesilovač nebo velký mikroprocesor, každý celek „gate“ odkazuje na jednu jednotku. Mít jednotky oddělené od entit umožňuje sdílení více entit stejnou jednotkou. Předpokládá se tedy, že entita jednoho logického celku „gate“ může odkazovat na stejnou jednotku např. čtyřikrát. Kromě jména má vývod i směr (pro ERC) a volitelně alternativní názvy vývodů (pinů) pro popis vývodů, které mají více funkcí, jak je běžné např. u Mikroprocesorů.

24.1.5 Symboly (Symbols)

Symbol se používá ve schématu k reprezentaci jednotky. Na rozdíl od jiných EDA aplikací, symbol právě zobrazuje vývody z jeho jednotky a nedefinuje je.

Správce fondu (Pool manager)

Správce fondu a Průvodce součástmi pomáhají se správou komponent jako jsou symboly, entity a součástky ve fondu. Pravděpodobně budete používat Správce fondu pro vytváření nových dílů. Chcete-li otevřít správce fondu, spusťte `horizon-eda` („`.exe`“) a vyhledejte soubor `pool.json` fondu, který chcete upravit. Podle toho, jakou komponentu chcete vytvořit, je k dispozici několik pracovních postupů:

25.1 Zdědění nové součásti z existující součásti

Když součástka, kterou se chystáte vytvořit, již existuje v jiné variantě (jiná hodnota, nebo jiný teplotní rozsah), ale jinak shodná, tak by nová součástka měla být zděděna ze stávající součástky. Chcete-li to provést, vyberte požadovanou základní součástku na záložce „Parts“ a klikněte na „Create Part from Part“. Po zadání umístění souboru nové součástky se zobrazí Editor součástek. Zrušte zaškrtnutí možnosti „zdědit“ (inherit) pro atributy, které chcete změnit a uložte novou součástku.

25.2 Vytvoření nové součásti ze stávající entity

Tento pracovní postup je vhodný, pokud již entita pro novou součást existuje. Odporů nebo LED v nestandardních pouzdrech jsou typickým příkladem. Pro vytvoření nového pouzdra viz [Vytvoření pouzdra součástky](#). Na záložce „Parts“ klikněte na „Create Part“ pro vytvoření nové součásti. Potom zadejte entitu, pouzdro a umístění souboru součástky, kterou můžete pozměnit a mapovat pájecí místa na vývody v Editoru součásti.

25.3 Vytvoření zcela nové součásti

Mnoho součástek, jako jsou mikroprocesory (MCU), FPGA, ADC a další zázraky dnešního světa vyžadují vytvoření nových jednotek a entit. To by bylo manuálně velmi zdlouhavé, v tomto vám může pomoci průvodce součástmi (Part Wizard). Poté, co vyberete pouzdro součástky (pro vytváření pouzder součástek viz [Vytvoření pouzdra součástky](#)) na záložce „Packages“ klikněte na „Part Wizard...“. Budete uvítáni seznamem všech pájecích míst pouzdra.

Vyplňte názvy vývodů podle katalogového listu součástky. Vložte pouze primární jméno vývodu (např. PB5) na MCU do sloupce zcela vlevo a vložte všechny ostatní názvy (jako UART0_TX, TA0) oddělené mezerou do sloupce „Alt. names“. Pokud je vaše součástka *opravdu* velká (jako FPGA nebo velký MCU), může potřebovat, aby se ve schématu zobrazil více než jeden symbol. Vyberte všechny vývody, které chcete mít se stejným symbolem a typem v alternativním názvu vývodu. V případě, že je více podložek elektricky identických (např. několik vývodů GND), můžete je seskupit jejich výběrem a kliknutím na tlačítko „Propojit podložky“ na dolním panelu nástrojů. Tímto způsobem bude pro vybrané položky vytvořen pouze jeden vývod.

U opravdu velkých součástek s více než 100 vývody může být ruční vkládání příliš zdlouhavé. Chcete-li se tomu vyhnout, je možné použít průvodce součástí (Part wizard) a importovat názvy vývodů ze souboru s příponou json. Tento soubor může být generován např. nějakým skriptem v jazyce Python nebo podobném. Struktura souboru json by měla vypadat takto:

```
{
  "1": {"pin": "PB0", "alt": ["TXD", "SDA"], "gate": "Main"},
  "2": {"pin": "PB1", "alt": ["RXD", "SCL"], "gate": "Main"}
}
```

Klíč určuje název vývodu součástky. Záznamy s pin-gate přiřazují názvy k číslu vývodu.

Vhodnými zdroji údajů se jmény vývodů a pájecích míst jsou:

- IBIS modely
- BSDL soubory
- katalogové listy PDF dané součástky

Není tak složité extrahovat údaje o vývodech a pájecích místech z jednoho konkrétního katalogového listu pomocí funkcí kopírovat/vložit text do programu LibreOffice Calc, vyčistit a přeuspořádat ho a následně exportovat jako soubor CSV. Tento soubor poté převést na výše uvedený soubor s příponou json.

Jakmile vyplníte názvy vývodů, klikněte vlevo nahoře na „Další“ pro postup na další obrazovku. Vyplňte položky podle vaší součástky. Pokud si nejste jisti, co tam má být, podívejte se na stávající součástky ve fondu. Pokud je vaše součástka k dispozici vícekrát v téměř identických variantách, které se liší pouze v aspektech jako teplotní rozsah nebo možnost balení (Páska / Role, Trubka, atd.) vytvoří součástku, kterou se chystáte použít. Pro vytvoření dalších variant postupujte podle pokynů v horní části. Postarejte se o správné zadání umístění jednotek / symbolů / entit a částí tak, aby končily v podsložkách jejich příslušné složky ve fondu.

Pro každý celek (gate) klikněte na „Edit Symbol“ pro spuštění interaktivního manipulátoru vytvoříte symbol pro tuto jednotku (unit). Použijte příkaz „Map pin“ pro umístění vývodů do symbolu a „Draw line rectangle“/„Edit line rectangle“ pro nakreslení těla symbolu. Nezapomeňte dát symbolu smysluplné jméno a umístěte texty „\$REFDES“ a „\$VALUE“.

Když nakreslíte všechny symboly a vyplníte všechny údaje, kliknutím na „Finish“ konečně vložíte součást do fondu.

25.4 Kam ukládat komponenty

Při vytváření nových symbolů, součástek a podobně správce fondu / průvodce tvorbou součástek vás dříve nebo později požádá o název souboru nebo složky (v případě pouzder) k uložení nové součástky. Technicky, vámi specifikovaná cesta pouze musí splňovat dva požadavky:

- **Musí být ve správné složce nejvyšší úrovně, tj. pro každou schematickou značku** musí být někde ve složce /symbols atd. Pájecí obrazec specifický pro a pouzdro musí být umístěno do složky /padstack.
- Název souboru musí mít příponu .json

Pro získání představy jak to všechno prakticky vypadá se podívejte do veřejného fondu na adrese <<https://github.com/carrotIndustries/horizon-pool/>>‘__

25.5 Databáze fondu

Fond uchovává údaje (názvy souborů, UUID, jména atd.) V SQLite databázi usnadňující vyhledávání. Normálně si správce fondu aktualizuje databázi pokaždé, když se něco změní. Nicméně, pokud externě manipulujete / odstraňujete soubory, musíte kliknout na „Aktualizovat“ (Update Pool) databázi, která zahrne provedené změny.

25.6 Integrace GitHub

Aby integrace GitHub fungovala, musí být fond stažen pomocí tlačítka „Download...“ na úvodní stránce správce fondu. Správce fondu naklonuje veřejný fond do složky `.remote` v systému vašeho místního fondu. Pokud vše půjde dobře, neměli byste nikdy zasahovat do této složky. K dispozici jsou dvě operace pro uchování vaší místní kopie aktuální a zahrnutí vašich součástí do veřejného fondu

25.6.1 Aktualizace fondu

Tento příkaz aktualizuje vaše kopie globálního fondu ve složce `.remote` od posledního potvrzení a zeptá se vás, jaké změny chcete použít ve vašem místním fondu.

25.6.2 Vytvoření žádosti o zahrnutí součástí do veřejného fondu

Nejprve přidejte součástky / entity / atd. do seznamu „items to be merged“, poté vyplňte název a tělo žádosti o zahrnutí. Správce fondu bude automaticky přidávat položky, které jsou nutné, aby nebyly porušeny závislosti. Takže když vytvoříte zcela novou částku s novou jednotkou, entitou a pouzdrům přidejte ji do seznamu pro přidání součástí. Nezapomeňte přidat nové symboly. Poté, co se ujistíte, že to je to, co chcete, klikněte na „Create pull request“. Budete vyzváni k zadání přihlašovacích údajů pro server GitHub.

Vytvoření pouzdra součástky

Rozložení desky plošných spojů může být tak dobré jak dobře jsou vytvořeny pájecí obrazce (footprints) pro jednotlivá pouzdra součástek, které program používá, proto je důležité vytvořit vysoce kvalitní podklady obrazců těchto obrazců (footprints).

V Horizonu-EDA se pouzdro součástky se skládá z těchto částí:

- Pájecí místa (pads), na které se součást připájí
 - Spoje - měděné vrstvy tras jednotlivých propojení v jednotlivých (horní / dolní / vnitřní) vrstvách
 - Otvory (pro díly TH)
 - Vynechání nepájivé masky
 - Maska pájecí pasty
- Obrys pouzdra součástky (Package outline)
- Montážní obrys a referenční označení (Assembly outline)
- Textový popis součástek (Silkscreen)
- Obrys zástavby součástky (Courtyard outline)

26.1 Pájecí obrazce (pads)

Každý pájecí obrazec je definován několika pájecími místy (padstack) uspořádanými do různých tvarů a parametry aplikovanými na tento tvar. Podrobnosti o pájecích místech viz. Pájecí místa / plošky (Padstacks). Tyto pájecí obrazce jsou pravděpodobně nejdůležitější vlastností pouzdra součástky, je vhodné s nimi začít. Pájecí místa můžete umístit ručně pomocí příkazu „Place pad“ nebo je nechejte umístit automaticky podle běžně používaných vzorů pomocí příkazu „Footprint gen.“, který je k dispozici na horním panelu. Po umístění pájecích míst mají stále svou výchozí velikost, která patrně není ta, co potřebujete. Chcete-li velikost změnit, vyberte parametry, které chcete upravit a pomocí nástroje „Edit pad“ přidejte parametry do pájecího místa. V závislosti na vybraném tvaru pájecího místa jsou přiřazeny určité parametry. Nejčastěji se používají šířka a výška pájecího místa. Použijte tlačítko zaškrtnutí vedle parametru pro parametry, které se mají použít na všechny vybrané podložky.

26.2 Obrys pouzdra součástky

Obrys pouzdra se používá k vizualizaci obrysu součásti. Dá se říct, že by tedy měla sledovat jmenovité rozměry součásti. Můžete použít nástroj „import DXF“ pro import výkresu DXF získaného ze STEP modelu nebo jinak. Protože účel obrysu součásti je čistě vizuální, můžete použít buď čáry, nebo mnohoúhelníky. Vývody přidejte, pouze pokud výrazně přispívají ke vzhledu součásti.

26.3 Montážní obrys

Montážní obrys končí na výkresu sestavy (zatím není implementováno) a jeho účelem je pomoc při montáži a kontrole PCA. Montážní obrysová vrstva tedy obsahuje pouze tyto položky: Mnohoúhelník označující obrys součásti, volitelně vývody, pokud se výrazně přispívají ke vzhledu součásti a referenční označení součásti. Na rozdíl od obrysu pouzdra je montáž pouze hrubá náhrada tvaru součásti. Musí zahrnovat nějaký druh vizuálního označení umístění prvního vývodu součásti. Použijte příkazy „Draw polygon rectangle“ a jeho možnosti dekorace pro kreslení takového obrysu. Pro referenční označení, vložte text obsahující „\$ RD“ v takové velikosti, aby se vešel do obrysu sestavy, i když je předpona o 4 číslice delší.

26.4 Textový popis součástek

Účelem textového popisu je objasnit umístění dílů a orientace při ruční montáži a vizuální kontrole, také by měla pomoci pokud je součást citlivá na orientaci, použijte nějakou značku prvního vývodu. Nepoužívejte tečku pro označení prvního vývodu, místo toho sklopte nebo prodlužte grafickou značku. Doporučená tloušťka čáry je 0,15 mm. Také vložte text „\$ RD“, s tloušťkou čáry 0,15 mm do hladiny popisu.

26.5 Obrys zástavby součástky

Obrys zástavby označuje prostor potřebný pro plochu kolem součástky, která nesmí být obsazena jinými součástkami, aby byl ponechán dostatečný prostor pro montáž. Vzhledem k tomu, že velikost obrysu zástavby je třeba upravit v závislosti na uživatelských výrobních požadavcích, musí být nastaven pomocí parametru programu. Při zvětšení obrysu zástavby 0 mm je obrys zástavby (obdélníkový) kolem pájecích míst a obrysu pouzdra součástky. Chcete-li vytvořit obdélníkový obrys zástavby, který lze parametrizovat, proveďte toto:

Pomocí příkazu „Generate courtyard“ vygenerujete počáteční obrys zástavby na 0 mm rozšíření. Pokud to nemá za následek požadovaný mnohoúhelník, použijte příkaz „Draw polygon rectangle“ pro nakreslení počátečního obrysu a nastavení jeho skupiny parametrů na „courtyard“ pomocí editoru vlastností na pravé straně okna.

Otevřete okno „Parameters“ a klikněte na „Insert courtyard program“. Je-li vše jde správně, měl by se přidat program obrysu zástavby a jeho parametr „Courtyard expansion“ nastavený na 0,25 mm.

Contribute to the Pool

The official pool at <https://github.com/horizon-eda/horizon-pool/> lives from it's user contributions. There are multiple ways you can help. The most obvious one is by submitting parts you made.

To keep the pool nice and clean, only add parts you can actually buy with their corresponding symbols, entities, etc. So don't add some part called 7805, instead add a MC7805BDTRKG manufactured by ON Semiconductor. Once you created something you'd like to share, you can use the *Pool Manager* to upload your creation to the official pool:

27.1 Using the GitHub integration.

For the GitHub integration to work, the pool has to be downloaded using the „Download...“ button on the start Page of the pool manager. The pool manager will clone the global pool into the `.remote` directory in your local pool. If all goes right, you should never need to touch that directory. Two operations are available for keeping your local copy up-to-date and merging your parts into the global pool

27.1.1 Upgrade pool

This will update your copy of the global pool in the `.remote` directory to the latest commit and ask you which changes you'd like to have applied to your local pool.

27.1.2 Create pull request

First, add the Parts/Entities/etc. to the „items to be merged“ list, then fill in Pull Request title and body. The pool manager will automatically add items that are needed to not break references. So if you create an all-new Part with new Unit, Entity and Package, these will get added to the list when you add the Part. Don't forget to add the new symbols. After making sure that this is what you want, click the „Create pull request“ button. You'll be prompted for your GitHub credentials as well as your name and email address for the commit author information.

27.1.3 Helping by reviewing

Adding parts is a great thing, but checking parts other people made could be a good thing as well. More eyes that crosscheck a part against its datasheet will decrease the chance of something that works. If you successfully produced a PCB with certain parts on it, you can say something about solderability as well and this is a stronger indicator, that the part has no critical mistakes.

Sestavení (kompilace) programu na operačním systému Windows

28.1 Nainstalujte program MSYS2

Stáhněte si a spusťte instalátor programu msys2 z <http://msys2.github.io/> Mám testováno pouze s 64bitovou verzí, 32bit by měla fungovat také. Ujistěte se, že cesta, kterou jste vybrali pro instalaci neobsahuje žádné mezery (v názvech složek).

28.2 Spusťte konzolu MSYS

Spusťte položku nabídky Start „MSYS2 mingw 64 bit“, která by se měla zobrazit v okně konzoly. Všechny níže uvedené kroky se týkají toho, co by jste měli napsat do toho okna.

28.3 Nainstalujte aktualizace

Napište

```
pacman -Syu
```

pokud vám řekne, že chce restartovat před restartováním, zavřete okno konzoly a po restartu znovu spusťte program `pacman -Syu`.

28.4 Nainstalujte související součásti

Napište / vložte

```
pacman -S mingw-w64-x86_64-gtkmm3 git base-devel \
mingw-w64-x86_64-yaml-cpp mingw-w64-x86_64-boost \
mingw-w64-x86_64-sqlite3 mingw-w64-x86_64-toolchain \
mingw-w64-x86_64-zeromq mingw-w64-x86_64-glm zip \
mingw-w64-x86_64-libgit2 mingw-w64-x86_64-occe \
mingw-w64-x86_64-podofo --needed
```

Až budete vyzváni, stačí stisknout klávesu Enter. Posad'te se a počkejte, až instalátor dokončí instalaci téměř kompletního linuxového vývojového prostředí.

Než budete pokračovat, můžete přejít do jiné složky. Je to jednoduché zadejte “ cd“, mezeru a přetáhněte složku, do které chcete přejít do okna konzoly.

28.5 Klonovat zdrojový kód programu Horizon-EDA

```
git clone http://github.com/carrotIndustries/horizon
cd horizon
```

28.6 Sestavení (kompilace) programu ze zdrojového kódu

```
make -j 4
```

Číslo 4 na konci můžete upravit podle počtu procesorů ve vašem systému pro rychlejší kompilaci. Očekávejte 100% zatížení procesoru (CPU) po dobu několika minut dle výkonu počítače. Z důvodu zapnuté volby vkládání ladících symbolů mají výsledné spustitelné soubory značnou velikost.

28.7 Spuštění programu

Po kompilaci nebudete moci dvakrát kliknout na výsledné spustitelné soubory protože požadované knihovny DLL nejsou ve složce známé systému Windows. Budete muset spustit z příkazového řádku prostředí Mingw například pomocí příkazu `./horizon-eda`. Aby fungovalo stahování fondu, musíte zkopírovat soubor `/mingw64/ssl/certs/ca-bundle.crt` do složky obsahující `horizon-eda.exe`.

28.8 Vytvoření archivu

Chcete-li vytvořit archiv ZIP, jak je dostupný ke stažení, spusťte příkaz `./make_bindist.sh`.

Sestavení (kompilace) programu na operačním systému Linux

Sestavení programu Horizon-EDA v Linuxu je podobně jednoduché jako ve Windows

29.1 Nainstalujte závislosti

Ujistěte se, že máte nainstalované tyto související knihovny:

- Gtkmm3 >= 3.20
- cairomm-pdf
- librsvg
- util-linux
- yaml-cpp
- sqlite
- boost
- zeromq
- glm
- libgit2
- curl
- opencascade / opencascade community edition
- zeromq with C++ bindings: <https://github.com/zeromq/cppzmq>
- podofo
- libzip

pro různé distribuce jsou zde předpřipraveny příkazy pro případnou instalaci knihoven.

Pro verzi Ubuntu >= 17.04:

```
sudo apt install libyaml-cpp-dev libsqlite3-dev util-linux librsvg2-dev \  
libcairomm-1.0-dev libepoxy-dev libgtkmm-3.0-dev uuid-dev libboost-dev \  
libzmq5 libzmq3-dev libglm-dev libgit2-dev libcurl4-gnutls-dev liboce-oaf-dev \  
libpodofo-dev
```

V systému Arch Linux:

```
sudo pacman -S yaml-cpp zeromq gtkmm3 cairomm librsvg sqlite3 libgit2 curl \  
opencascade boost glm podofo libzip
```

Na Fedoře 25/26/27:

```
sudo dnf install git make gcc gcc-c++ pkg-config cppzmq-devel OCE-devel\  
gtkmm3-devel libgit2-devel libuuid-devel yaml-cpp-devel sqlite-devel librsvg2-  
devel\  
cairomm-devel glm-devel boost-devel libcurl-devel podofo-devel
```

Na openSUSE Tumbleweed:

```
sudo zypper in git make gcc gcc-c++ pkg-config cppzmq-devel oce-devel\  
gtkmm3-devel libgit2-devel libuuid-devel yaml-cpp-devel sqlite3-devel librsvg-  
devel\  
cairomm-devel glm-devel boost-devel libcurl-devel libpodofo-devel binutils-gold
```

Na FreeBSD 12:

```
sudo pkg install git gmake pkgconf e2fsprogs-libuuid sqlite3 yaml-cpp \  
gtkmm30 cppzmq libgit2 boost-libs glm opencascade podofo libzip
```

29.2 Klonovat zdrojový kód programu Horizon-EDA

```
git clone http://github.com/carrotIndustries/horizon  
cd horizon
```

29.3 Sestavení (kompilace) programu ze zdrojového kódu

```
make -j 4
```

Číslo 4 na konci můžete upravit podle počtu procesorů ve vašem systému pro rychlejší kompilaci. Očekávejte 100% zatížení procesoru (CPU) po dobu několika minut dle výkonu počítače. Z důvodu zapnuté volby vkládání ladících symbolů mají výsledné spustitelné soubory značnou velikost.

29.4 Spuštění programu

Výsledné binární soubory jsou samostatné a nevyžadují žádné externí datové soubory jako ikony nebo podobně. horizon-eda je hlavní spustitelný program. Spouštějte jej ze složky kde byl sestaven pomocí příkazu:

```
./horizon-eda
```

Parametry programu

Jak již bylo popsáno jinde v dokumentu, Horizon-EDA podporuje parametrizovatelné tvary pájecích míst a (v omezené míře) pouzder součástek. Chcete-li použít dané parametry na existující geometrii, každé pájecí místo a podobné objekty je možné změnit pomocí krátkého makra.

Tyto makra jsou psány ve vlastním zásobníkovém jazyce. Uživatelé kalkulaček HP se měli cítit jako doma. Protože v programu není možné provádět žádné smyčky, budou tyto programy ukončeny v daném čase. Zásobník obsahuje 64bitová celá čísla se znaménkem. Koncepčně roste shora dolů.

30.1 Syntaxe

Na nejvyšší úrovni je program tvořen značkami. Značky jsou oddělené libovolným množstvím mezer.

Typy značek:

- Celá čísla: číslo, volitelně opatřené znaménkem
- Rozměr: číslo s volitelnou zlomkovou částí, s příponou „mm“ plovoucí řádová značka před značkou mm se vynásobí 1×10^6 , od vnitřní jednotka měření programu Horizon-EDA je 1nm
- Matematické operátory, jako například: + - * /
- Řetězce znaků
- Parametry začínají znakem [a končí] jakákoliv značka mezi těmito dvěma znaky bude přidána jako parametr předchozího příkazu

30.2 Obecné příkazy

30.2.1 Nulový operand

`get-Parameter [<parameter>]` načte paramter a vloží jej na zásobník

30.2.2 Jeden operand

Před operací vypadá zásobník takto:

```
.   .  
.   .  
+---+  
| a |  
+---+
```

Operátory: | Hodnoty v zásobníku
dup | a a
chs | -a

30.2.3 Dva operandy

Před operací vypadá zásobník takto:

```
.   .  
.   .  
+---+  
| a |  
+---+  
| b |  
+---+
```

Operátory: | Hodnoty v zásobníku
+ | a+b
- | a-b
* | a*b
/ | a/b
dupc | a b a b (Duplikování souřadnic)

30.2.4 Tři operandy

Před operací vypadá zásobník takto:

```
.   .  
.   .  
+---+  
| a |  
+---+  
| b |  
+---+  
| c |  
+---+
```

Operátory: | Hodnoty v zásobníku
+xy | a+c b+c
-xy | a-c b-c

30.3 Příkazy pro pájecí místa

Aby program mohl manipulovat s objektem (tvaru atd.), musí být přiřazena třída parametrů. `## set-shape set-shape [<třída parametrů> <form>]` Nastaví tvar na zadaný formulář nebo jej přesune na určenou pozici Platného formuláře:

- `obdélník`, zobrazí výšku, šířku
- `kruh`, zobrazí průměr
- `obround`, zobrazí výšku, šířku
- `pozice`, zobrazí polohu y, x

30.3.1 Zadání otvoru

`set-hole [<třída parametrů> <shape>]` Nastaví díru na specifikovaný tvar z přednastavených možností:

- `kulatý`, zobrazí průměr
- `slot`, zobrazí délku, průměr

30.4 Polygonové příkazy (pájecí místa a pouzdra součástek)

30.4.1 Zadání polygonu

`set-polygon [<třída parametrů> <shape> <x0> <y0>]` Nastaví polygon před připraveného tvaru se středem na (x0, y0) z vybraných možností:

- `obdélník`, zobrazí výšku, šířku
- `kruh`, zobrazí průměr

30.4.2 Zadání vrcholů polygonu

`“ set-polygon-vertices [<třída parametrů> <n_vertices>] “` Načte `n_vertices` souřadnic vrcholů ze zásobníku a vytvoří z nich mnohoúhelník.

30.4.3 Vytvoření polygonu

`expand-polygon [<třída parametrů> <x0> <y0> <x1> <y1> ... <xn> <yn>]` Vytvoří polygon určený hodnotami souřadnic v parametrech načtených ze zásobníku.

30.5 Příklad programu (pro SMD obdélníkové pájecí místo)

```
get-parameter [ pad_width ]
get-parameter [ pad_height ]
dupc dupc
set-shape [ pad rectangle ]
get-parameter [ solder_mask_expansion ]
```

(continues on next page)

(pokračujte na předchozí stránce)

```
2 *  
+xy  
set-shape [ mask rectangle ]  
  
get-parameter [ paste_mask_contraction ]  
2 *  
-xy  
set-shape [ paste rectangle ]
```


31.1 Interaktivní manipulátor

Primárním rozhraním Horizontu je tzv. „Interaktivní manipulátor“. Je to jednotný editor symbolů, schémat, pájecích míst (padstacks), pouzder součástek (packages) a desky plošných spojů (PCB).

31.1.1 Pracovní plocha (Canvas)

Na pracovní ploše se vykreslují objekty, jako jsou symboly schématu, pouzdra součástek nebo jednotlivé spoje. Výstupem vykreslení jsou segmenty čar a trojúhelníky, které jsou následně vykresleny pomocí grafického procesoru (GPU). Chcete-li vykreslit na pracovní ploše jiné než grafické (OpenGL) objekty, pracovní plocha poskytuje nástroje pro získání více informací o tom, co je vykresleno. Zatím program umožňuje výstup ve formátu Gerber, 3D náhled na desku a kontrolu dle pravidel návrhu (DRC - Design rule checking)

31.1.2 Jádro (Core)

Protože některé dokumenty, jako jsou symboly a schémata, obsahují ty samé typy objektu (např. texty), schémata a seznam spojů (netlist) musí být modifikované synchronizovaně, musí dojít k zapouzdření. Jádro je spojovací článek mezi dokumentem, pracovní plochou a nástroji.

31.1.3 Nástroje (Tools)

Pro každou akci, kterou může uživatel udělat, existuje nástroj. Vyvolaný nástroj začne přijímat vstupy z klávesnice a myši a podle funkce upravuje dokument pomocí jádra. V případě potřeby může nástroj vyvolat další dialogy pro vyžádání dodatečných informací od uživatele.

31.1.4 Editor vlastností (Property editor)

Jednoduché úpravy parametrů objektů, jako je šířka čáry, neumožňují přímo jejich nástroje, proto jádro poskytuje rozhraní úprav vlastností. ovládací prvky editoru vlastností jsou automaticky generovány z popisu objektů.

Použití příkazového řádku (CLI Command Line Interface)

Správce projektu a správce fondu do značné míry eliminovali potřebu spouštět interaktivní manipulátor a další nástroje přímo z příkazového řádku, ale je to stále užitečné pro vývoj.

Všechny níže uvedené příkazy vyžadují proměnnou prostředí `HORIZON_POOL`, nastavenou na složku fondu (obsahující soubor `pool.json` a `pool.db`)

32.1 Použití horizon-imp

Režim symbolů:

```
horizont-imp -y <soubor symbolu>
```

Schematický režim:

```
horizon-imp -c <soubor schematu> <soubor bloku>
```

Režim padstack:

```
horizon-imp -a <soubor pájecího obrazce>
```

Režim balíčku:

```
horizon-imp -k <soubor pouzdra součástky>
```

Režim desky:

```
horizon-imp -b <soubor desky> <soubor bloku> <přes složku>
```

32.2 Použití horizont-pool

Většina příkazů `-edit` a `-create` spustí příslušný `$EDITOR`, který vytvoří odpovídající soubor ve formátu YAML.

```
horizon-pool create-unit <soubor jednotky>
horizon-pool edit-unit <soubor jednotky>
horizont-pool create-symbol <soubor symbolu> <soubor jednotky>

horizon-pool create-entity <soubor entity> [<soubor jednotky> ...]
horizon-pool edit-entity <soubor entity>

horizon-pool create-package <soubor pouzdra>
horizont-pool create-padstack <soubor pájecího obrazce>

horizont-pool update #Aktualizuje SQLite databázi fondu knihovny.
```

Po vytvoření souborů nezapomeňte spustit `horizon-pool update`

32.3 Použití horizont-prj

Pomocí těchto příkazů můžete vytvářet prázdné bloky, schémata atd.

```
horizon-prj create-block <souboru bloku>

horizon-prj create-schematic <soubor schematu> <soubor bloku>

horizon-prj create-board <soubor schematu> <soubor bloku>
```

Parts of Horzion EDA are available as a python module for use in scripts.

33.1 Installation

The python module isn't included in the `all` target. To build it, run `make horizon.so`. This requires the python 3 headers to be installed. You can then place it in python's `sys.path` and import it using `import horizon`.

33.2 Usage

```
import horizon

#open project
p=horizon.Project("/path/to/project.hprj")

#open schematic
sch = p.open_top_schematic()

#export PDF
pdf_settings = sch.get_pdf_export_settings()
pdf_settings['output_filename'] = '/tmp/sch.pdf'
sch.export_pdf(pdf_settings)

#export BOM
bom_settings = sch.get_bom_export_settings()
bom_settings['output_filename'] = '/tmp/bom.csv'
sch.export_bom(bom_settings)

#open board
brd = p.open_board()
```

(continues on next page)

(pokračujte na předchozí stránce)

```
#export gerber
gerber_settings = brd.get_gerber_export_settings()
gerber_settings["output_directory"] = "/tmp/gerber"
brd.export_gerber(gerber_settings)
```

To further adjust the export settings, have a look at the dicts returned by the `get_*_export_settings` methods.