Symbolic Pads Close the Gap to Whole Symbolic Designs

D. Archambaud, A. Greiner, F. Pétrot, V. Reibaldi, F. Wajsbürt
Laboratoire MASI, Université Pierre et Marie Curie
4, place Jussieu, 75252 Paris cedex 05, France

Abstract

In this report, we present a layout library of symbolic pads. First we examine why such a library is required. Then we detail the pad functionalities and clock use specificities. Since these pads are symbolic, they are meant to be usable in many ways, and thus are simple. The electrical and architectural design principles are justified. We finally present the effects and issues involved in the portability of such a library, since the bounding pad itself is still technology dependent. This library has been validated with a test chip on a technology we were not familiar with and has shown to be first silicon right. This library is available within the public domain **Alliance** *CAD system.*

1 Introduction

Symbolic layout design methods are now usual for creating part of a circuit's masks. However, most of these methods are meant to be usable only in a standard cells design context, or for the design of optimized circuits, ie not on a whole chip. Among other reasons, the design of the pads using a symbolic approach has been felt as difficult.

As far as we know, only one concluding experiment has been conducted by Bull in designing a symbolic pad library, and still, some logic parts of the pads are substituted by technology dependent devices at the end of the symbolic to real translation process.

Based upon this experience, we decided to design a set of symbolic pads compatible with the symbolic layout approach used within the **Alliance** CAD system.

2 Why symbolic pads?

2.1 Why were pads only available in real view?

So far, the **Alliance** pad libraries where composed of almost *empty cells*. The symbolic cells were used when building the pad ring, providing the size of the pads and the position of the connectors to perform the pad to core routing correctly. The symbolic pads were discarded and replaced at the very end of the symbolic to real translation process.

Since the pads are different from a foundry to another, and from a process to another, several dedicated pad libraries were provided (pad10, pad12, pad15). Each of these libraries had to contain a version of the symbolic pads, since the connector locations and cells size were different — once divided by the λ — for each target process.

The method using such libraries was as follows:

- Design the circuit core in symbolic layout. Create a symbolic pad ring using the relevant symbolic pad library.
- Route between the symbolic core and the symbolic pads, using either the `ring` or `cheops` tools.
- Transform the symbolic figure into a real one, targeting a given process using `s2r`.
- Replace the pads by their real design provided by the relevant foundry. Actually this is done by `s2r`, using a correct catalogue file.

This method does not give enough flexibility, because the last routing phase depends on the target technology. So for each new target process several steps were necessary to create a correct pad library: for each pad find the closest λ coordinates for each of its connectors and for its size. Then modify the real layout so that its connectors match perfectly the symbolic ones. This is a painful job that requires the ability to edit **cif** or **gdsII** files, and can be avoided if the pads are also symbolic.

2.2 Why is it worth having symbolic pads?

Empty symbolic pads used so far only gave minimum geometric information :*"what is the size of the pad"* and *"where are the connectors"*.

Information such as fanin/fanout of the connectors and transistors used in the cell were not available when doing a transistor level layout extraction. It was necessary to describe the netlist of transistor of each pad for each process.

Designing symbolic pads extends the control of Alliance tools over the cells: Caracterisation is provided in the VHDL description; transistors can be extracted and cells can be visualized using Graal, such as any other symbolic cell of any other **Alliance** library.

Moreover, having symbolic cells makes the pads independent from the process technology. So far, such a portability of the design was limited to the core. The pad ring had to be modified when changing the targeted process. Using symbolic pads, the pad ring does not need to be modified provided the distance between the passivation openings fulfill the rules of the bounding machine.

Having editable symbolic pads has an other interest: one can easily modify the cells in accordance to its requirements. For instance, a clock wire has been added inside the cells, in such a way that the pad ring includes a clock ring that efficiently distributes the clock signal. A symbolic pad library including boundary scan functionnalities is also under study.

3 Design and Functionnalities

3.1 Pads global description

All the pads have the same size: 172 lambdas x 500 lambdas. They all have 5 *horizontal* wires going thru the cell and intended to construct the power supply and clock rings by mere abutment. Figure 1 shows the general pad skeleton. Two pairs of power supply wires are used: an external pair and an internal pair. The internal pair supplies the core and the logical part of the pads. The external pair is used for the final stage transistors in the pads - i.e. the transistors used as a diode in input pads and as final driver in output pads. The internal pair is not electrically connected to the external pair. Because of the large amount of current necessary when a pad drives an heavy capacitive load — there is an LC effect implied by the bounding wire inductance —, the noise that is generated on the power lines could have an impact on the logic levels in the circuit. So the pads external drivers power supplies are independent from the internal logic ones.

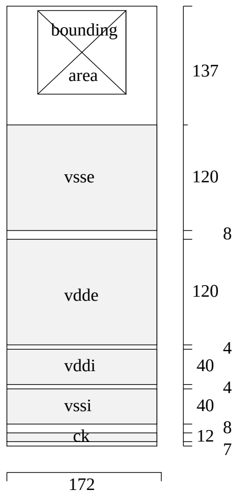

Figure 1: The basic pad description, numbers represent the object sizes, in λ

3.2 Pads functionnalities

The list of the pads in the library with their functionnality is given below:

`pi_sp`	input pad
`po_sp`	ouput pad
`pow_sp`	weak output pad
`pot_sp`	tristate output pad (may emit or not)
`potw_sp`	weak tristate output pad
`piot_sp`	input output tristate pads
`piotw_sp`	weak input output tristate pads
`pck_sp`	special clock input pad
`pvdde_sp`	external vdd power supply
`pvsse_sp`	external vss power supply
`pvddi_sp`	internal vdd power supply
`pvssi_sp`	internal vss power supply
`pvddeck_sp`	pvdde_sp plus ck buffer
`pvsseck_sp`	pvsse_sp plus ck buffer
`pvddick_sp`	pvddi_sp plus ck buffer
`pvssick_sp`	pvssi_sp plus ck buffer

Figure 2 shows the schematic of the tristate input output pad — `piot_sp` —, the more complex pad of the library.

The big transistors driving the **PAD** connector behave as an inverter when **EN** is set to one, since their gates are connected together. When **EN** is set to 0, the P transistor's gate is connected to **vddi** and the N transistor's gate is connected to **vssi**, then both act as diodes.

A pull-up transistor is connected to the **PAD** equipotential. It fixes the potential to **vdde** when **PAD** is not driven. This avoids "floating" signals that could cause high power consumption in the inverters driving **T** signal.

The schematics of other pads are derived from this one : **piotw_sp** is equivalent with approximately 500/1000 final stage buffer instead of approximately 1000/2000, **pot_sp** is a **piot_sp** with no input buffer.

Tristate busses in a circuit usually utilize a number of tristate pads with the same logical emission command. It is the reason why the **EN** and **/EN** inverters have particularly low sizes to reduce the pad command fanin. This is a way to drive a reasonnable number of pads with only one equipotential issued from logic glue in the circuit.

The output pad is a mere buffer string : two 30/60 inverters, then a 120/240 inverter, and finally a 840/1920 stage always configured as an inverter.

The input pad uses the final transistors (140/320) as protecting diodes. In addition the signal is bufferized twice through a 27/81 inverter and a 54/114 inverter. The first inverter has asymetric N/P sizes - it has been done on purpose to correctly trig some TTL board signals.

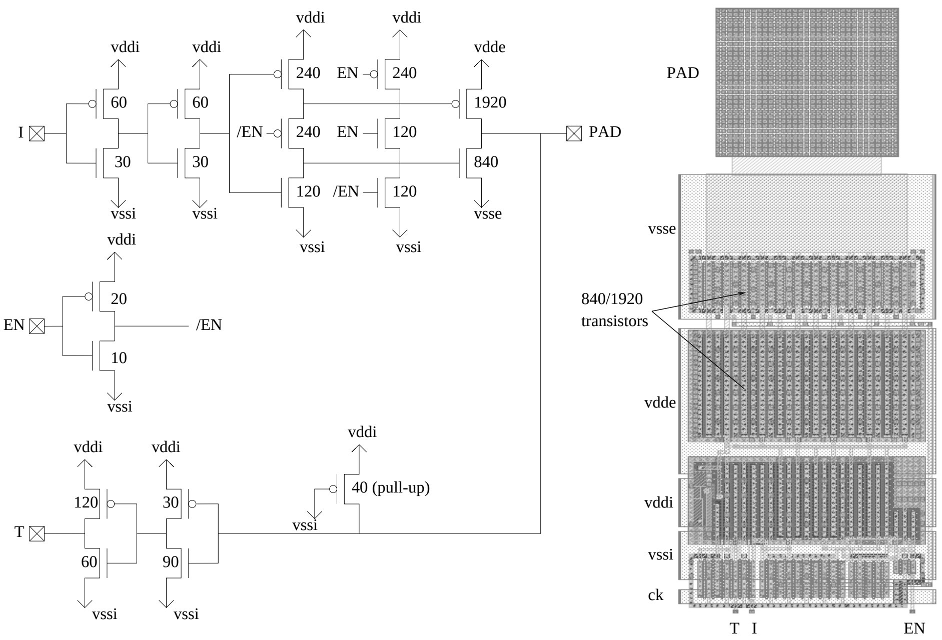

Figure 2: Schematic view (left) and layout (right) of the tri-state pad (piot_sp)
The numbers associated to the transistors indicate their length, in λ

3.3 Regular vs weak pads

All the pads that may drive an external signal have two versions: the regular one and the weak one. A weak pad performs the same logical function as the corresponding regular one - but its final stage driver is twice as small as the regular one. This makes the pads more adaptative to the outside capacitance. For instance, driving a board wire with a too powerfull driver could cause overshoot and therefore noise.

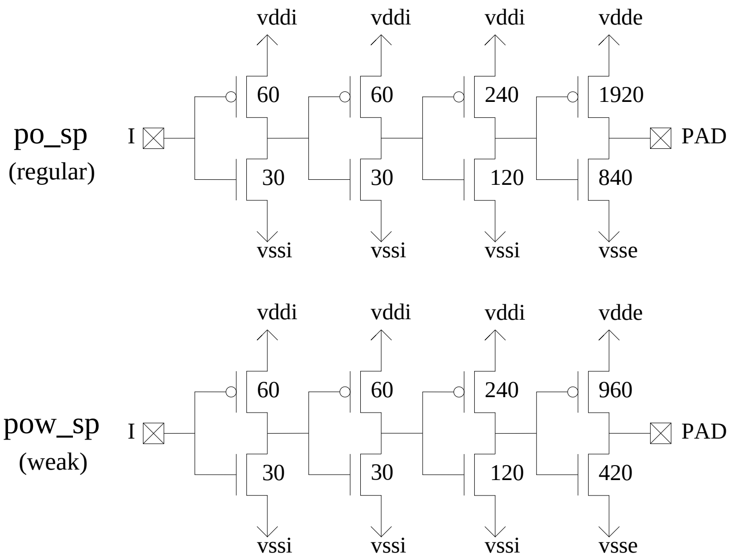

Figure 3: Schematic view of both version of the output pad

3.4 A built-in clock ring

Instead of using a simple input pad to connect the circuit clock, this library offers 5 pads dedicated to a built-in ring. The ring enables connection to the clock signal everywhere around the core. It is an easy way to build a well connected clock grid in the core.

Actually, the clock ring is a wire — called **ck** — that exists in all the pads. It is driven by a dedicated pad called `pck_sp`. This pad must be instanciated at least once in the pad ring when reasonable clock distribution is required. This pad does not provide any signal to the core, instead it drives the clock wire that is present inside all the pads.

The only way to drive the core clock signal is to use one or several `pv*ck_sp` pads. These pads both correspond to a power supply pin, and provide a clock connector to the core. Figure 4 shows the way clock pads must be used. The number of `pv*ck_sp` pads must be calculated according to the core clock capacitance estimation.

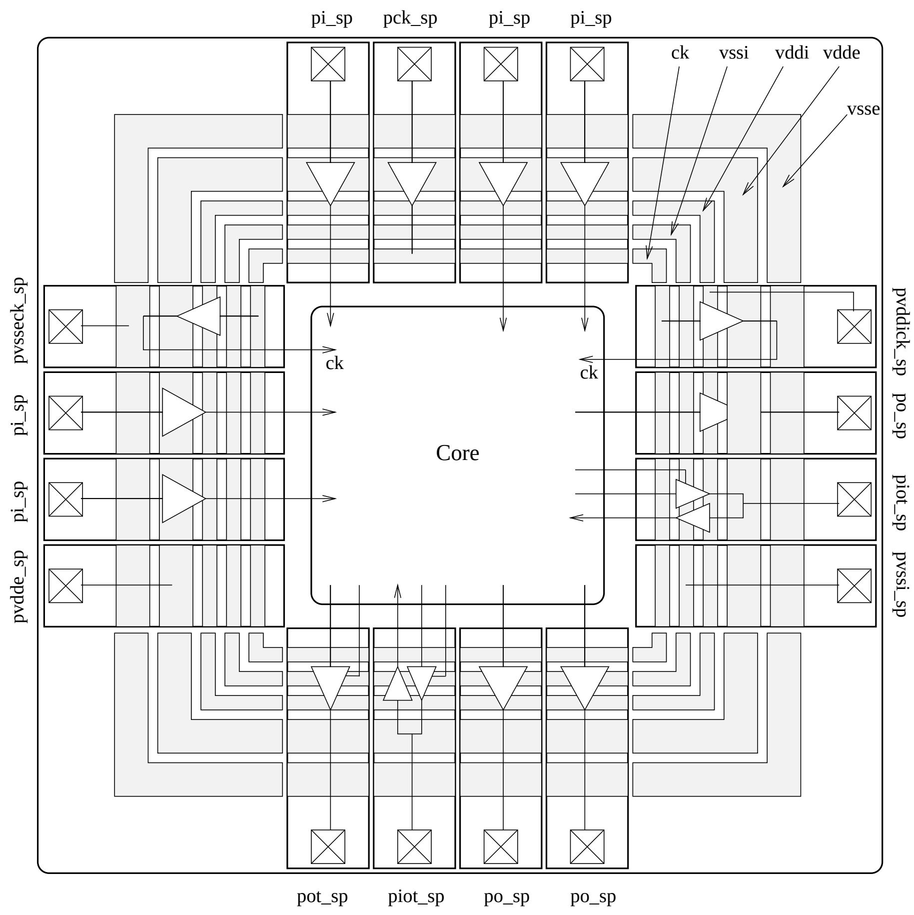

Figure 4: Using the clock ring

4 Symbolic to real translation

4.1 Pad hierarchy

The only part of the pad that cannot be correctly designed at the symbolic level is the bounding area: passivation openings, metal and via layers organized in a special fashion.

This led us to build a two-level hierarchy for each pad: each pad has a main view that instanciates a logical block and a bounding-area block. For instance, the input pad main view is called `pi_sp` which instanciates `pali_sp` — the logical block containing all the transistors and the equipotentials of the pad — and `padreal` — the bounding area —.

Figure 5 shows this hierarchy. There is also a symbolic layout view of `padreal`, but it only is a preview of what has to be designed in real.

The real `padreal` view depends on requirement given by the foundry: it may be two metal layers plus passivation oppening, or it may be a via matrix or any other design.

The main advantage of having as much symbolic design as possible in a circuit is that it gives the opportunity to easily adapt it to several different technologies. A circuit designed using this pad library may be ported to different technologies provided it exists a real view of `padreal` for each technology. Indeed, designing only one cell to change the process is a low cost operation — let's say a quarter of an hour —.

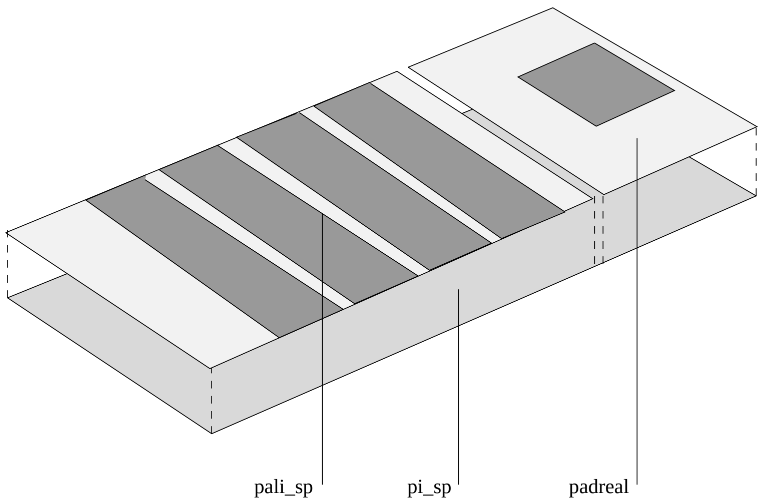

Figure 5: The hierarchy of the input pad - `pi_sp` instanciates `padreal` and `pali_sp`

4.2 Pad spacing vs process scale

All pads are 172 λ wide. The bounding area size is indicated in microns by the foundry. Let's have a x microns technology which requires a y microns wide bounding area and z microns pad spacing. The bounding area width is equivalent to $y/x\lambda$, thus the thiner process the bigger area when compared to the rest of the pad. The pad spacing is $z/x\lambda$. If $z/x = 172$ then the pads may be abutted. If $z/x < 172$ then pads overall size is a bigger constraint than the pad spacing requirement, so you lose some area on pad limited chips. If $z/x > 172$ then abutting the pads will violate the design rules: you need to leave some more space between the pads to respect the pad spacing specification.

Figure 6 shows the evolution of the bounding area size when changing the technology. With sufficiently thin technology, the designer can adapt the pad spacing by spacing the pads. For 1 micron technology and larger, the size of the pad is the constraint that determines the pad spacing.

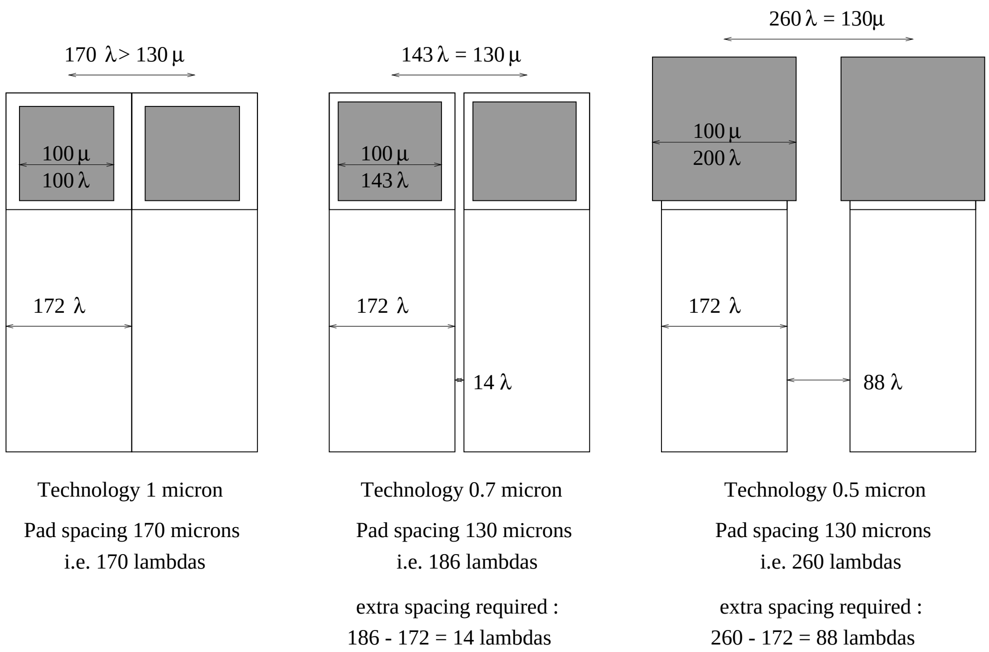

Figure 6: Pad spacing situations for three different technologies

5 Experimental results

A small test chip including all the symbolic pads has been realized. It includes all the pads and makes possible many functional and physical tests. It prooved that the pads are valid, and showed they were fast enough for the applications we were aiming at. Figure 7 shows the basic output signal of an output pad, when attacked by a clock. The signal actually goes through 2 successive pads : an input pad and an output pad.

Pads have been used – since this test – for bigger circuits. The Rapid-2 project involved the design of two circuits of approximately 170,000 transistors. The pad rings were approximately made of 200 pads in a 7mm side square. Tests showed the pads behave correctly, concerning in/out functionalities, and clock ring timing.

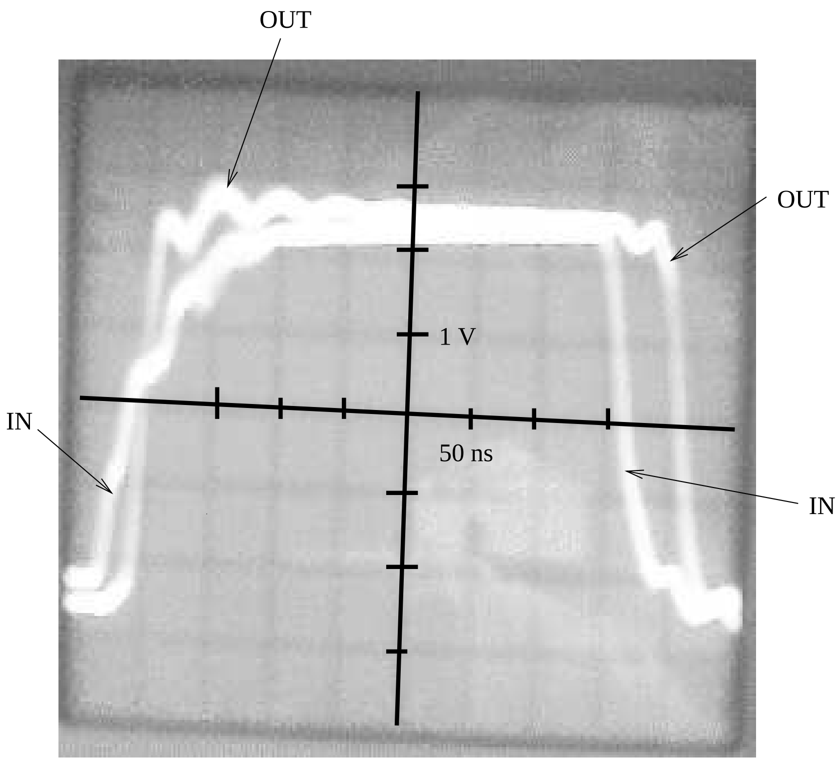

Figure 7: signal

6 conclusion

In this report, we have shown the interest in having a symbolic approach also for pad design: a whole circuit can now be described using a symbolic layout approach, and its final layout can be targeted to many technologies without having to reroute anything. We have proven that this approach works fine in developing a small test circuit. We have also used the new symbolic pad library while developing important circuits: it has been prooved to be easy to use and valid.