Cell Library Design

- Teams of four or five.
- Up to sixteen gates to be designed:
  - Scannable D-type flip flop, Scannable Register
  - Full Adder, Half Adder
  - 2 input Multiplexer
  - 2 input NAND/NOR/XOR/AND/OR
  - 3 input NAND/NOR, 4 input NAND
  - Inverter, Buffer, Tristate Buffer
- Additionally five support cells to be designed:
  - Tie High and Tie Low cells
  - Row Crosser cell
  - Left and Right End of Row cells

Cell Library Design

- Core Library (12 gates)[1] for a team of five.
  - Scannable D-type flip flop, Scannable Register
  - Full Adder, Half Adder
  - 2 input Multiplexer
  - 2 input NAND/XOR
  - 3 input NAND, 4 input NAND
  - Inverter, Buffer, Tristate Buffer
- Support cells required for all teams:
  - Tie High, Tie Low, Row Crosser, Left and Right End of Row cells
- Optional gates[2] for a team of five.
  - 2 input NOR/AND/OR, 3 input NOR

[1]These 12 are the only cells which you will include in your cell library databook

[2]Optional gates require no documentation and will receive no marks but may be useful for design exercise 4.

Cell Library Design

- Core Library (9 gates)[3] for a team of four.
  - Scannable D-type flip flop, Scannable Register
  - Full Adder
  - 2 input Multiplexer
  - 2 input NAND
  - 3 input NAND
  - Inverter, Buffer, Tristate Buffer
- Support cells required for all teams:
  - Tie High, Tie Low, Row Crosser, Left and Right End of Row cells[4]
- Optional gates[5] for a team of four.
  - Half Adder, 2 input NOR/XOR/AND/OR, 3 input NOR, 4 input NAND

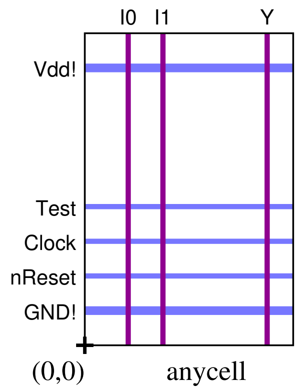
[3]These 9 are the only cells which you will include in your cell library databook

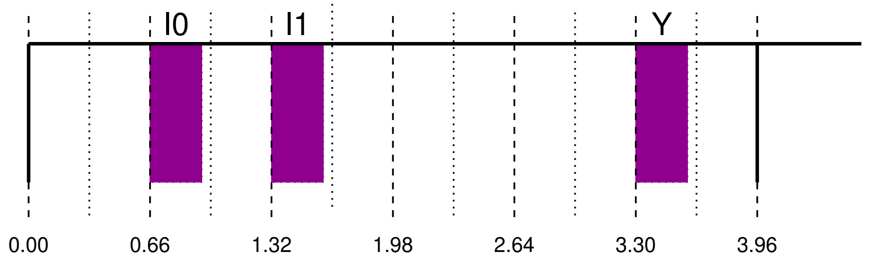
[4]leftend cell for team of 3 contains no buffer

[5]Optional gates require no documentation and will receive no marks but may be useful for design exercise 4.

Common Cell Specification

- Technology: TSMC 0.18μm CMOS (use `magic -T tsmc180`)
- All transistors fixed size: W_P = 1.78μm , W_N = 0.98μm , L_P = L_N = 0.18μm
- Horizontal i/o in metal1 (including power/ground and global signals)
- Vertical i/o in metal2 (aligned on 0.66μm grid)
- Power rails: 0.6μm (all other dimensions kept to a minimum)
- Taps arranged along power rails - joined by continuous nohmic/pohmic
- Cell aligned to origin (0,0) - cell width divisible by 0.66μm
- Gate Matrix construction (where possible)

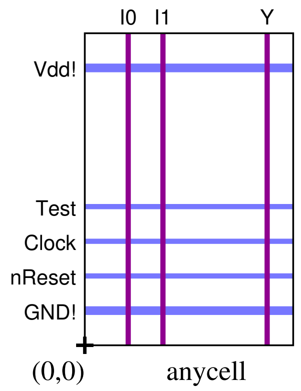


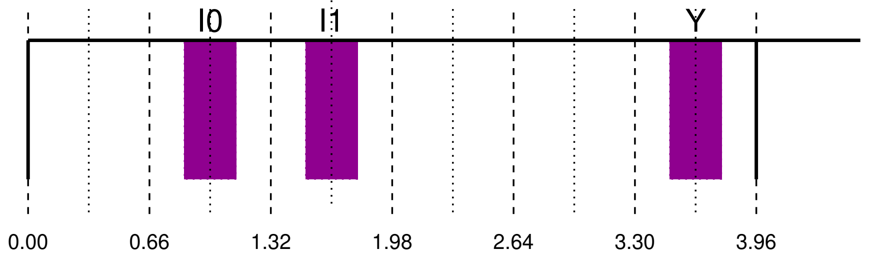


Alignment of vertical i/o to 0.66 micron grid
(to support magic auto–router)

Common Cell Specification

- Technology: TSMC 0.18μm CMOS (use `magic -T tsmc180`)
- All transistors fixed size: $W_P = 1.78\mu m$, $W_N = 0.98\mu m$, $L_P = L_N = 0.18\mu m$
- Horizontal i/o in metal1 (including power/ground and global signals)
- Vertical i/o in metal2 (aligned on 0.66μm grid)
- Power rails: 0.6μm (all other dimensions kept to a minimum)
- Taps arranged along power rails - joined by continuous nohmic/pohmic
- Cell aligned to origin (0,0) - cell width divisible by 0.66μm
- Gate Matrix construction (where possible)





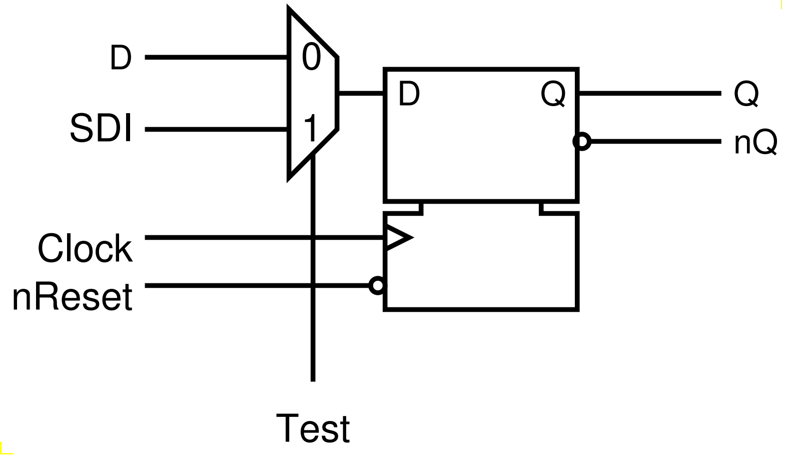
Alignment of vertical i/o to 0.66 micron grid
(to support magic auto–router)

Scannable D-type flip-flop

- triggered on rising edge of clock
- with asynchronous active low reset

Scan path support

- In normal operation the D-type receives data on its D input but when the Test signal is 1, data is received on the *scan data in* (SDI) input.



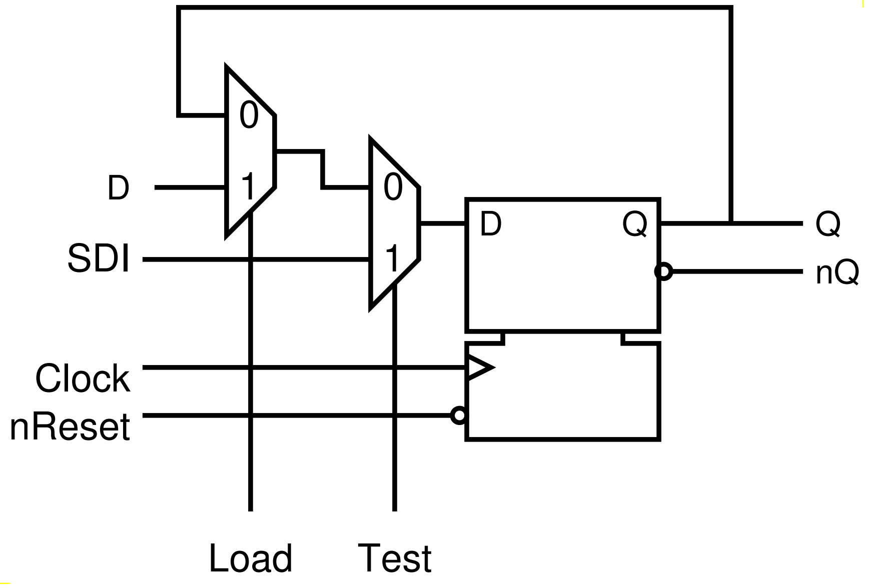
Cell Library Design

Scannable Register

- synchronous load signal allows for a new value to be loaded
- otherwise Q is fed back to the input

Scan path support

- Scan operation overrides register operation.

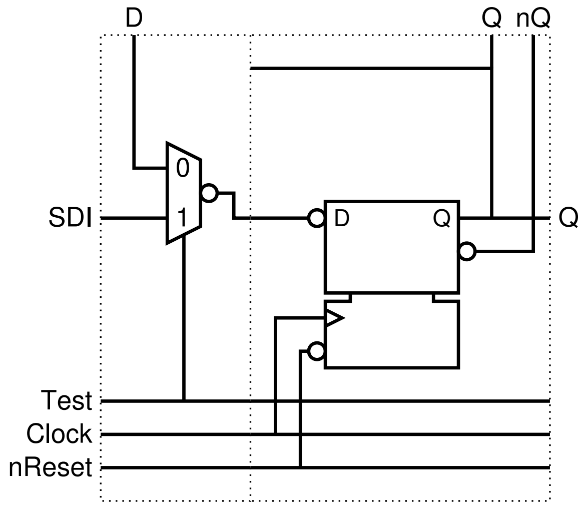


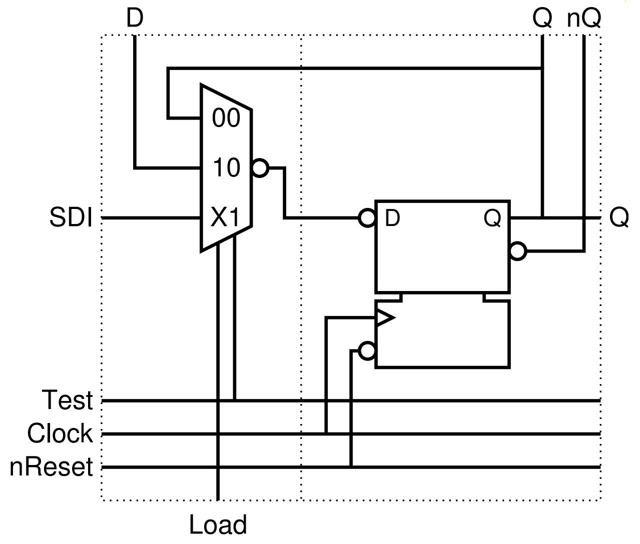
Re-use of cells

To avoid duplication of effort on the D-type, the actual cells designed will be:

- Raw D-type (no multiplexer) with inverted input
- 2 and 3 input scan multiplexers with inverted output

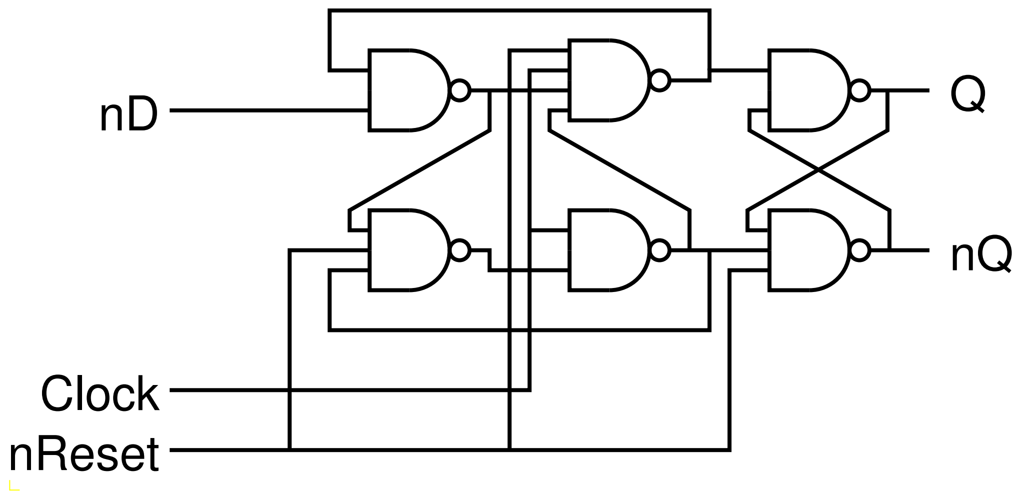
The Scannable D-type is constructed by butting the raw D-type and the 2 input scan multiplexer. The scan path is automatically joined when scannable gates are butted. The cell inputs and outputs are arranged to facilitate this butting:

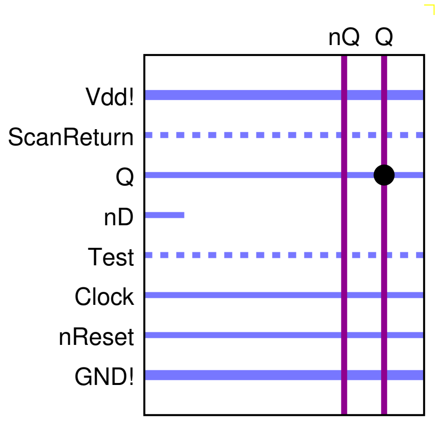




Raw D-Type

- 6 nand gate implementation
- horizontal i/o in metal1
- vertical i/o in metal2
- *order of signals may be customized by teams*

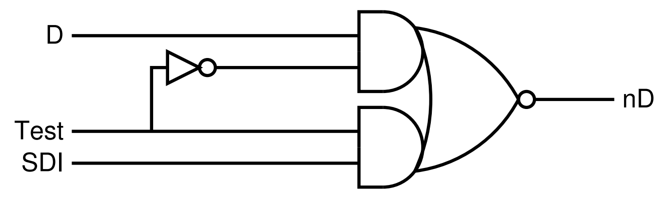


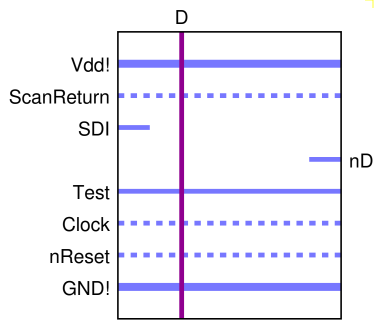


rdtype

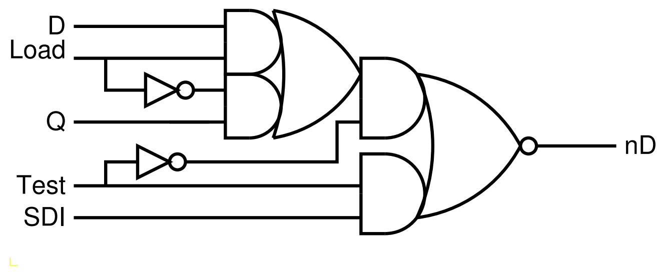
Scan Multiplexers

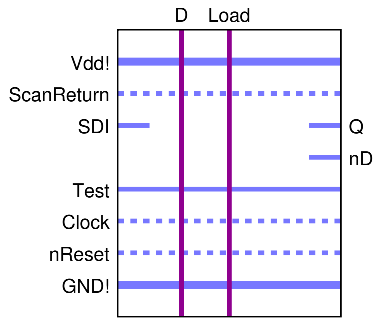
- compound gate implementation
- investigate Euler paths





smux2



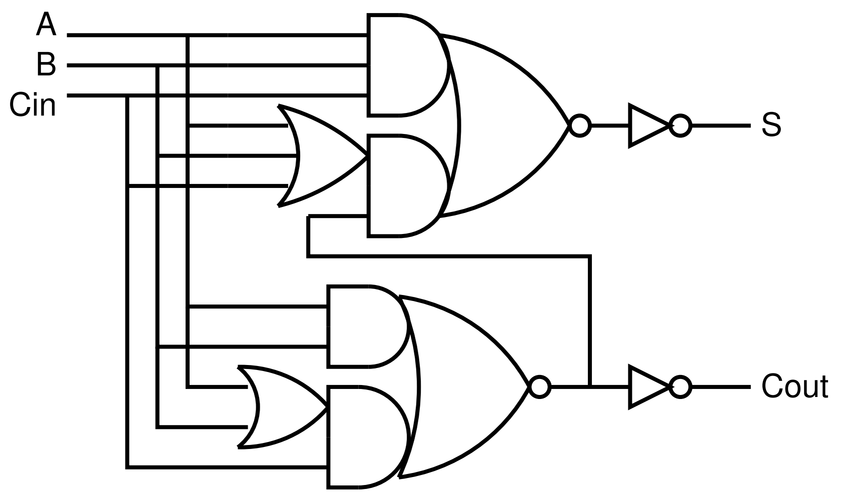


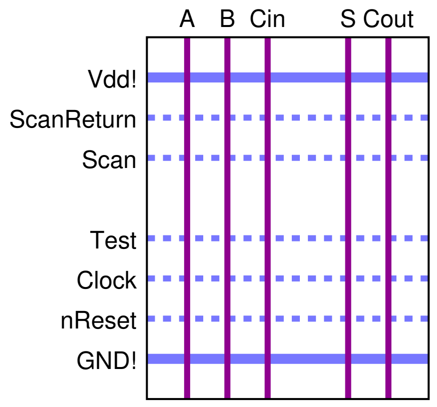
smux3

Cell Library Design

Full Adder

- compound gate implementation
- investigate Euler paths
- *Clock, nReset, Test, Scan and ScanReturn pass through without connection*

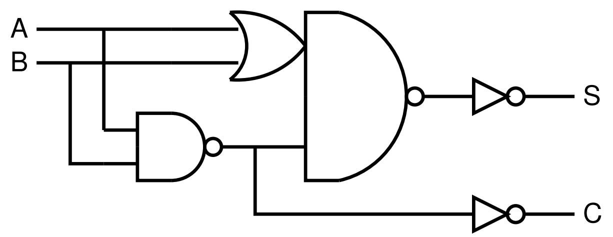


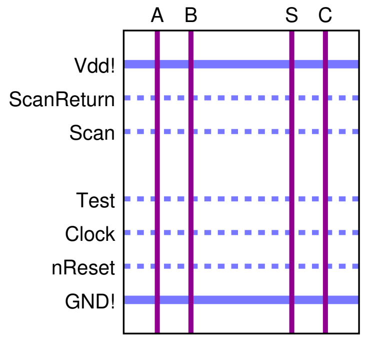


fulladder

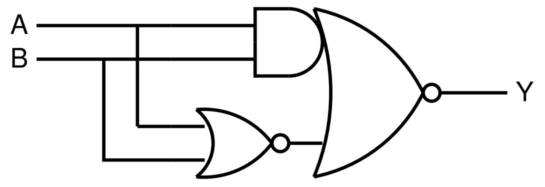
Half Adder and XOR

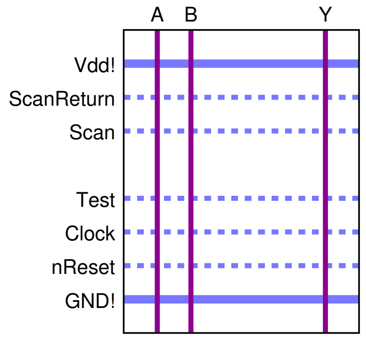
- compound gate implementation
- investigate Euler paths





halfadder



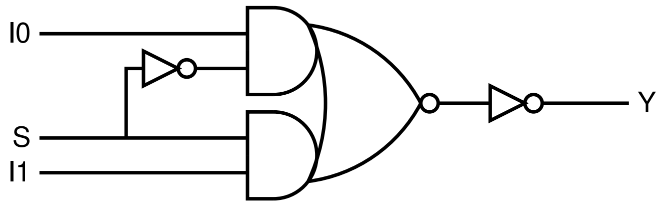


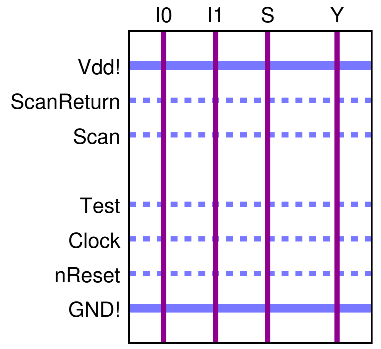
xor2

Standard 2 Input Multiplexer

This gate is a modified version of the **smux2** cell

- extra inverter to give non-inverted output.
- vertical i/o (I0, I1, S, Y)
- Scan/Test pass through without connection

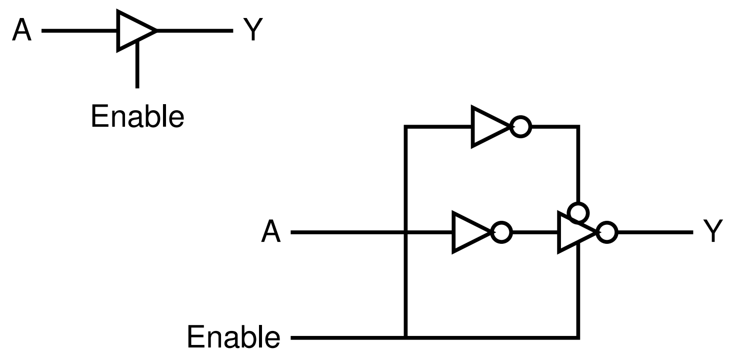


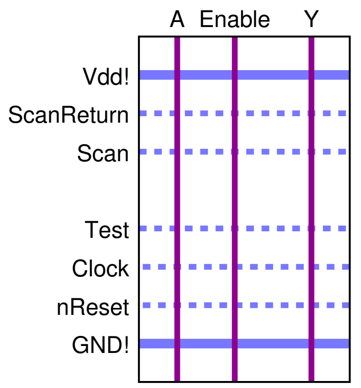


mux2

Cell Library Design

Tristate Buffer



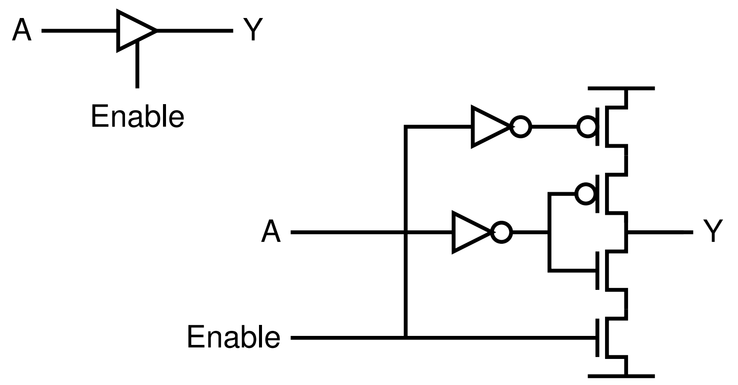


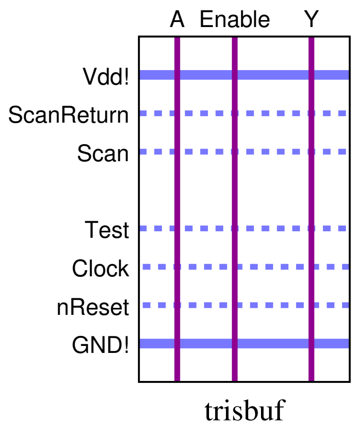
trisbuf

- trisbuf(Y, A, Enable)

Cell Library Design

Tristate Buffer

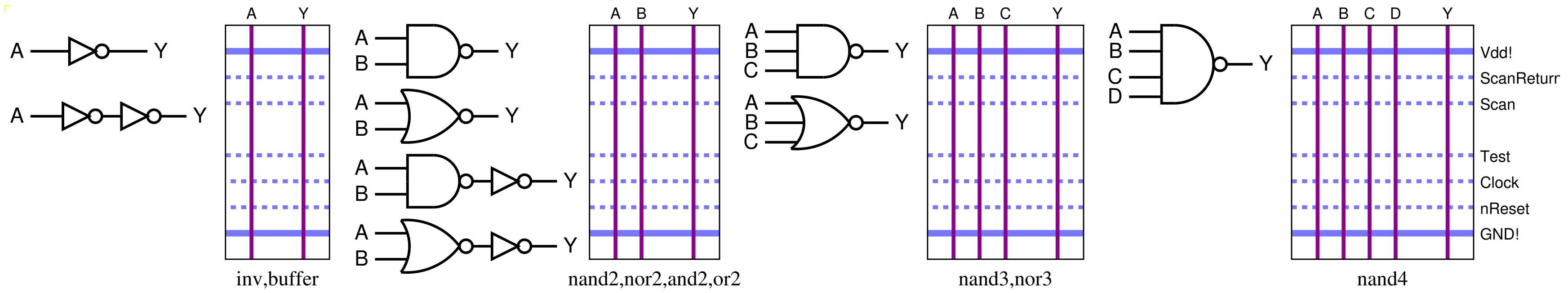




- trisbuf(Y, A, Enable)

Cell Library Design

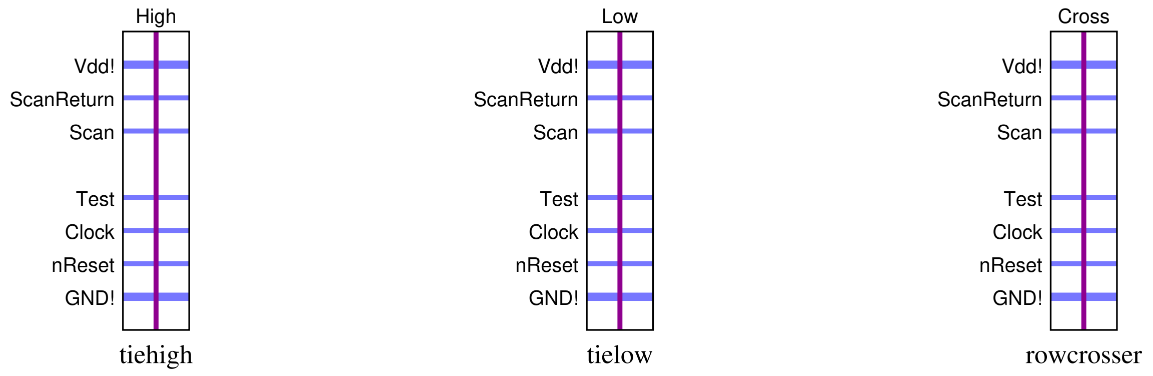
Simple Cells



inv,buffer nand2,nor2,and2,or2 nand3,nor3 nand4

- inv(Y, A)
- buffer(Y, A) - *use single Euler path for both transistors*
- nand2(Y, A, B)
- nor2(Y, A, B)
- and2(Y, A, B) - *use single Euler path for all three transistors*
- or2(Y, A, B) - *use single Euler path for all three transistors*
- nand3(Y, A, B, C)
- nor3(Y, A, B, C)
- nand4(Y, A, B, C, D)

Tie High, Tie Low, Row Crosser



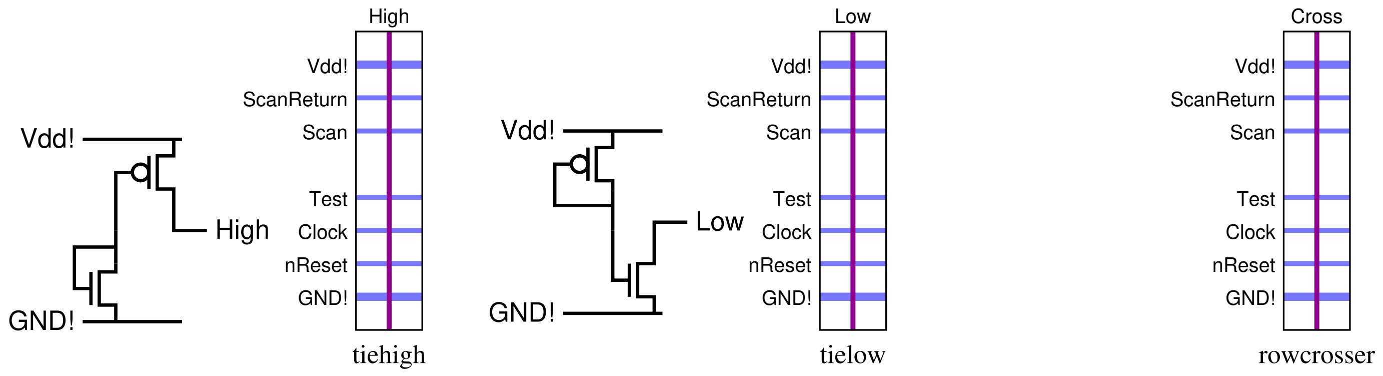
- Tie High & Tie Low

  These cells, named `tiehigh` and `tielow`, are used to tie unused cell inputs to either logic 1 or logic 0 as appropriate.

- Row Crosser

  This cell, named `rowcrosser`, is used where a signal needs to be routed over a row of cells.

Tie High, Tie Low, Row Crosser



- Tie High & Tie Low

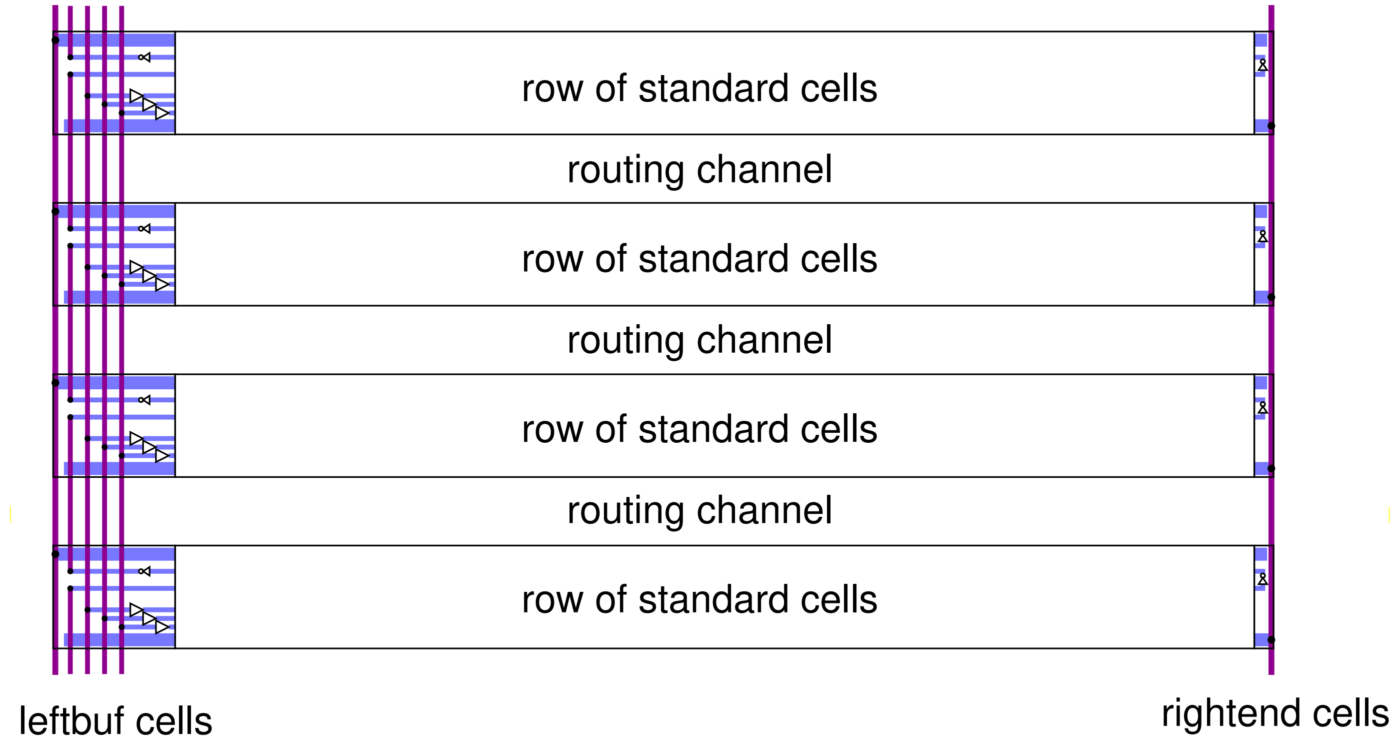
  These cells, named `tiehigh` and `tielow`, are used to tie unused cell inputs to either logic 1 or logic 0 as appropriate.

- Row Crosser

  This cell, named `rowcrosser`, is used where a signal needs to be routed over a row of cells.

End of Row Cells

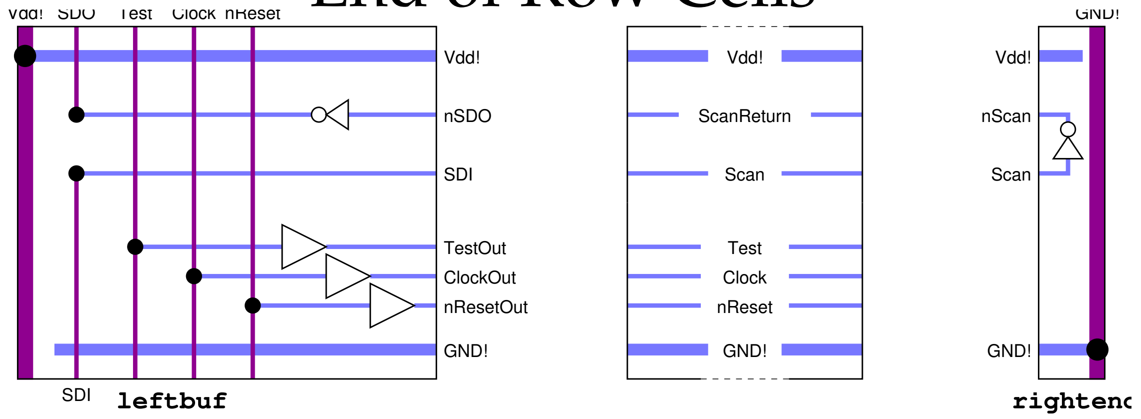
Each row of standard cells will have a `leftbuf` cell at one end and a `rightend` cell at the other.



These cells facilitate distribution of global signals: Vdd!, GND!, Clock, nReset and Test. They also include support for the routing of the scan path.

Cell Library Design

End of Row Cells



- Global Signal Distribution

  The `leftbuf` cell buffers and distributes Clock, nReset and Test.

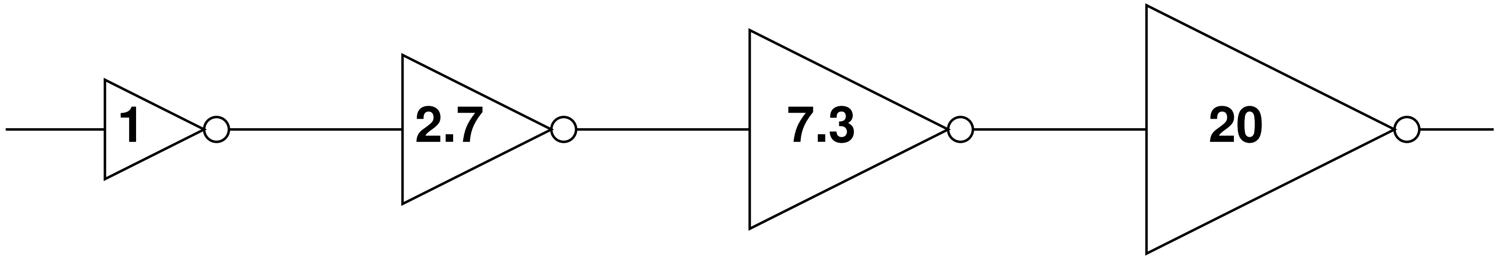
  The `leftbuf` cell distributes Vdd! while the `rightend` cell distributes GND!.

  Note: Vertical power rails in these cells should be 6μm wide to take account of the large supply currents.

- Scan Path Support.

  The inverters in the `leftbuf` and `rightend` cells provide buffering in the scan path and avoid name conflicts when the design is extracted.

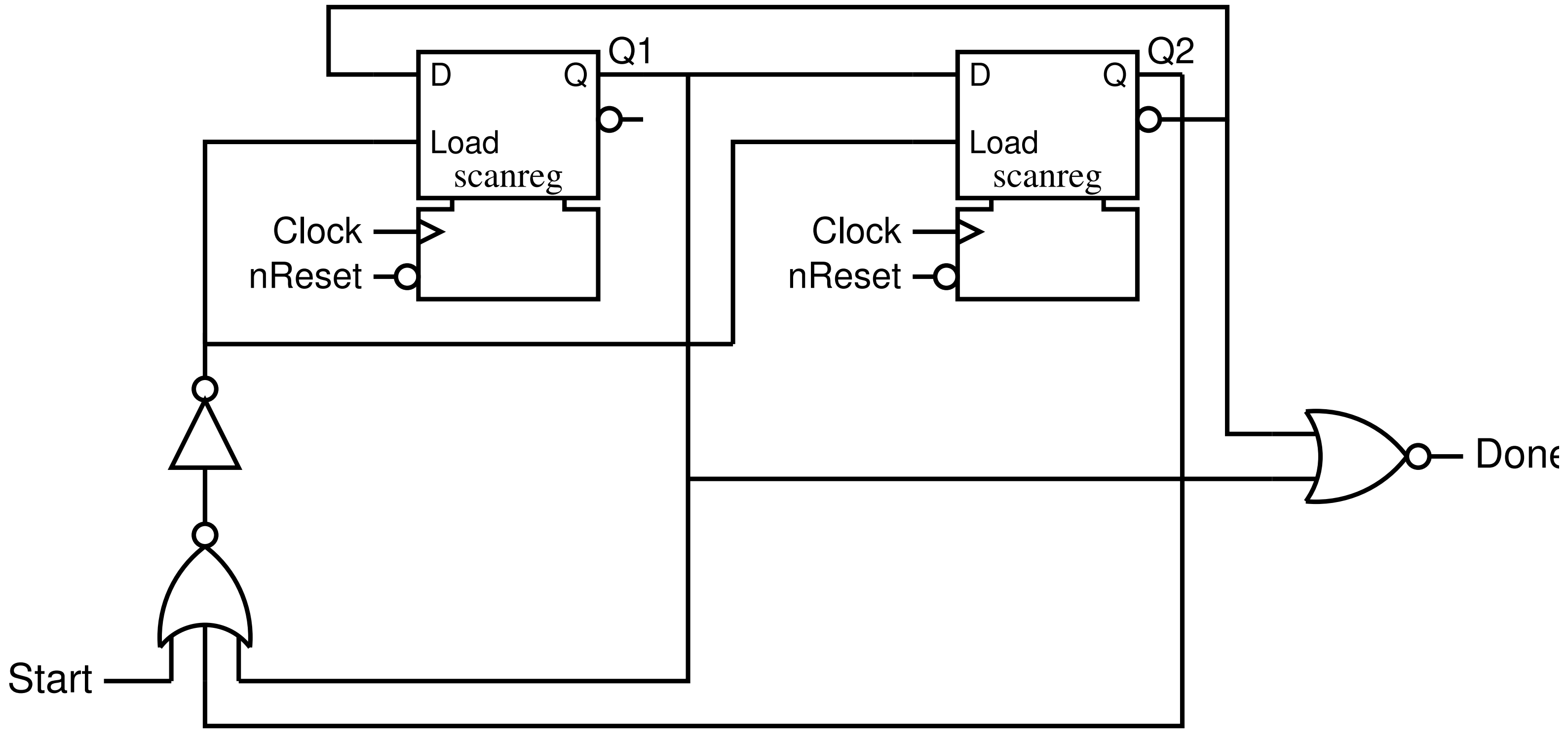
Left End of Row Cell: Buffer Design



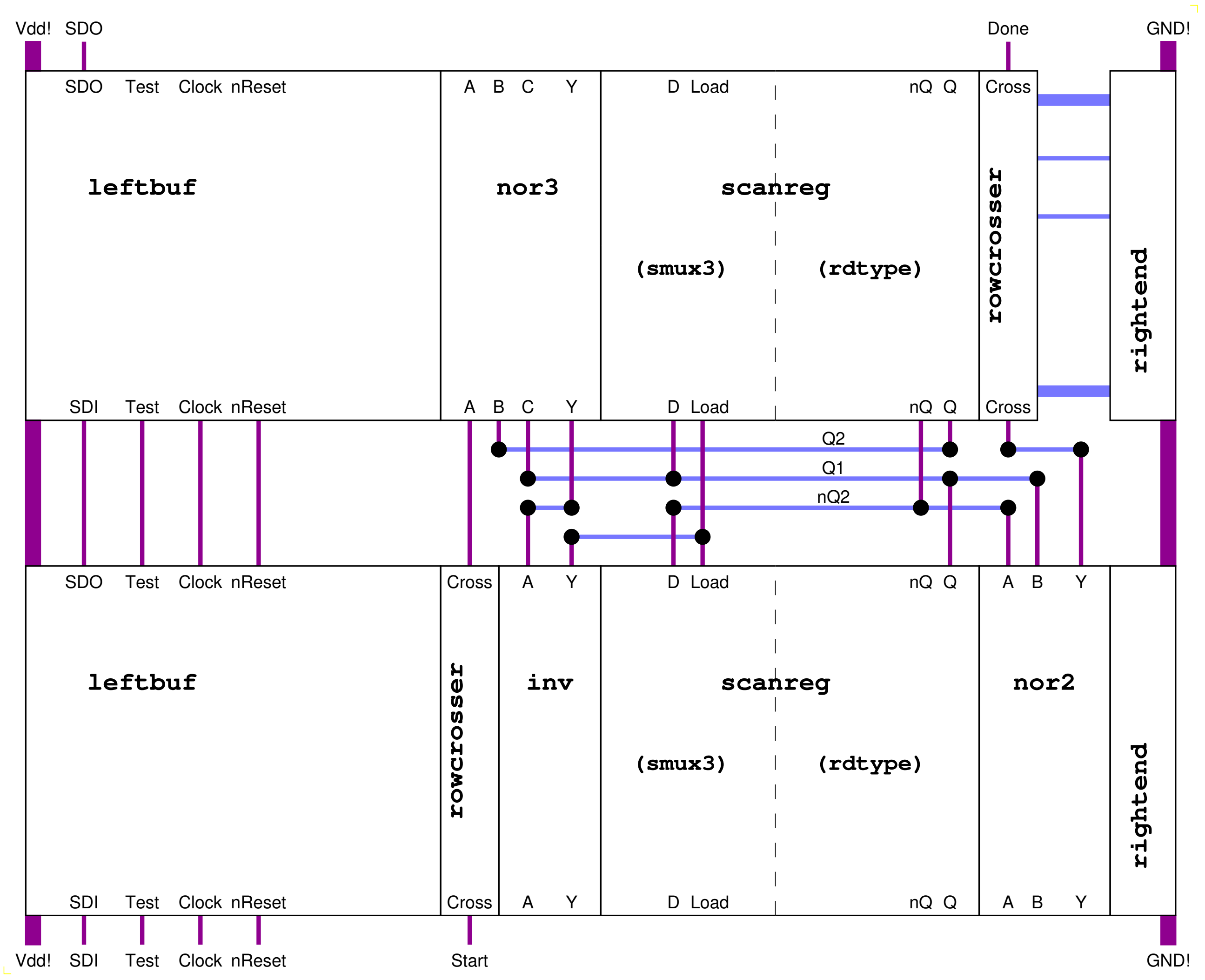
- Each non-inverting buffer is constructed from 4 inverters
- Each inverter is 2.7 times the size of the previous one
- The final inverter will drive up to 20 D-types with minimal clock skew
- The first inverter is a modified version of the standard inverter with transistor widths adjusted to give approximately equal rise and fall times (W$_P$ = 2.02μm , W$_N$ = 0.74μm)
- The complete leftbuf cell is designed without hierarchy

Example Design - Schematic

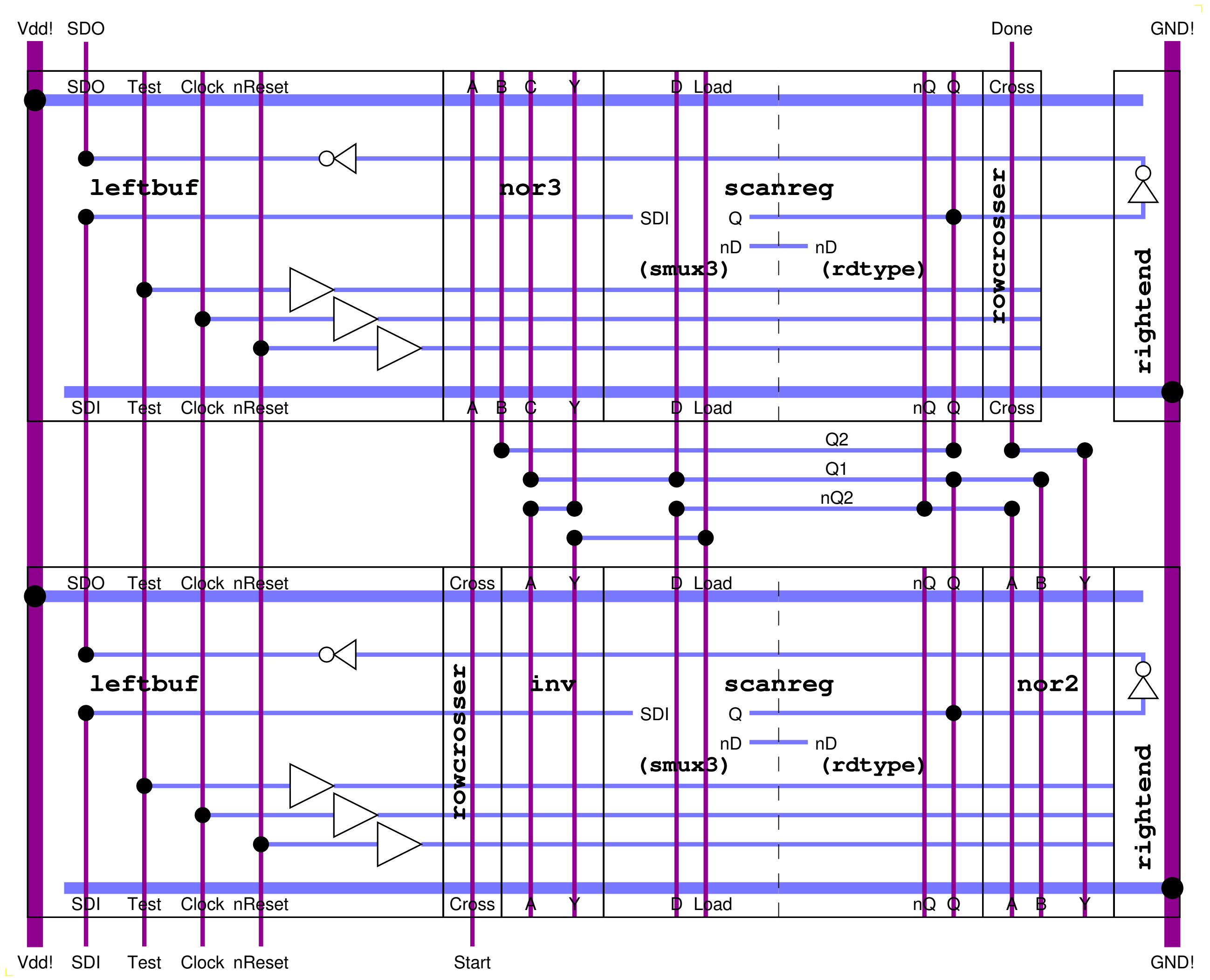
The use of the end of row cells and the row crosser cell can be seen in the following example:



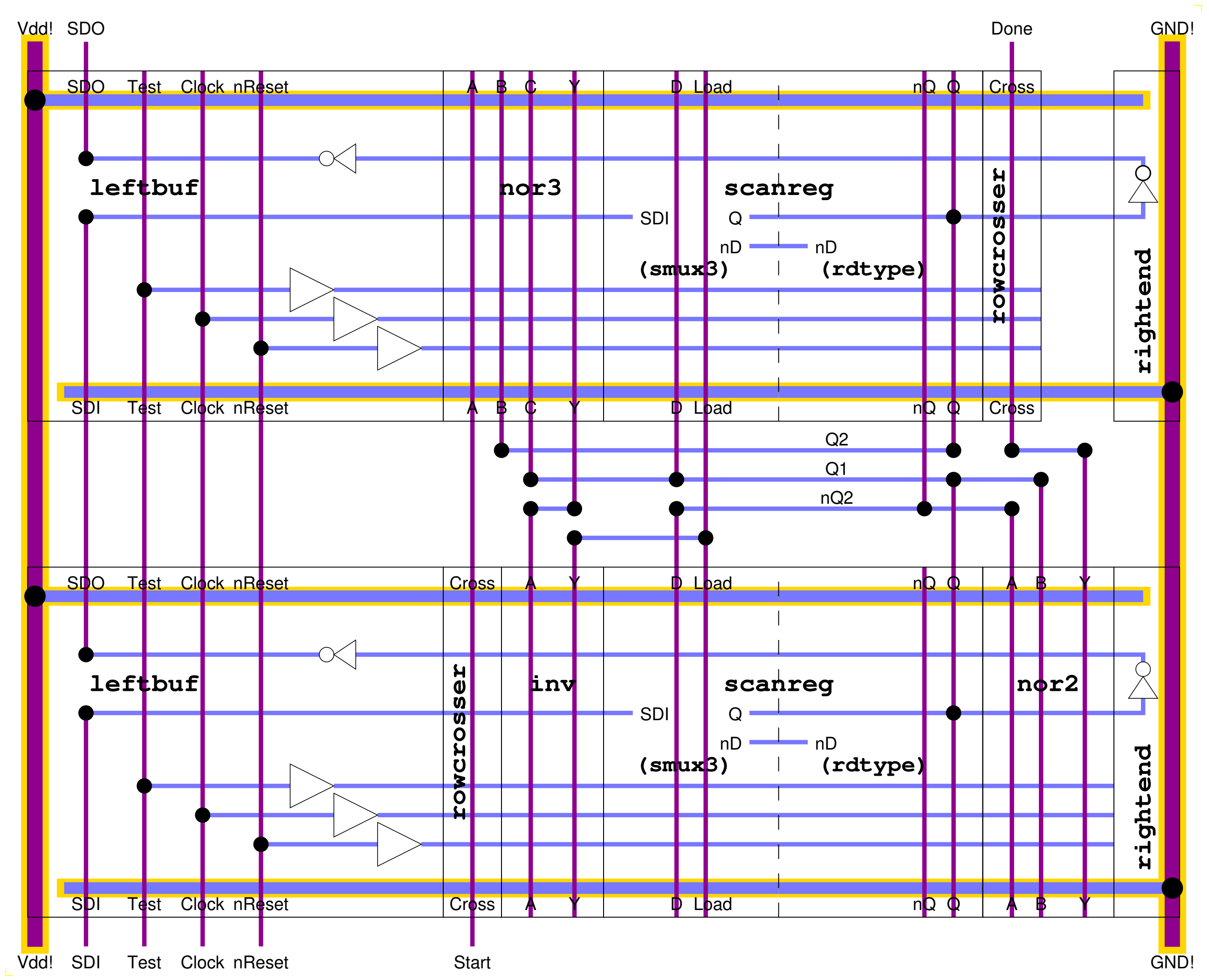
Example Design - Layout



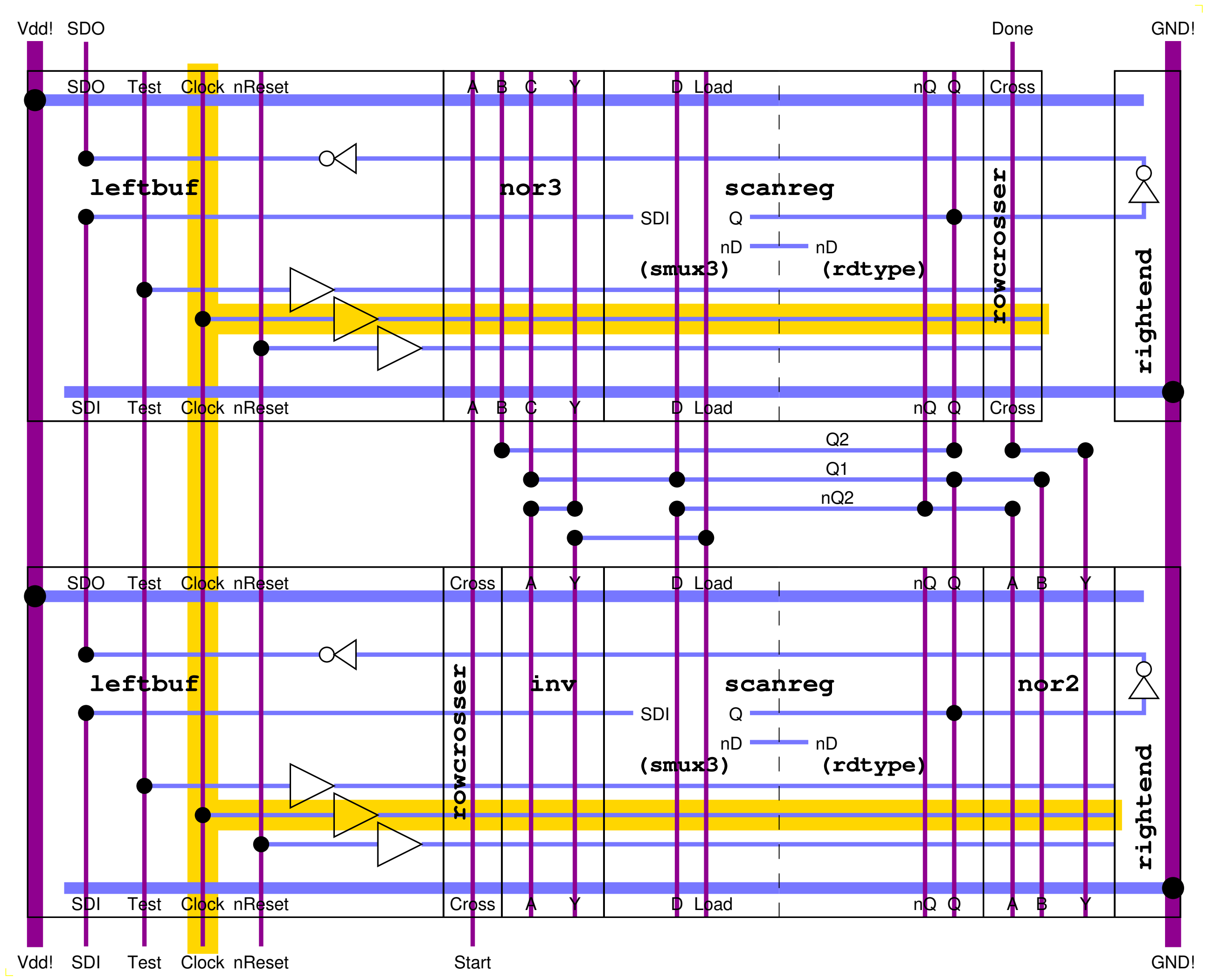
Example Design



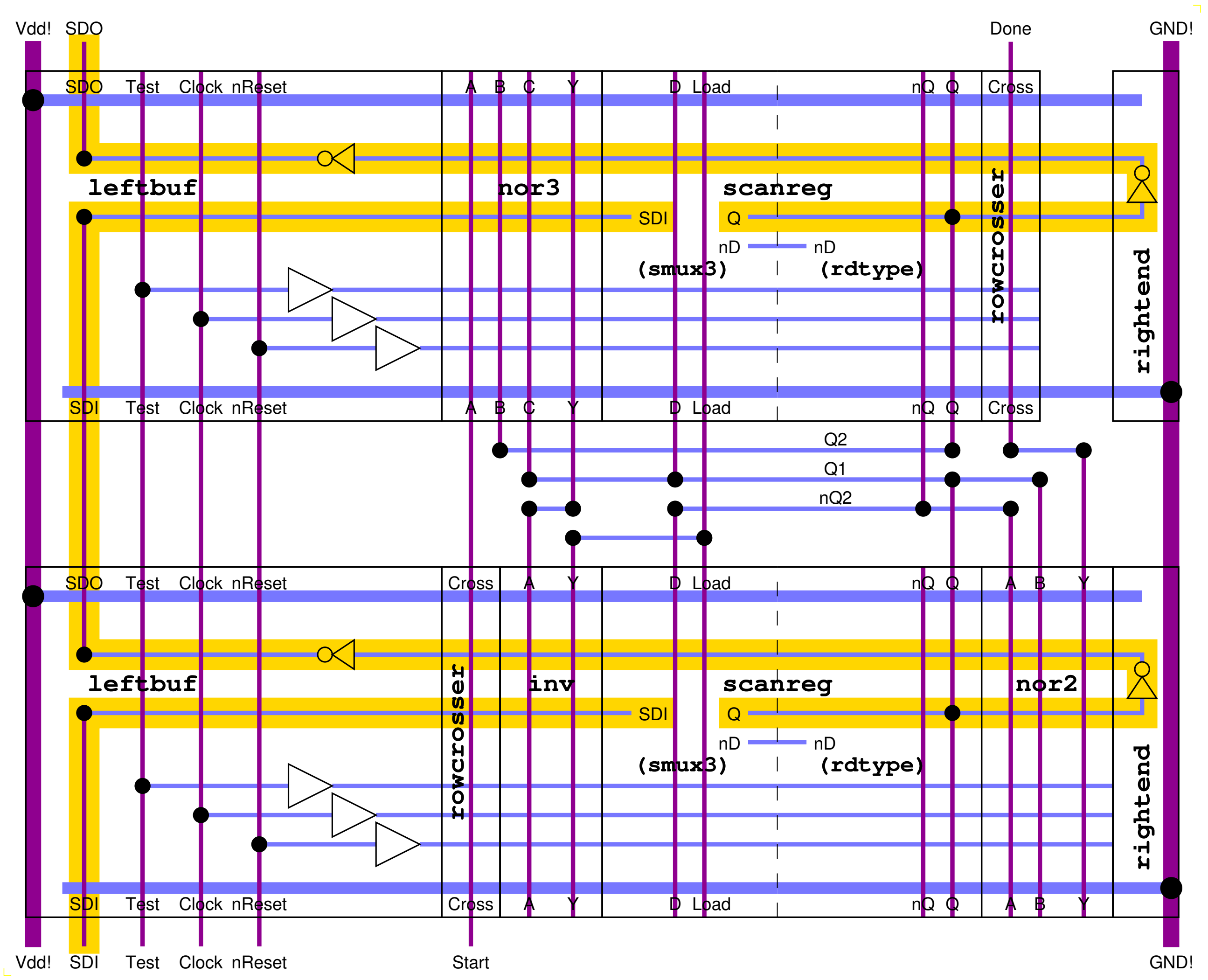
Example Design - Power Distibution



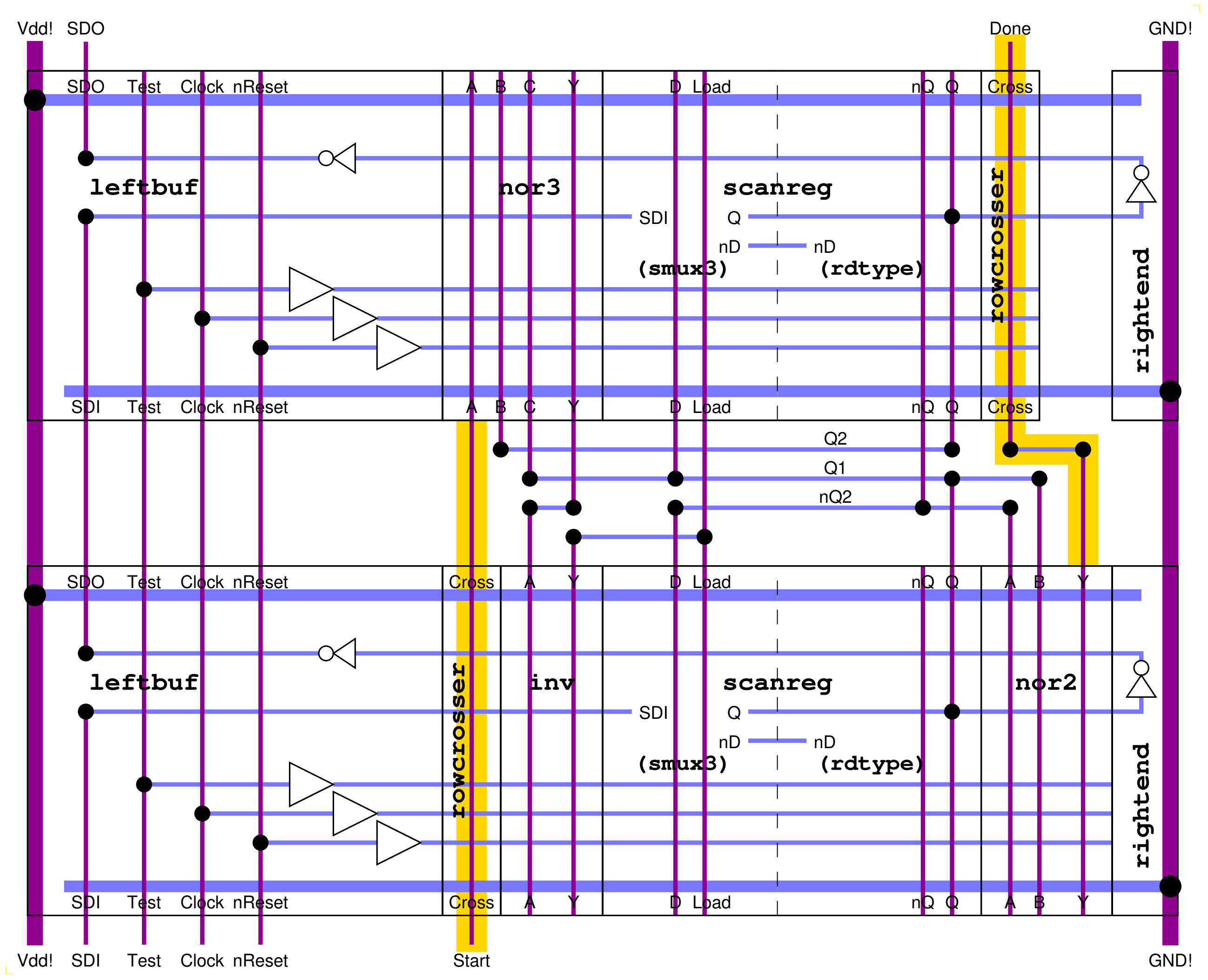
Example Design - Clock Distibution



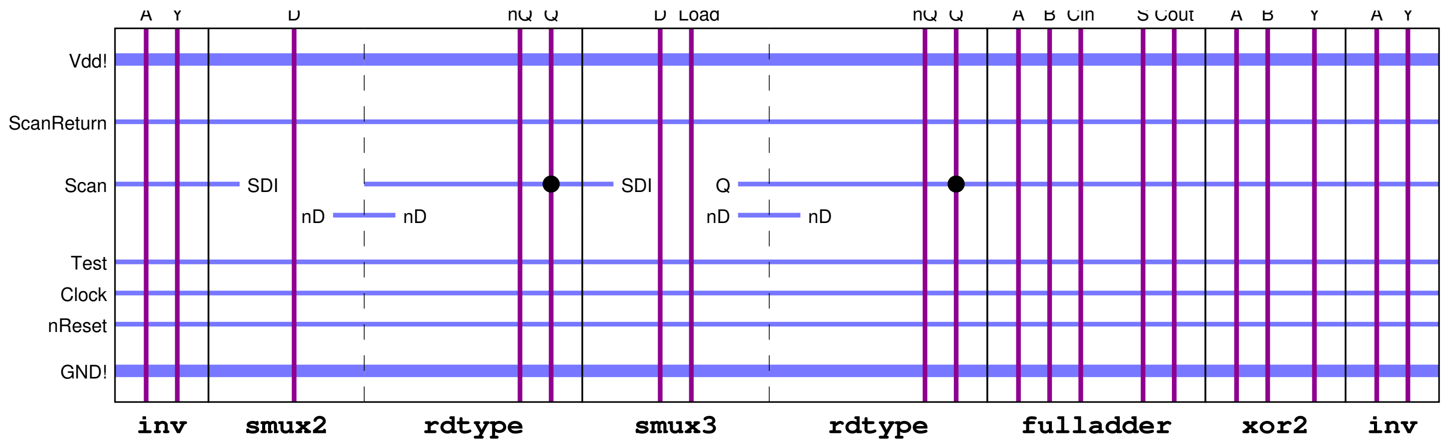
Example Design - Scan Path

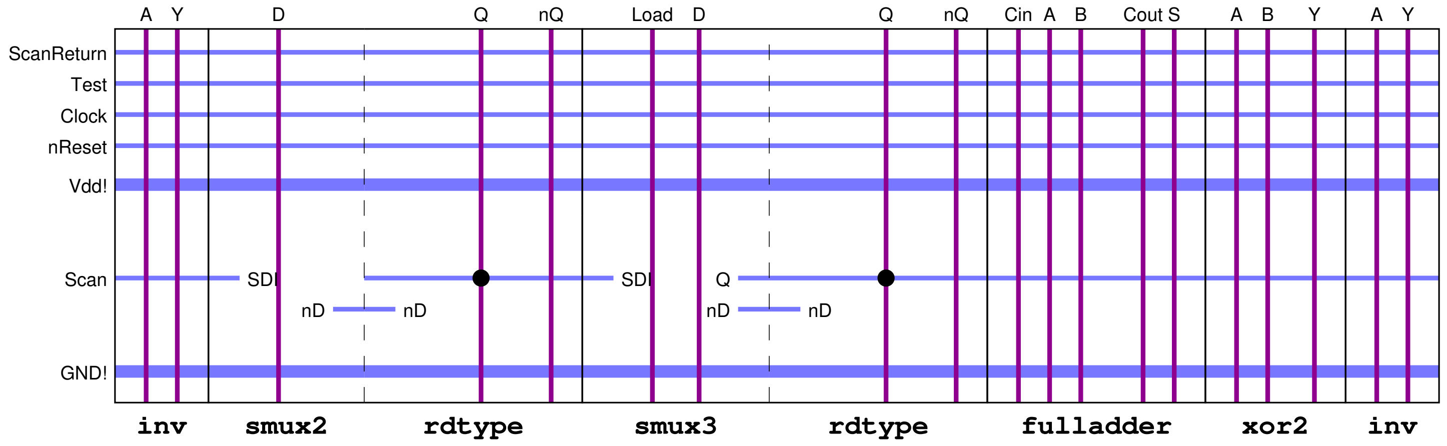


Example Design - Row Crosser



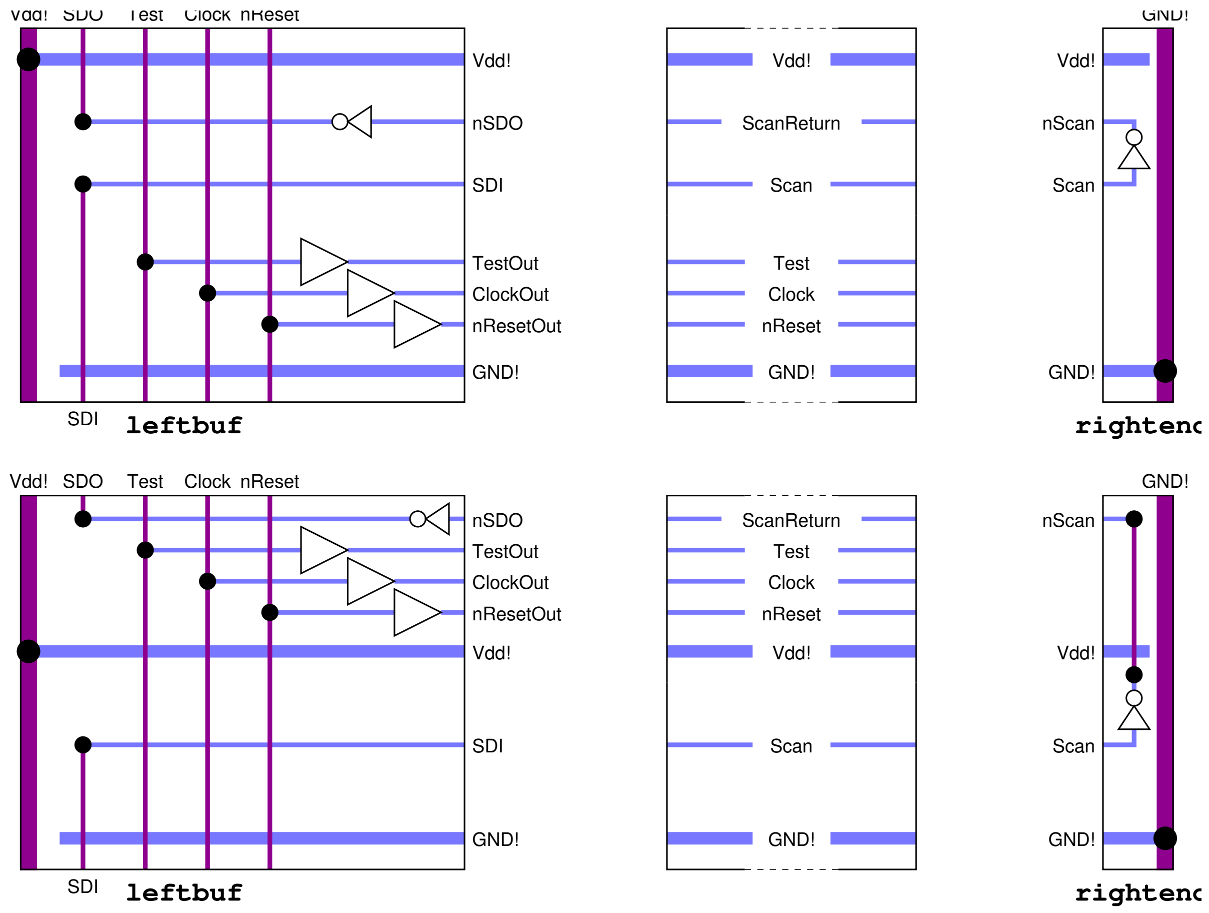
Cell Library Design





The arrangement of i/o is up to you. You must ensure the correct alignment of the horizontal i/o (especially the scan path) between cells.

Cell Library Design



The arrangement of horizontal i/o in end of row cells should match your other cells.

Division of Labour - Teams of 5

For teams of five, the tasks will be divided as follows:

- Raw D-Type `(rdtype)`
- Two Scan Multiplexer `(smux2, smux3)` and Standard Multiplexer `(mux2)`
- Full Adder `(fulladder)`
- Half Adder `(halfadder)` and XOR `(xor2)`
- End of Row Cells `(leftbuf, rightend)`

Note that for a complete Scannable D-Type we need the raw D-Type cell from one designer and a 2 input scan multiplexer from a second designer.

It is up to you who designs each of the other cells (`inv, buffer, nand2, nand3, nand4, trisbuf, tiehigh, tielow, rowcrosser,` *`nor2?, and2?, or2?, nor3?`*). You should use this flexibility as a means of balancing the load.

Division of Labour - Teams of 4

For teams of four, the tasks will be divided as follows:

- Raw D-Type `(rdtype)`
- Two Scan Multiplexer `(smux2, smux3)` and Standard Multiplexer `(mux2)`
- Full Adder `(fulladder)`
- End of Row Cells `(leftbuf, rightend)`

Note that for a complete Scannable D-Type we need the raw D-Type cell from one designer and a 2 input scan multiplexer from a second designer.

It is up to you who designs each of the other cells (`inv, buffer, nand2, nand3, trisbuf, tiehigh, tielow, rowcrosser,` *`halfadder?, nor2?, xor2?, and2?, or2?, nor3?, nand4?`*). You should use this flexibility as a means of balancing the load.

Submission – Files

- single team submission
- magic files in directory ~`/design/magic/tsmc180/cell_lib`
  - leaf cells:

    `rdtype smux2 smux3 fulladder` *`halfadder xor2`* `mux2 inv buffer nand2` *`nor2 and2 or2`* `nand3` *`nor3 nand4`* `trisbuf tiehigh tielow rowcrosser leftbuf rightend scandtype scanreg`
  - parent cell:

    `all`

    instances all leaf cells butted together in one row arranged as follows:

    `leftbuf inv smux2 rdtype buffer scandtype nand2 smux3 rdtype nand3 scanreg fulladder mux2 trisbuf tiehigh tielow rowcrosser` *`halfadder xor2 nor2 and2 or2 nor3 nand4`* `rightend`
- preparation script creates `handin.tar` file for web based handin

```
prepare_vlsi desex3
```

Deadline for file submission is 16:00 on Monday 2nd December.

Characterisation and Cell Library Databook

25% of the marks for the assignment will be awarded for a cell library databook which gives information about the cells.

The sort of information that we expect to see in a databook will include: *cell name, cell function, port names, propagation delays (preferably with a variety of different load capacitances), input capacitances, cell area, port positions*

The aim is use scripting[6] to generate a simple document describing the cells. Significant scripting skills are not expected and a proof of concept document (e.g. only dealing with a subset of the cells) is acceptable where scripting is employed. For teams who struggle with scripting, a hand generated document may be submitted as an alternative[7].

[6]Modern cell library databooks generate themselves. The cell designer creates a new cell and runs a script that adds another page to the databook.

[7]For all databooks, however generated, the accuracy of the presented numerical results and the consistency of the presentation is the very important (and will greatly influence the marks awarded).

Design Exercise Report

Main Body

- Title page + table of contents
- Brief introduction covering team decisions (max 2 sides)
- Design documentation for major cells (max 4 sides per designer)
- A conclusion sumarising the work done (max 1 side)

Appendices

- **A**: Team Management (max 1 side)
- **B**: Division of Labour Form (1 side)

The design exercise report provides evidence of the design, implementation and testing work that you have done and should include justifications for any important design decisions.[8]

[8]Since I will be reading your report in order to judge the quality of your work, you will need to do good work and write it up well in order to get a good mark.

Design Exercise Report

- Design documentation for major cells:

  `rdtype`, `smux2`, `smux3`, `fulladder`, `leftbuf`, *`halfadder`*, *`xor2`*[9]

  A sub-section should exist for each designer (max 4 sides), dealing with the major cells they have designed. This will cover one or two major cells and any work done on the merged cells: `scandtype` and `scanreg`.

  For each major cell, justification should be given for important design decisions. To support the text, the following information should be provided:[10]

  - transistor level circuit diagram
    with comments on derivation if appropriate
  - stick diagram
  - details of testing carried out

[9]only teams of five should provide documentation for the `halfadder` and `xor2` gates

[10]take care with diagrams, don't scan/capture my schematics, avoid screen capture of simulation and layout if possible, and avoid using inappropriate pre-defined circuit symbols (e.g. MOS transistors with visible substrate connection).

Design Exercise Report

- Presentation
  - Use full sentences and paragraphs
  - Further advice on report writing can be found on the module homepage.

- Editing
  - The page limits will require you to make decisions such as *what to take out and what to leave in?* or *how big should a figure be so that it is clear and easy to read but not so big that there is no room for the text to explain it?*

    If you leave the report to the last minute, you will not have time to consider these decisions properly.

Submission – Documentation

The following documentation files should be submitted via the ECS Handin system:

- The auto-generated cell library databook should be submitted as a simple .txt or .html file (if no script has been created, a PDF version of the databook may be submitted instead).
- The script used to generate the databook (if one was used) should also be submitted.
- A README.txt should be submitted which explains how to use the script and what it does (or, if there is no script, this file will tell us that).
- The design exercise report should be submitted as a PDF file.

Deadline for documentation is 16:00 on Wednesday 4th December.