CMOS Combi-3 (H.3)

20151111

Copyright (c) 2015 Young W. Lim.
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

References

Some Figures from the following sites

[1] http://pages.hmc.edu/harris/cmosvlsi/4e/index.html
Weste & Harris Book Site

[2] en.wikipedia.org

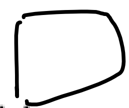
[3] Digital Integrated Circuits : A Design Perspective,
Jan M. Rabaey,
(http://bwrcs.eecs.berkeley.edu/Classes/IcBook/)

[4] Digital Electronics and Design with VHDL
Pedroni

(http://bwrcs.eecs.berkeley.edu/Classes/IcBook/)

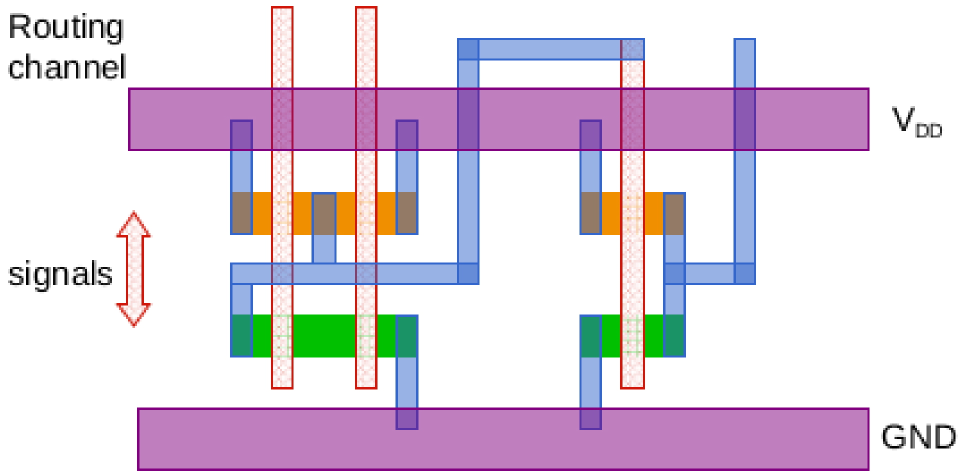
Cell Design

- Standard Cells
  - General purpose logic
  - Can be synthesized
  - Same height, varying width
- Datapath Cells
  - For regular, structured designs (arithmetic)
  - Includes some wiring in the cell
  - Fixed height and width

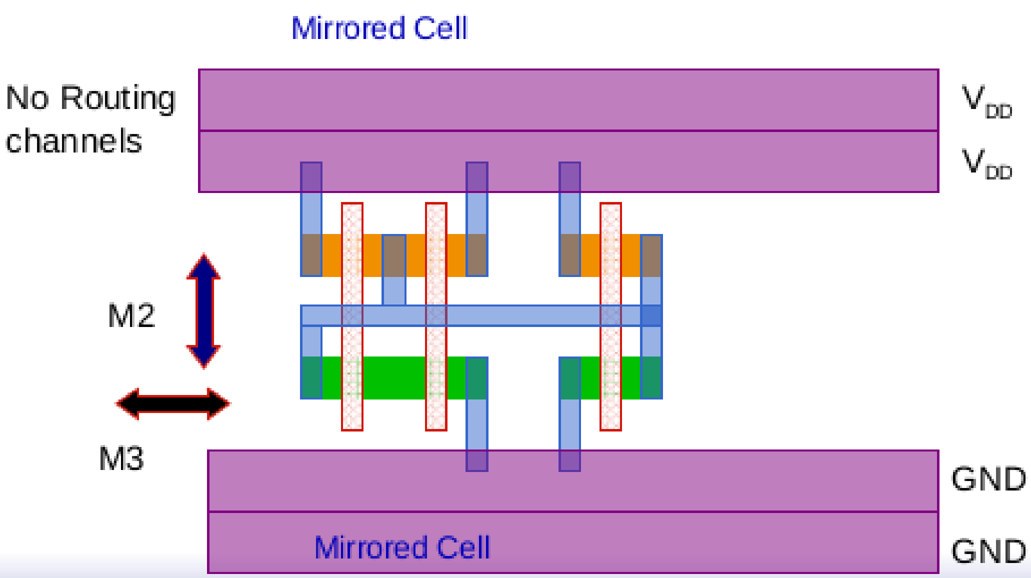


(http://bwrcs.eecs.berkeley.edu/Classes/IcBook/)

Standard Cell Layout Methodology – 1980s

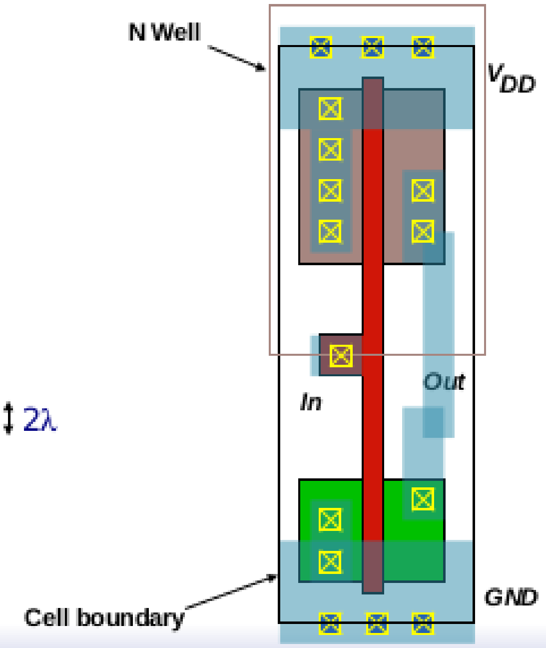


Standard Cell Layout Methodology – 1990s



(http://bwrcs.eecs.berkeley.edu/Classes/IcBook/)

Standard Cells



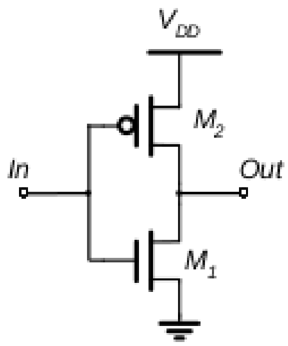
Cell height 12 metal tracks
Metal track is approx. $3\lambda + 3\lambda$
Pitch =
repetitive distance between obje

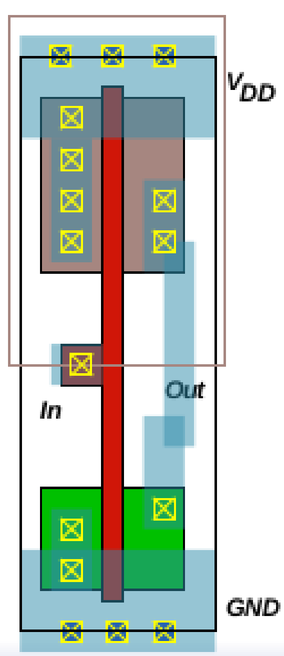
Cell height is "12 pitch"

Rails ~10λ

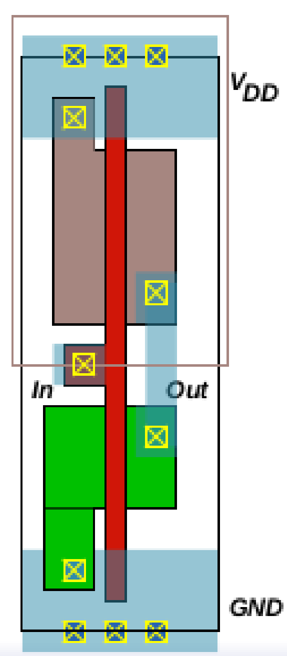
Standard Cells

With minimal diffusion routing



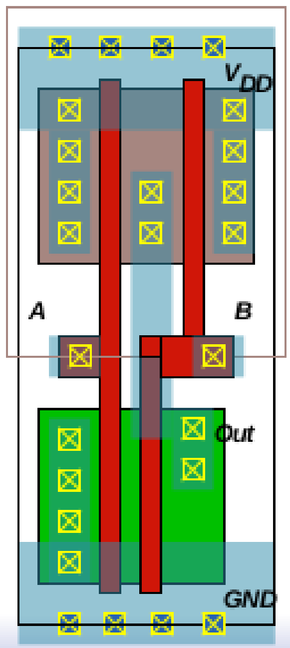


With silicided diffusion

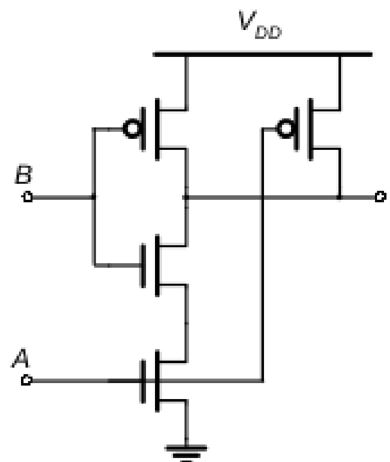


(http://bwrcs.eecs.berkeley.edu/Classes/IcBook/)

Standard Cells

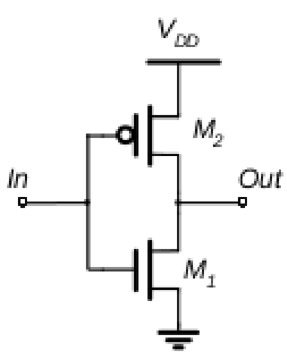


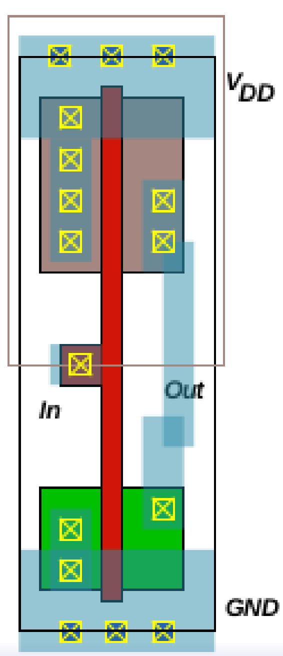
2-input NAND gate



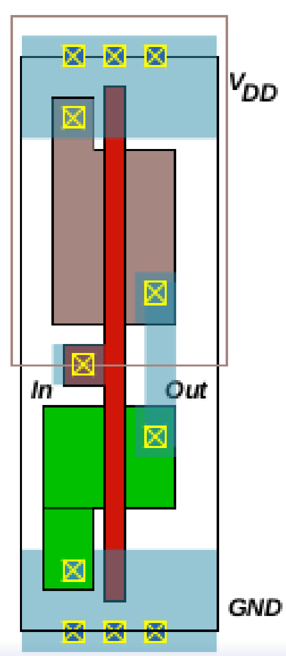
Standard Cells

With minimal diffusion routing





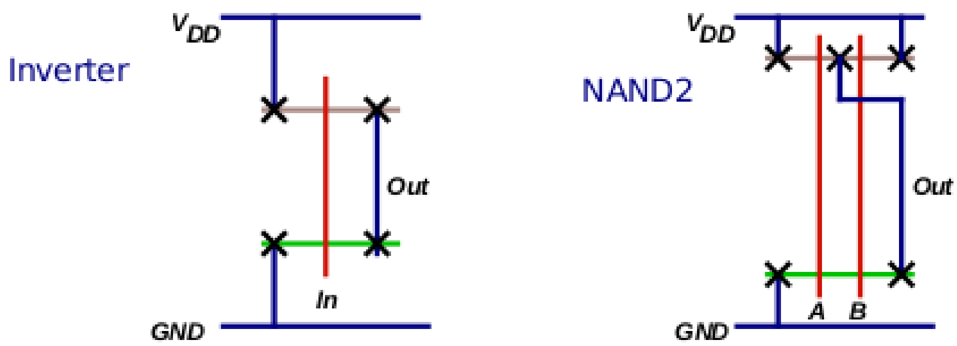
With silicided diffusion



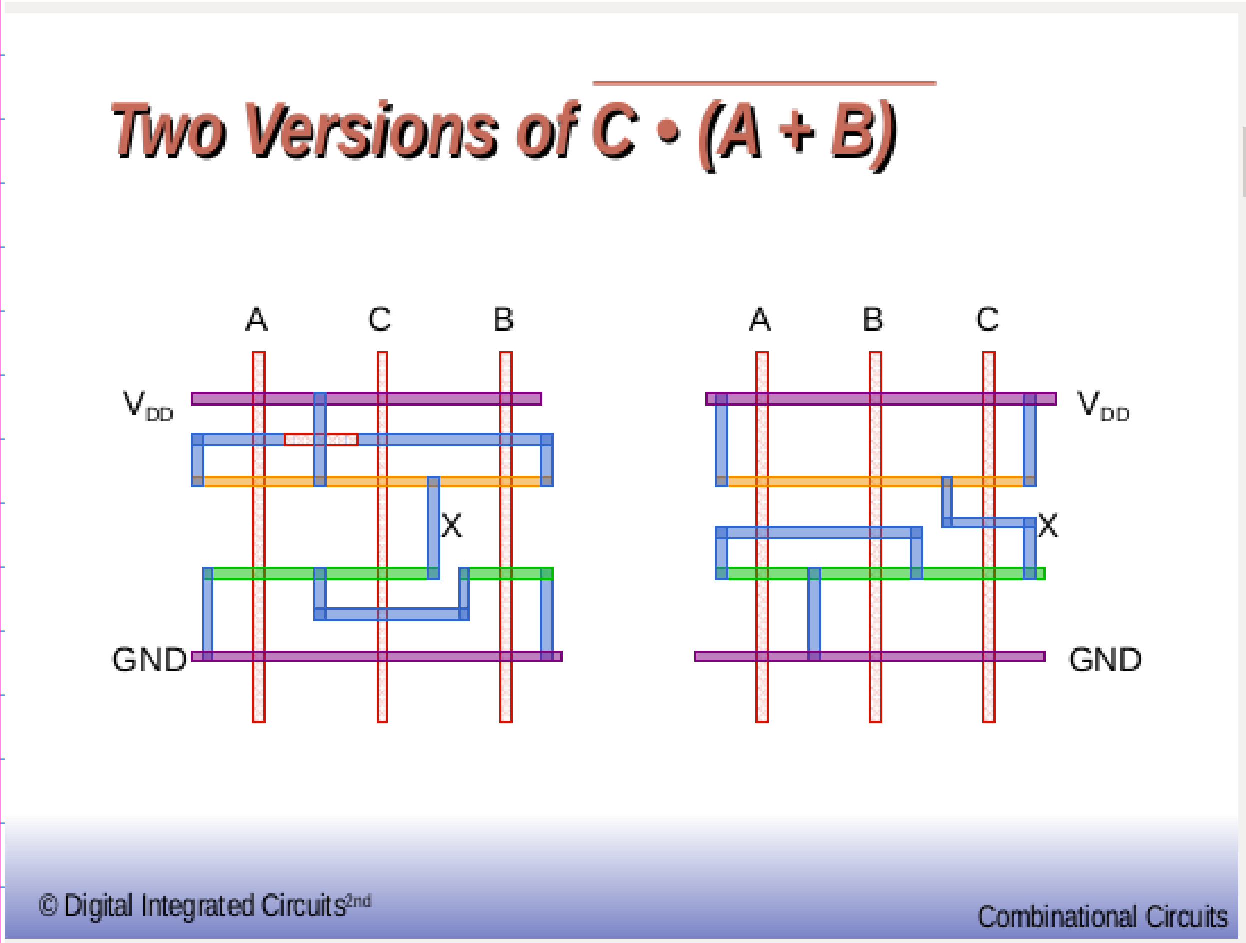
(http://bwrcs.eecs.berkeley.edu/Classes/IcBook/)

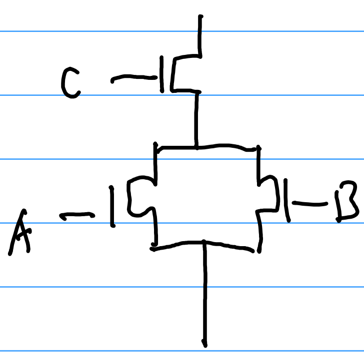
Stick Diagrams

Contains no dimensions
Represents relative positions of transistors



- (http://bwrcs.eecs.berkeley.edu/Classes/IcBook/)





Behavioral Decoder

```
`timescale 1ns/100ps

module decoder(A, D);
  input [2:0] A;
  output [7:0] D;
  reg    [7:0] D;

   always @(A)
     begin
       case (A)
          0: D = 8'b00000001;
          1: D = 8'b00000010;
          2: D = 8'b00000100;
          3: D = 8'b00001000;
          4: D = 8'b00010000;
          5: D = 8'b00100000;
          6: D = 8'b01000000;
          7: D = 8'b10000000;
          default: D = 8b'X;
      endcase
    end

endmodule
```

behavioral

Sythesis

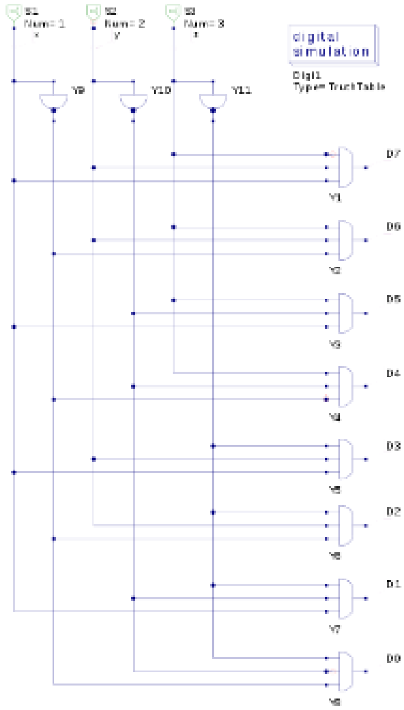
gate level design

decoder.v

```
`timescale 1ns/100ps

module decoder();
  wire  D0, D1, D2, D3, D4, D5, D6, D7;
  reg x, y, z;


  not (xb, x);
  not (yb, y);
  not (zb, z);

  and (D0, xb, yb, zb);  // D0 = minterm m0
  and (D1, xb, yb, z );  // D1 = minterm m1
  and (D2, xb, y , zb);  // D2 = minterm m2
  and (D3, xb, y , z );  // D3 = minterm m3
  and (D4, x , yb, zb);  // D4 = minterm m4
  and (D5, x , yb, z );  // D5 = minterm m5
  and (D6, x , y , zb);  // D6 = minterm m6
  and (D7, x , y , z );  // D7 = minterm m7
```

```
// Testbench Code goes here
initial begin
  $dumpfile("decoder.vcd");
  $dumpvars(0, decoder);

  $monitor ("[x y z] = %b%b%b [D0, D1, D2, D3]
                 = %b%b%b%b%b%b%b%b",
                x, y, z, D0, D1, D2, D3, D4, D5, D6, D7);

  x = 0;
  y = 0;
  z = 0;


  #8 $finish;
end


always #4 x = ~x;
always #2 y = ~y;
always #1 z = ~z;


endmodule
```

Decoder Schematic



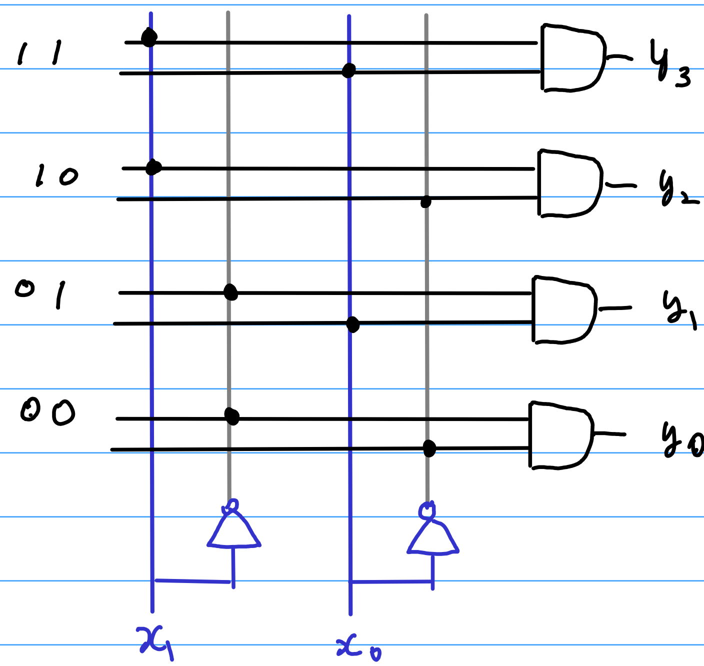
Synthesis

```
module decoder(A, D);
  input [2:0] A;
  output [7:0] D;
  reg      [7:0] D;

   always @(A)
     begin
        case (A)
            0: D = 8'b00000001;
            1: D = 8'b00000010;
            2: D = 8'b00000100;
            3: D = 8'b00001000;
            4: D = 8'b00010000;
            5: D = 8'b00100000;
            6: D = 8'b01000000;
            7: D = 8'b10000000;
            default: D = 8b'X;
         endcase
     end
```

⇒

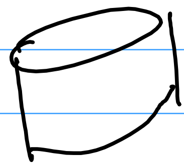
```
module decoder();
  wire  D0, D1, D2, D3, D4, D5, D6, D7;
  reg x, y, z;


  not (xb, x);
  not (yb, y);
  not (zb, z);

  and (D0, xb, yb, zb);  // D0 = minterm m0
  and (D1, xb, yb, z );  // D1 = minterm m1
  and (D2, xb, y , zb);  // D2 = minterm m2
  and (D3, xb, y , z );  // D3 = minterm m3
  and (D4, x , yb, zb);  // D4 = minterm m4
  and (D5, x , yb, z );  // D5 = minterm m5
  and (D6, x , y , zb);  // D6 = minterm m6
  and (D7, x , y , z );  // D7 = minterm m7
```

INV,
NAND

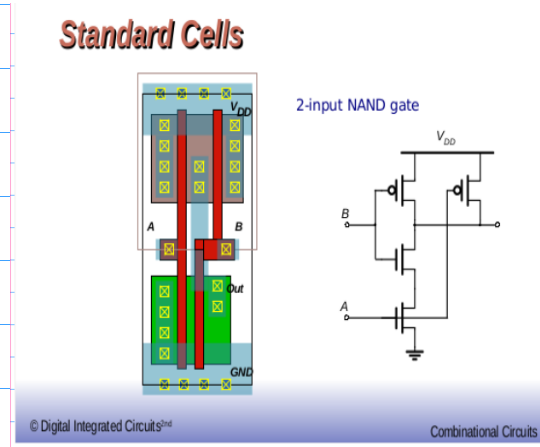


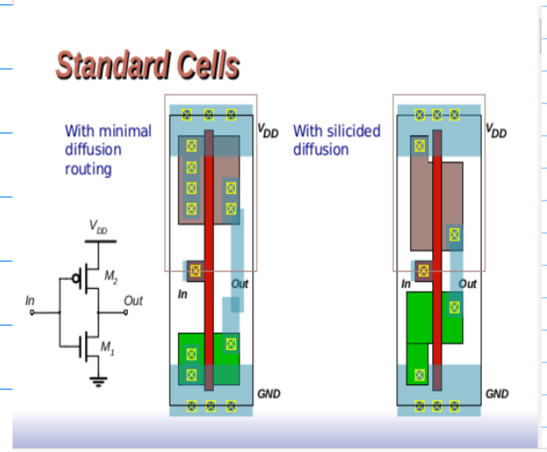
Standard cell Library

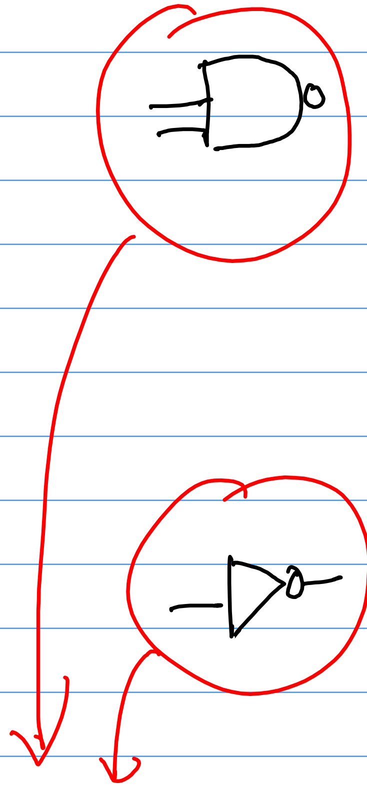
predefined
layout
collection

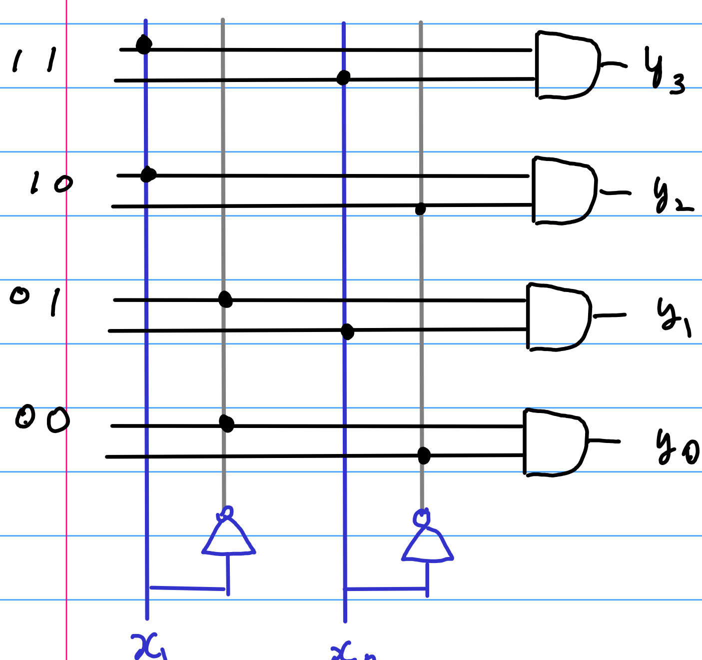


db









Placement

https://github.com/clothbot/Alliance-VLSI-CAD-System

Équipe Achitecture des Systèmes et Micro-Électronique
Laboratoire d'Informatique de Paris 6
Université Pierre et Marie Curie
4, Place Jussieu 75252 Paris Cedex 05,
France
http://www-asim.lip6.fr/alliance/
ftp://ftp-asim.lip6.fr/pub/alliance/
mailto:alliance-users@asim.lip6.fr

The figure 10 summarizes the followed process:

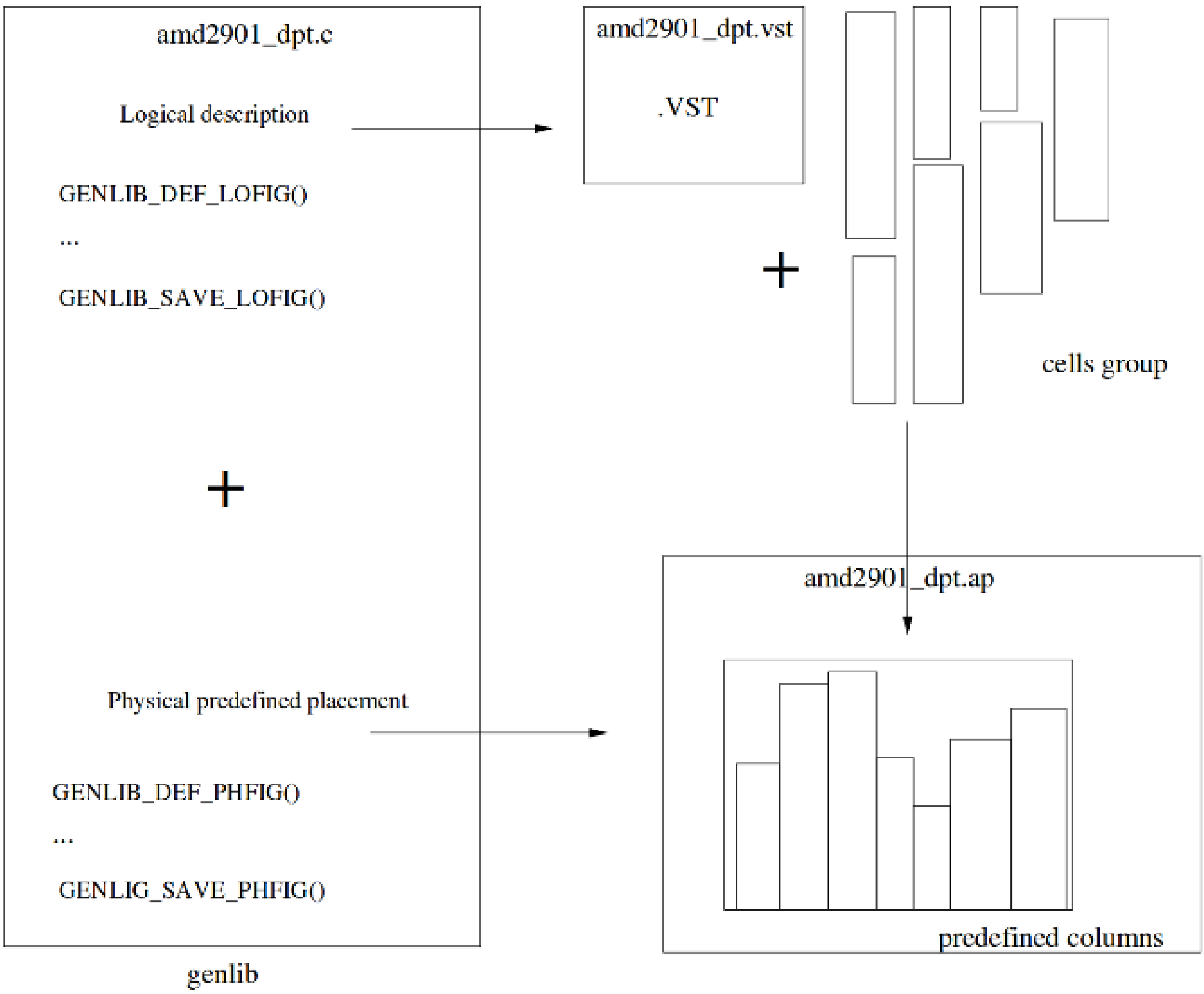


Figure 10: predefined placement

Placement

https://github.com/clothbot/Alliance-VLSI-CAD-System

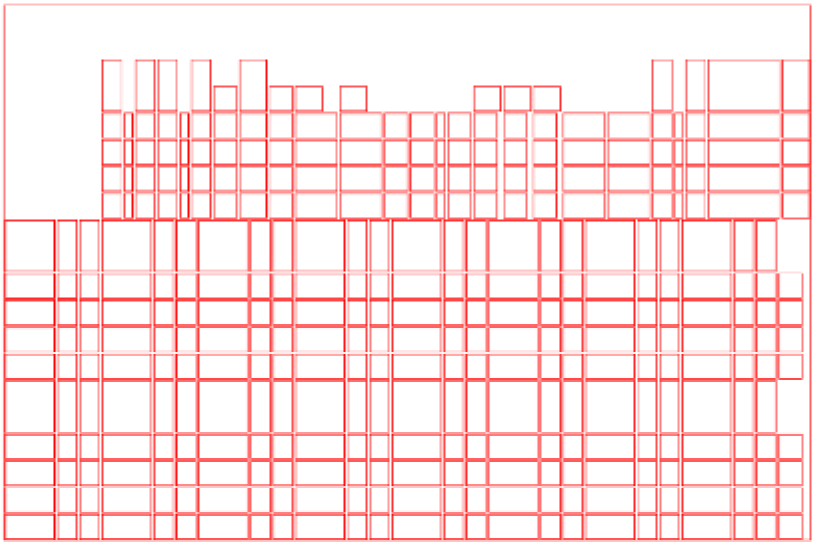


Figure 11: predefined Columns before placement of the part controls

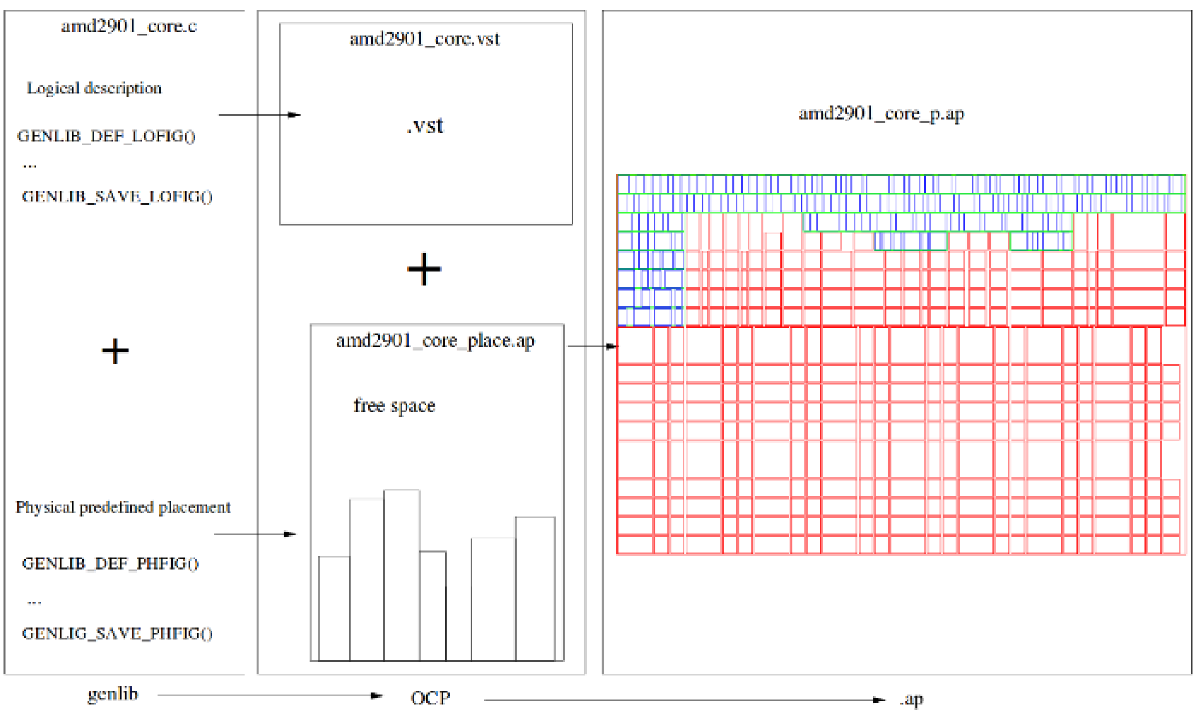
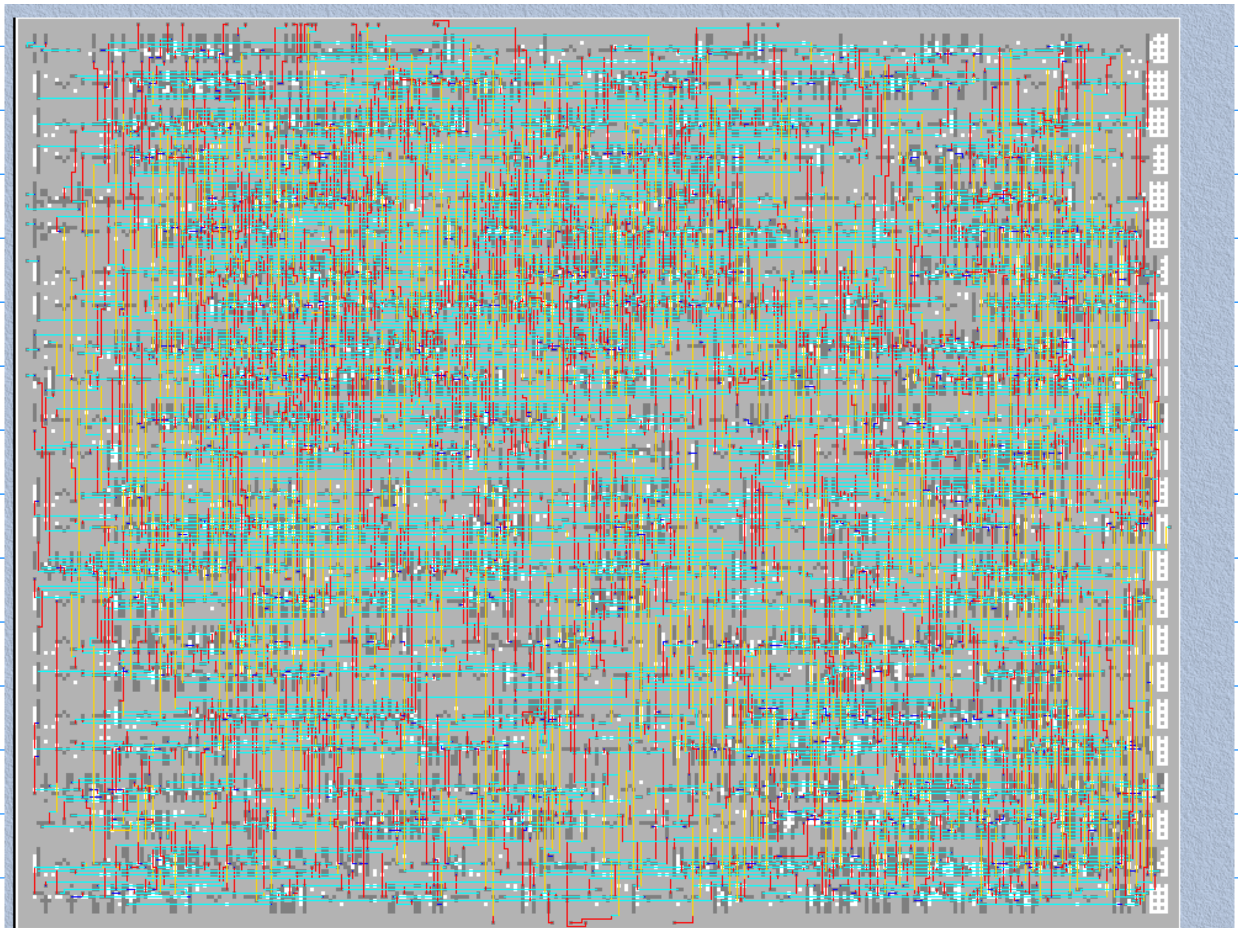


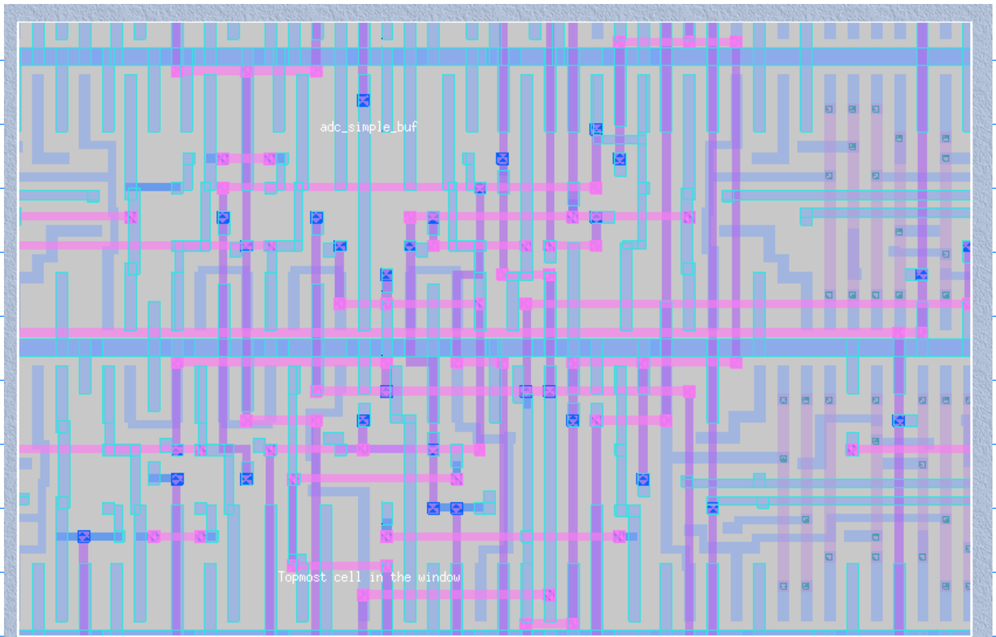
Figure 12: Placement

Route ..... connection

http://opencircuitdesign.com/qrouter/index.html

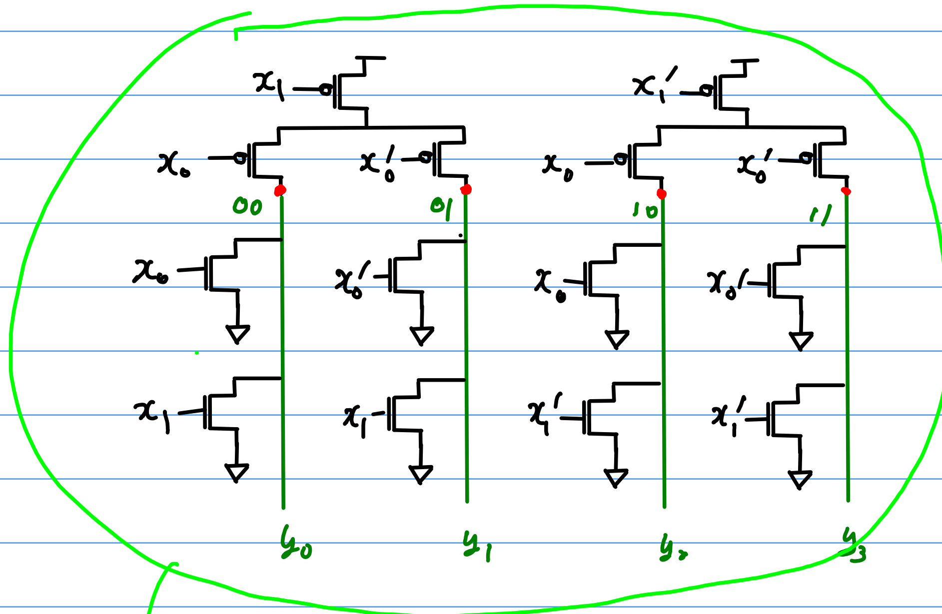


Graphic visualization of routing in qrouter 1.2. The design routed is i2c_master_top from Open Cores.



Detail of a layout generated by qrouter.

Custom Design → my own layout for my own CMOS design



draw a "Layout Design"