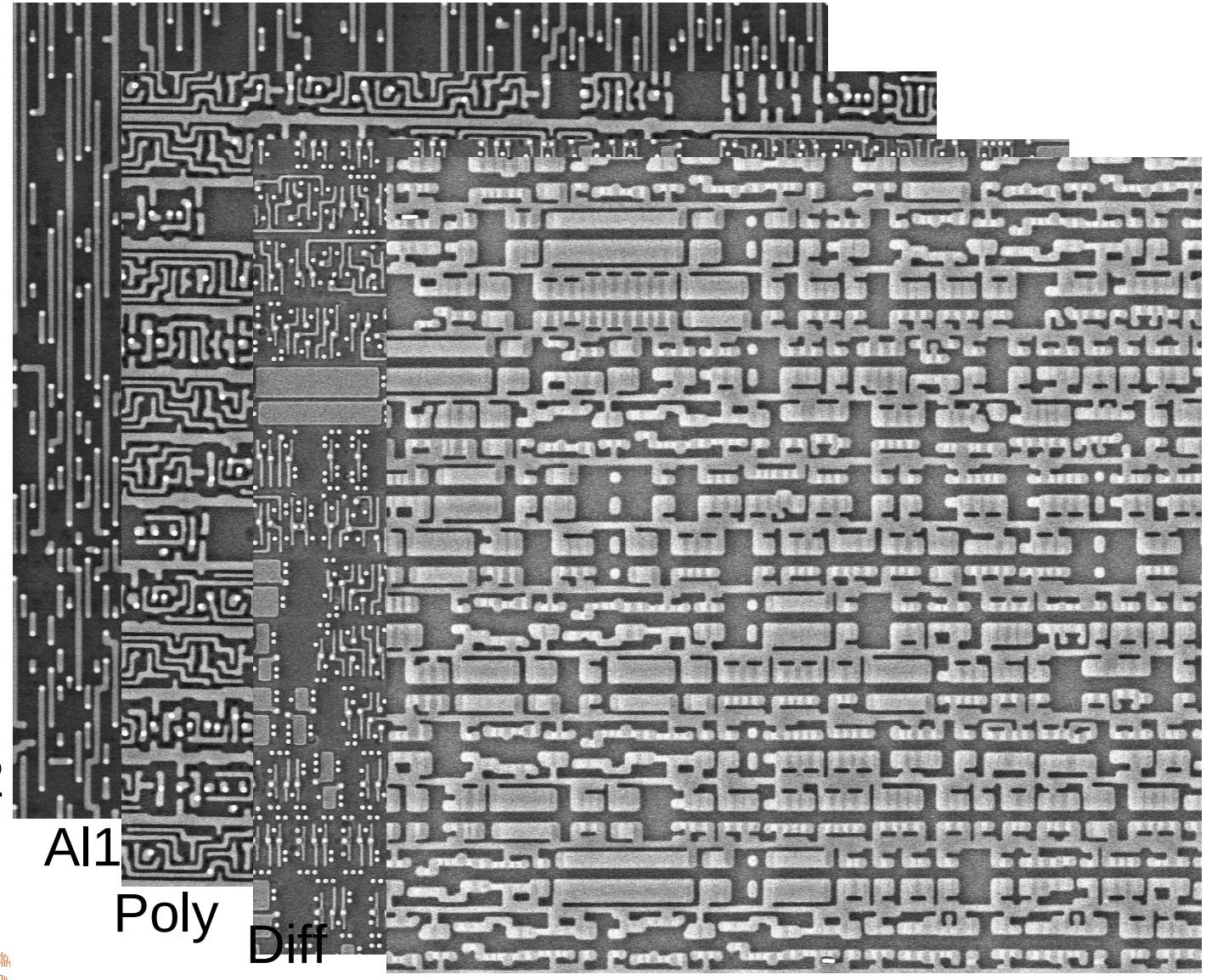
IC Reverse Engineering Netlist Extraction

Navid Asadi

Physical Inspection and AttacKs on ElectronicS (PHIKS)

Instituto de Microelectrónica de Barcelona

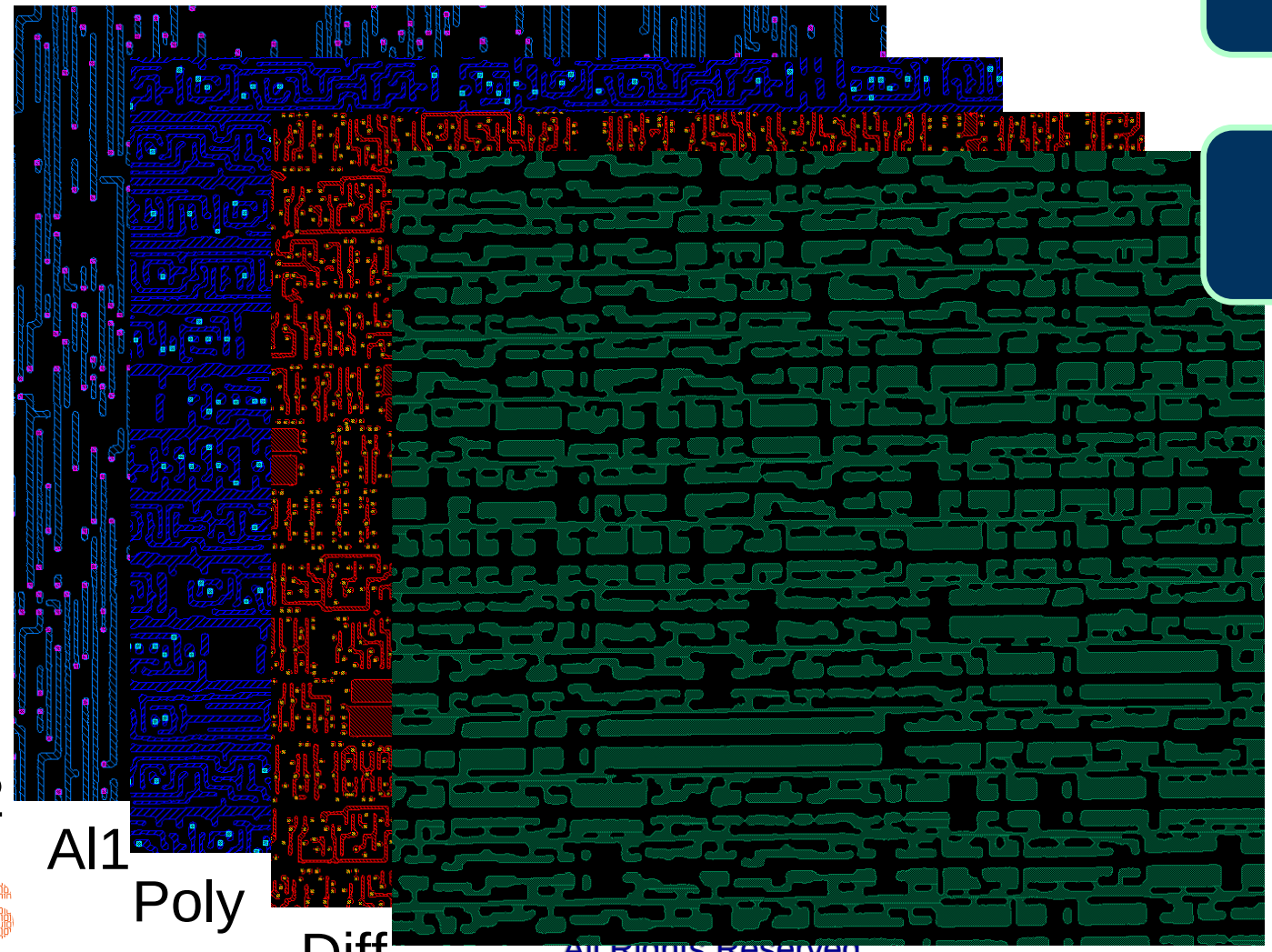
Physical Layer Overview



Polygon Extraction

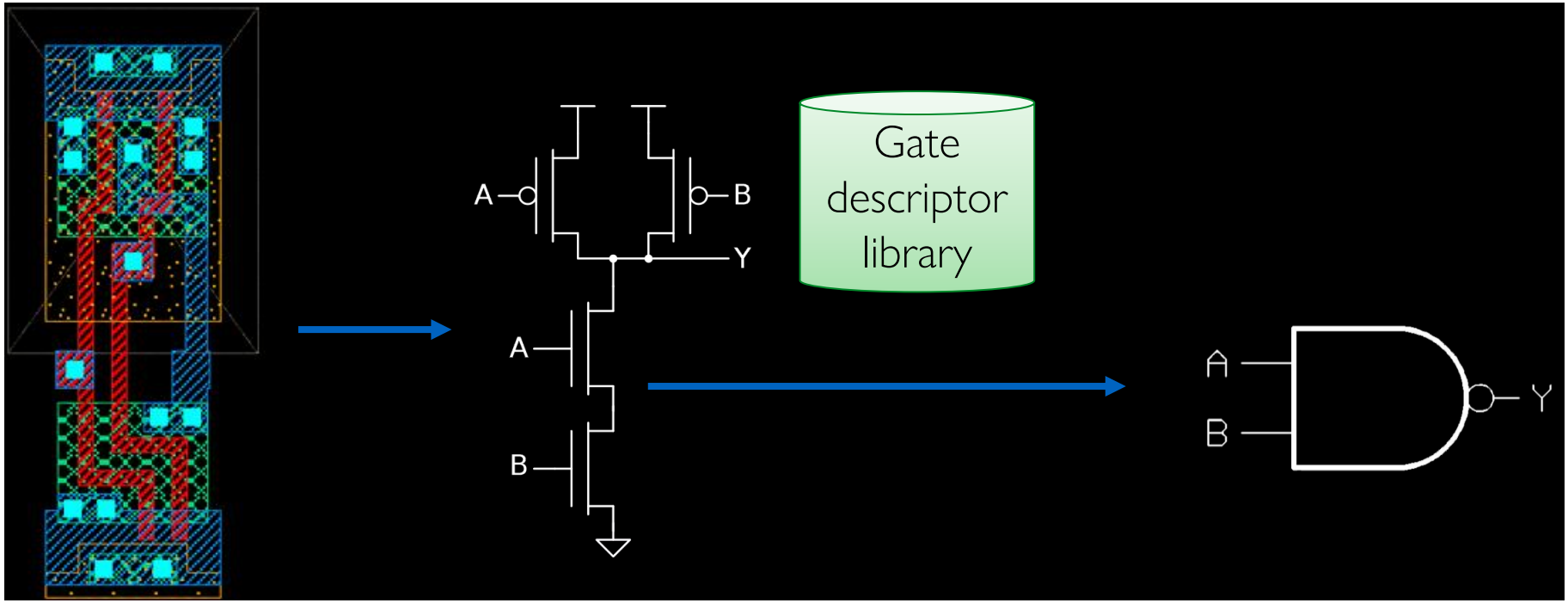
SEM Physical Imaging

Mask layout



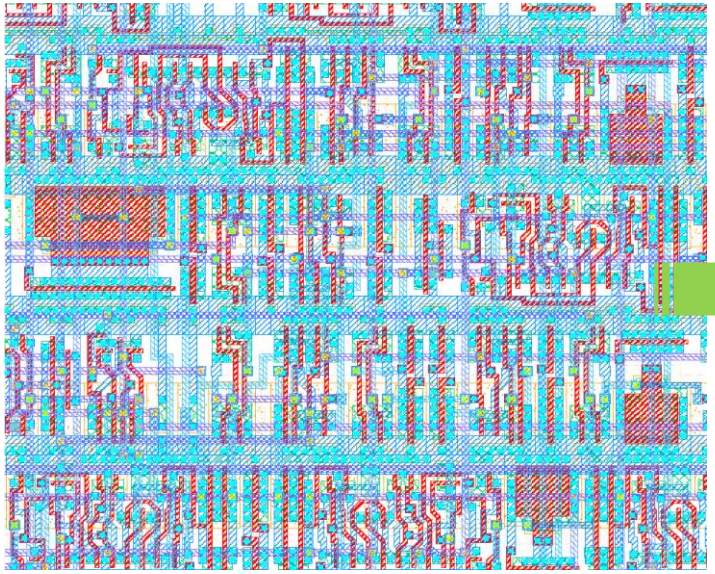
Logic Extraction

- Computation of a custom topological descriptor
  - Each gate topology has its unique descriptor
- Gate identification by fast comparison to an extensive gate topology descriptor database



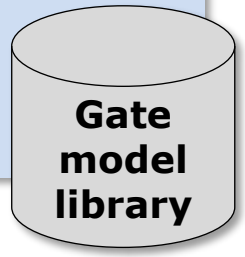
Logic Extraction

- Extraction of **logic netlist** from multi-layer vectorized layout:



e.g. 0.35µm **4-metal** CMOS technology

```
module myIC (pad_list…);
NOR3 I_341 (.A(3735), .C(3455), .Y(3787), .B(3640));
XOR2 I_340 (.A(172), .Y(194), .B(70));
OAI21 I_338 (.B1(2323), .A1(2315), .Y(2390), .A2(2198));
NAND2 I_334 (.A(3886), .Y(3817), .B(3820));
INV I_332 (.A(1851), .Y(1791));
AND2 I_329 (.A(89), .B(127), .Y(142));
DFFRes_up_preOutsPN I_322 (.D(194), .Yn(329), .Clk(165), .Q(127),
.RN(144), .Yp(329), .QN(172));
NAND2 I_328 (.A(142), .Y(248), .B(225));
OAI211 I_316 (.A2(560), .A1(571), .B1(585), .Y(478), .C1(594));
DFFRes_up_preOutsPN I_314 (.Q(962), .D(862), .QN(971), .RN(144),
.Yn(906), .Yp(906), .Clk(165));
NAND3 I_325 (.B(1433), .A(1948), .Y(1869), .C(1058));
XOR2 I_321 (.A(424), .Y(440), .B(459));
INV I_278 (.A(248), .Y(239));
INV I_277 (.A(153), .Y(121));
AND2 I_276 (.A(3347), .B(3104), .Y(3766));
NAND3 I_326 (.B(1948), .A(1433), .Y(1901), .C(1961));
…
endmodule
```



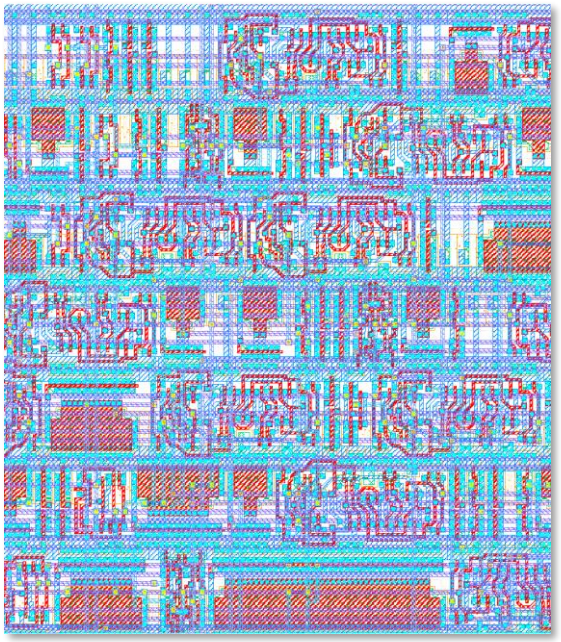
symbol & Verilog views

- Gate circuit extraction
- Gate identification based on topology
- Analysis of new topologies
- Annotation of routing
- Create full-IC logic netlist

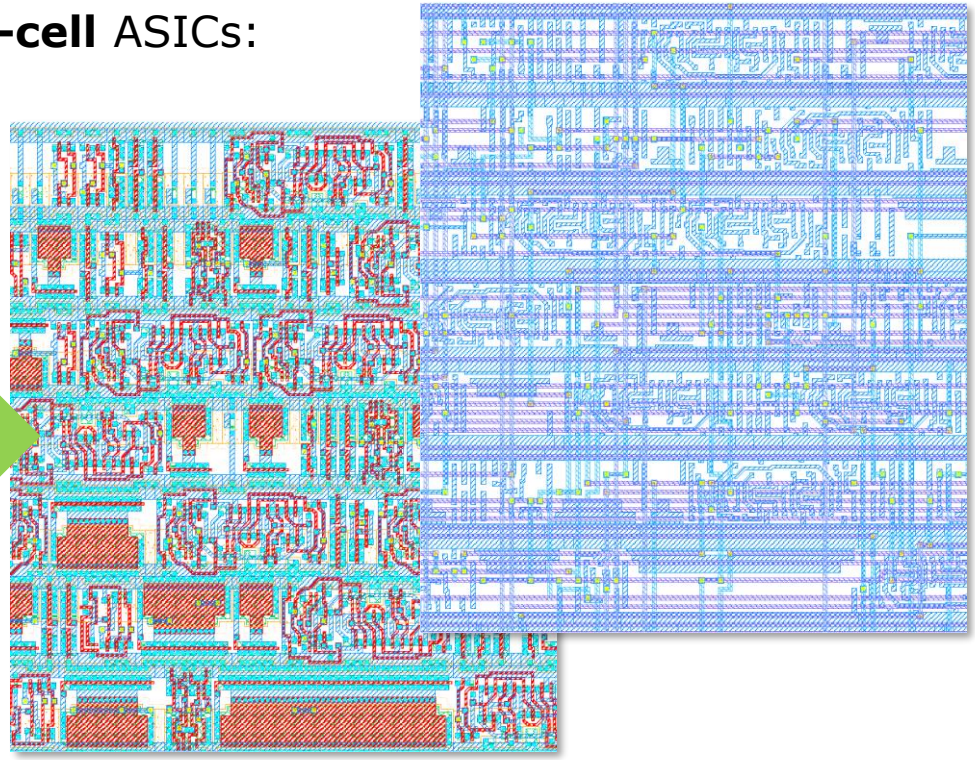
Gate Circuit Extraction

- Splitting between **gates** and **routing**
- Based on **layers** for **standard-cell** ASICs:

routing = met1 + via1 + met2 + via2 + met4 + via3 + met4



e.g. 0.35µm **4-metal** CMOS technology



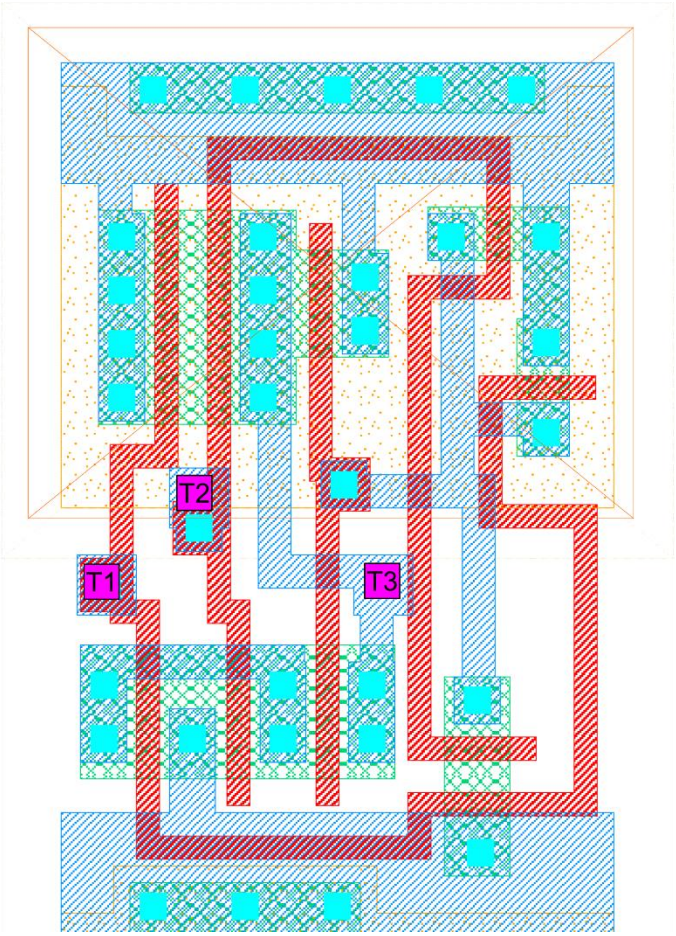
gates = well + diffs + poly + contact + met1 (via1 → **I/O pins**)

Gate Circuit Extraction

- Each **gate boundary** defined by its **isolated network** of **transistors**:

Cadence Diva/Assura extraction script

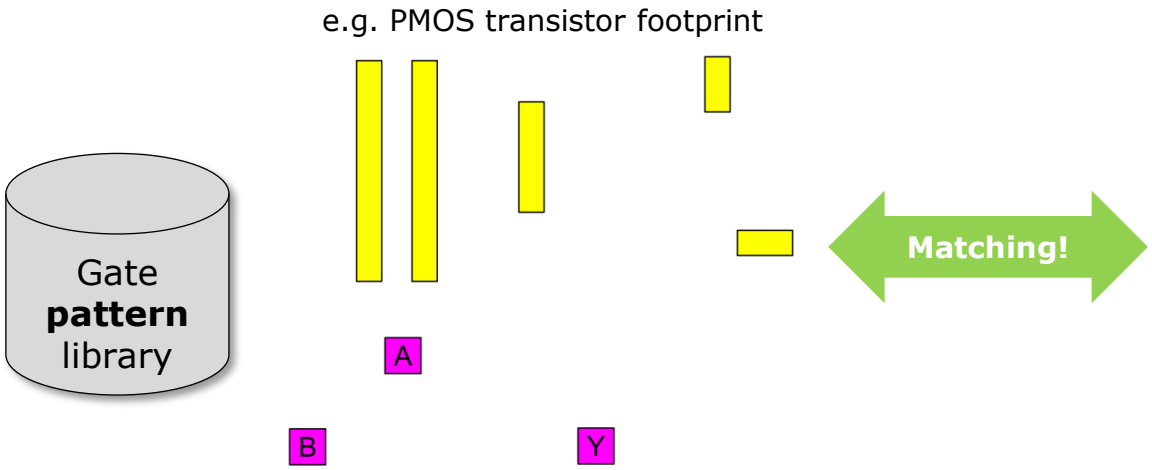
```
drcExtractRules (
    (nwell = geomOr("NTUB"))
    (gasad = geomOr("GASAD"))
    (nplus = geomOr("NPLUS"))
    (gatepoly = geomOr("POLY1"))
    (cont = geomOr("WINDOW"))
    (met1 = geomOr("METAL1"))
    ...
    (ndiff = geomAnd(gasad nplus))
    (pdiff = geomAndNot(gasad ndiff))
    (nmos = geomAndNot(geomAnd(poly ndiff) nwell))
    (pmos = geomAnd(geomAnd(poly pdiff) nwell))
    ...
    geomConnect(
        (via cont met1 nwell ndiff pdiff gatepoly)
        (via via1 met1 met2)
        (via via2 met2 met3)  ...)
    extractMOS(nmos (gatepoly "G") (ndiff "S" "D") "NBSIM")
    extractMOS(pmos (gatepoly "G") (pdiff "S" "D") "PBSIM")
    ...
    saveInterconnect(
        (nwell "NTUB")
        (ndiff "N_DIF")
        (pdiff "P_DIF")
        (cont "WINDOW")  ...))
```

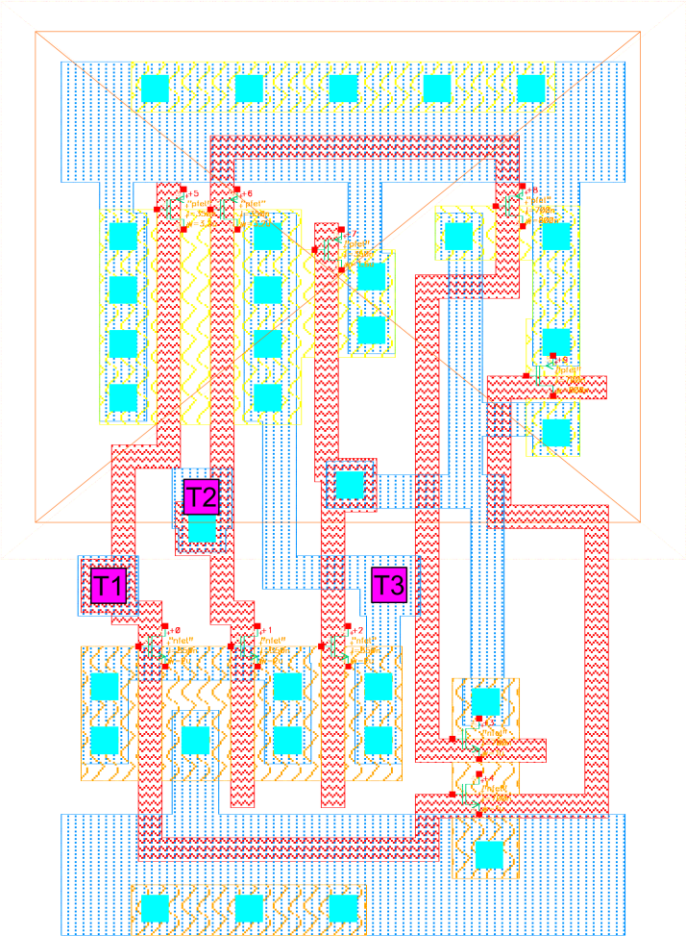


layout view

Gate Identification

- Several **matching** rules are possible:
  - Mask **pattern**?





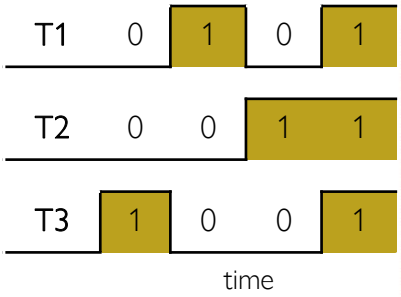
extracted view

- **Technology** strong dependence
- Sensitivity to **segmentation** accuracy
- **CPU-time** consuming

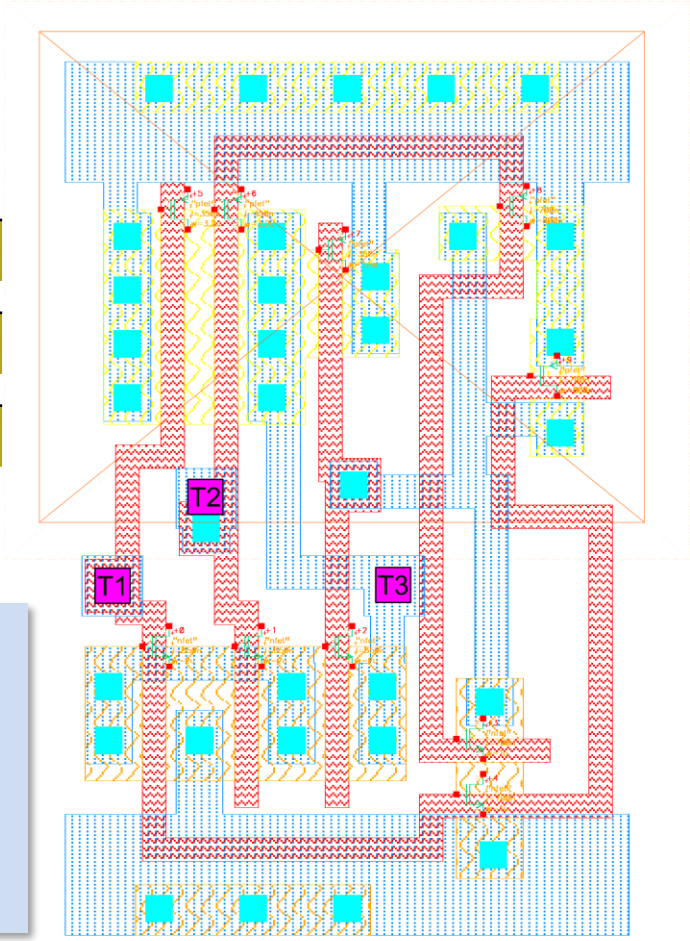
Gate Identification

- Several **matching** rules are possible:
  - Mask **pattern**?
  - **Electrical** simulation?

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1




```
.subckt I234 T1 T2 T3
M1 3 T1 VDD! VDD! PBSIM L=700n W=800n M=1
M2 VDD! T2 3 VDD! PBSIM L=700n W=800n M=1
M3 VDD! 3 T3 VDD! PBSIM L=350n W=1.6u M=1
M4 T3 T2 5 VDD! PBSIM L=350n W=3.2u M=1
M5 5 T1 VDD! VDD! PBSIM L=350n W=3.2u M=1
M6 4 T2 3 0 NBSIM L=700n L=1u M=1
M7 0 T1 4 0 NBSIM L=700n L=1u M=1
M8 T3 3 2 0 NBSIM L=350n W=2u M=1
M9 2 T2 0 0 NBSIM L=350n W=2u M=1
M10 0 T1 2 0 NBSIM L=350n W=2u M=1
.ends
```

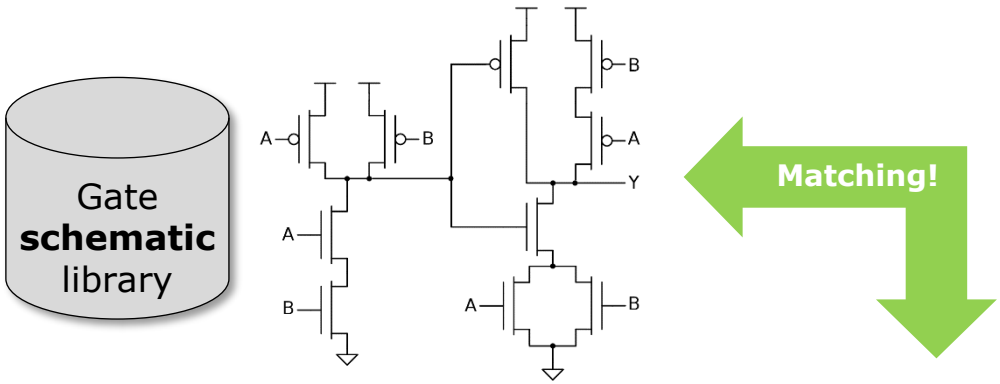


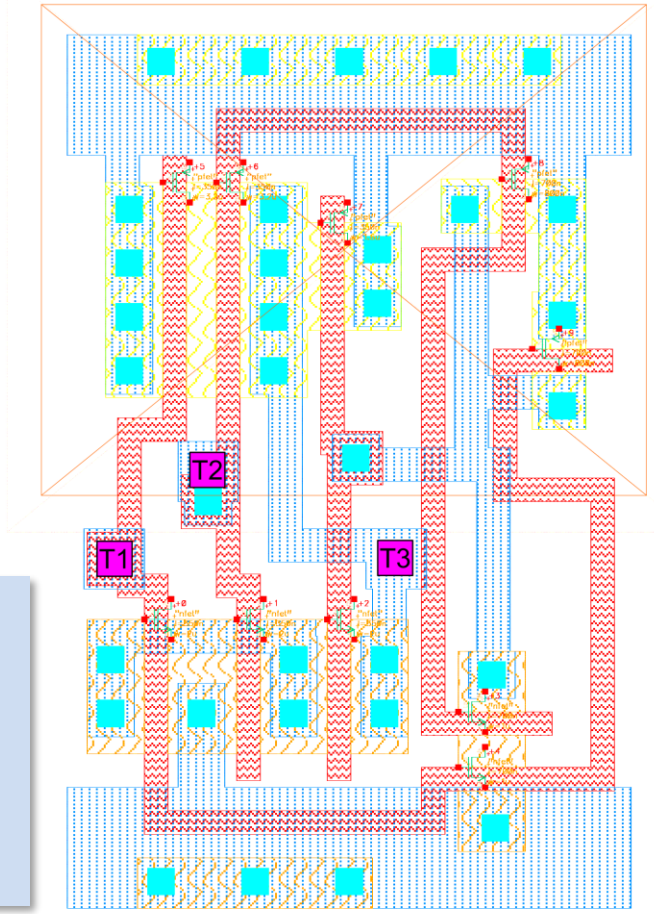
- ▲ Technology **independent**
- ▼ **I/O** term identification
- ▼ **CPU-time** consuming

Gate Identification

- Several **matching** rules are possible:
  - Mask **pattern**?
  - **Electrical** simulation?
  - Layout versus schematic **topology** (**LVS**)?

Geometric properties and spatial relations unaffected by the change of shape or size of figures



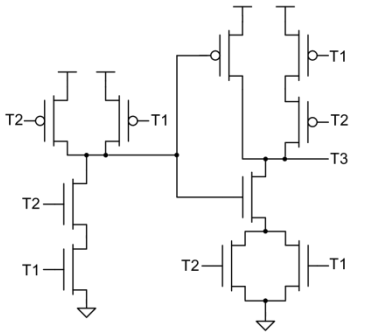



```
.subckt I234 T1 T2 T3
M1 3 T1 VDD! VDD! PBSIM L=700n W=800n M=1
M2 VDD! T2 3 VDD! PBSIM L=700n W=800n M=1
M3 VDD! 3 T3 VDD! PBSIM L=350n W=1.6u M=1
M4 T3 T2 5 VDD! PBSIM L=350n W=3.2u M=1
M5 5 T1 VDD! VDD! PBSIM L=350n W=3.2u M=1
M6 4 T2 3 0 NBSIM L=700n L=1u M=1
M7 0 T1 4 0 NBSIM L=700n L=1u M=1
M8 T3 3 2 0 NBSIM L=350n W=2u M=1
M9 2 T2 0 0 NBSIM L=350n W=2u M=1
M10 0 T1 2 0 NBSIM L=350n W=2u M=1
.ends
```

- ▲ Technology **independent**
- ▼ **CPU-time** consuming

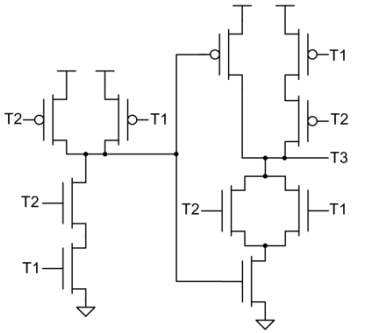
Gate Identification

- **Descriptor** vs **model** libraries:



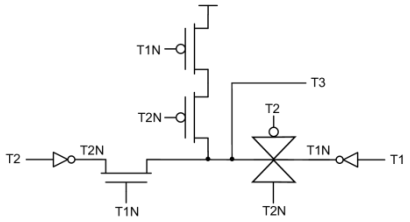
Matching!

"10 7":(("**XNOR2_TYPE1**" (1 2 1 3 0 1) (2 2 2 0 4 1) (1 0 3 3 0 0) (1 3 2 3 2 0) (1 0 2 2 0 0) (1 2 0 2 0 0)) (("**Y**" ((2 1 21) (1 1 10))) ("**B**" ((2 2 45) (1 0 0))) ("**A**" ((2 1 22) (1 1 10)))) nil)



Matching!

"10 7":(("**XNOR2_TYPE2**" (1 2 0 2 0 0) (1 0 3 3 0 0) (1 3 2 3 2 0) (1 2 2 4 0 1) (2 2 2 0 4 1) (1 0 2 2 0 0)) (("**Y**" ((2 0 0) (1 2 12))) ("**B**" ((2 1 34) (1 1 1))) ("**A**" ((2 1 23) (1 1 11)))) nil)



Matching!

"9 6":(("**XNOR2_TYPE3**" (1 2 1 0 3 1) (1 1 1 0 2 1) (1 2 2 4 0 1) (1 2 3 3 2 0) (1 3 3 4 2 0) (1 2 0 2 0 0)) (("**A**" ((2 1 11) (1 1 10))) ("**B**" ((2 2 22) (1 0 0))) ("**Y**" ((2 1 11) (1 1 10)))) (("**Y**" (1 2 2 4 0) (1 2 2 4 0)) ("**B**" (1 1 1 0 2) (1 1 1 0 2)) ("**A**" (1 2 1 0 3) (1 2 1 0 3))))

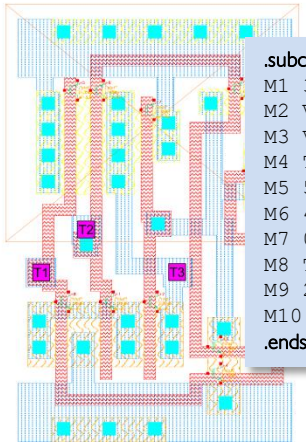
Mapping!

[@instanceName]


```
module XNOR2(Y,A,B);
  output Y;
  input A,B;

  specify
    specparam tphl$A$Y = 1:1:1;
    specparam tplh$A$Y = 1:1:1;
    specparam tphl$B$Y = 1:1:1;
    specparam tplh$B$Y = 1:1:1;
    (A => Y) = (tplh$A$Y,tphl$A$Y);
    (B => Y) = (tplh$B$Y,tphl$B$Y);
  endspecify

  xnor(Y,A,B);

endmodule
```

Gate Identification

- Descriptor-based **procedure**:

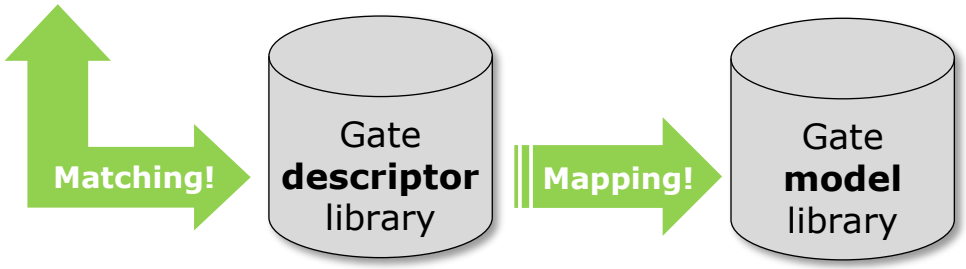



```
.subckt  I234  T1 T2 T3
M1  3  T1  VDD!  VDD!  PBSIM  L=700n  W=800n  M=1
M2  VDD!  T2  3  VDD!  PBSIM  L=700n  W=800n  M=1
M3  VDD!  3  T3  VDD!  PBSIM  L=350n  W=1.6u  M=1
M4  T3  T2  5  VDD!  PBSIM  L=350n  W=3.2u  M=1
M5  5  T1  VDD!  VDD!  PBSIM  L=350n  W=3.2u  M=1
M6  4  T2  3  0  NBSIM  L=700n  L=1u  M=1
M7  0  T1  4  0  NBSIM  L=700n  L=1u  M=1
M8  T3  3  2  0  NBSIM  L=350n  W=2u  M=1
M9  2  T2  0  0  NBSIM  L=350n  W=2u  M=1
M10  0  T1  2  0  NBSIM  L=350n  W=2u  M=1
.ends
```

"10 7":(("**I234**" (1 2 1 3 0 1) (2 2 2 0 4 1) (1 0 3 3 0 0)(1 3 2 3 2 0) (1 0 2 2 0 0) (1 2 0 2 0 0)) (("**T3**" ((2 1 21) (1 1 10))) ("**T1**" ((2 2 45) (1 0 0))) ("**T2**" ((2 1 22) (1 1 10)))) nil)

Cadence Skill script

```
procedure(createCellDescriptor(cellName library2import cellNameInLib view)
  ...
  prog((netDescriptorList netDescriptorList2evaluate)
    rep=dbOpenCellViewByType(library2import cellNameInLib view "" "a")

    /*Remove Vdd and Gnd*/
    dbDeleteObject(car(setof(net rep~>nets net~>name == nameVdd)))
    dbDeleteObject(car(setof(net rep~>nets net~>name == nameGnd)))

    /*Remove multiplicity*/
    removeMultiplicity(transNname transPname setof(inst rep~>instances
      inst~>cellName == transNname || inst~>cellName == transPname))

    /*Generation of net descriptors*/
    foreach(netCell rep~>nets
      createPNlists(netCell~>allInstTerms~>inst)
      createTermNamelists(netCell~>allInstTerms)
      isTerm=0
      when(netCell~>term!=nil isTerm=1);when
      netDescriptor=list(1 length(Plist) length(Nlist) length(SDlist) ...)
      netDescriptorList2evaluate=cons(netDescriptor netDescriptorList2...)
    );foreach
  ...
  );prog
);createCellDescriptor
```



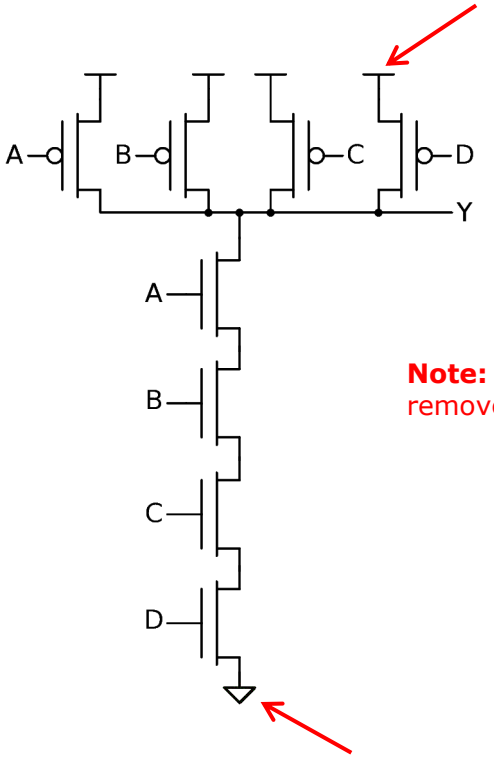
symbol & Verilog views

- **Unique** descriptor per topology
- **Faster** processing than LVS

Gate Identification

- **Topology**-based gate **descriptor** proposal:

1. Total number of **transistors** and **nets**



Note: supply nets are removed from now on!

number of transistors
number of nets

"8 8":(("**NAND4**")

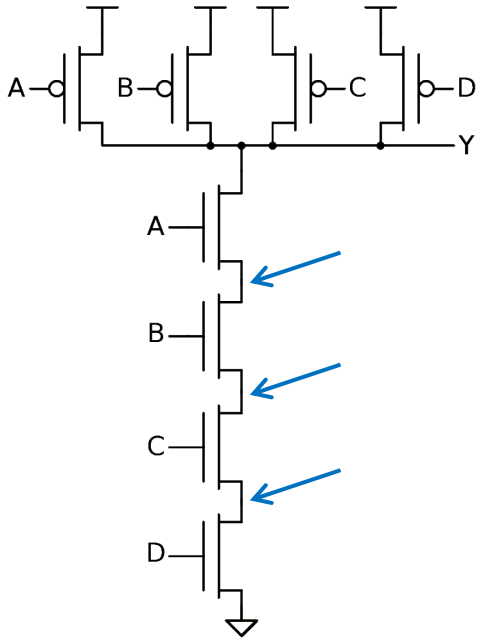
topology id

))

Gate Identification

- **Topology**-based gate **descriptor** proposal:

1. Total number of **transistors** and **nets**
2. Net classification: Number of **nets** with the same:
   a. Number of **PMOS** transistors
   b. Number of **NMOS** transistors
   c. Number of transistor **D/S-terms**
   d. Number of transistor **G-terms**
   e. Is gate **I/O term**?



3 nets with:
none PMOS
2 NMOS
not gate I/O

"8 8":(("**NAND4**" (3 0 2 2 0 0))

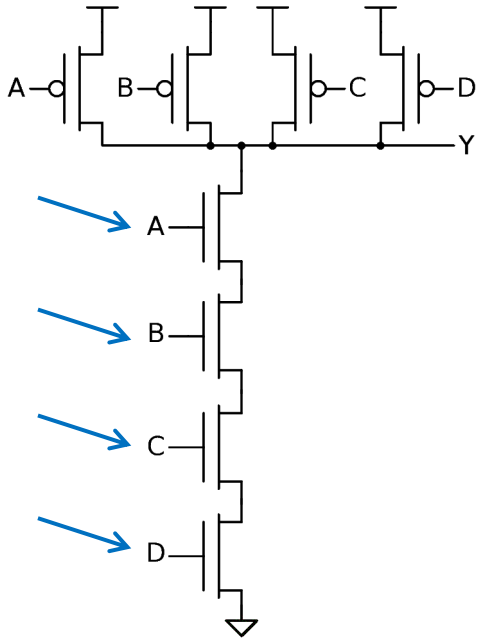
none G term
2 D/S terms

))

Gate Identification

- **Topology**-based gate **descriptor** proposal:

1. Total number of **transistors** and **nets**
2. Net classification: Number of **nets** with the same:
   a. Number of **PMOS** transistors
   b. Number of **NMOS** transistors
   c. Number of transistor **D/S-terms**
   d. Number of transistor **G-terms**
   e. Is gate **I/O term**?



4 nets with:
1 PMOS
1 NMOS
yes gate I/O

"8 8":(("**NAND4**" (3 0 2 2 0 0) (4 1 1 0 2 1))

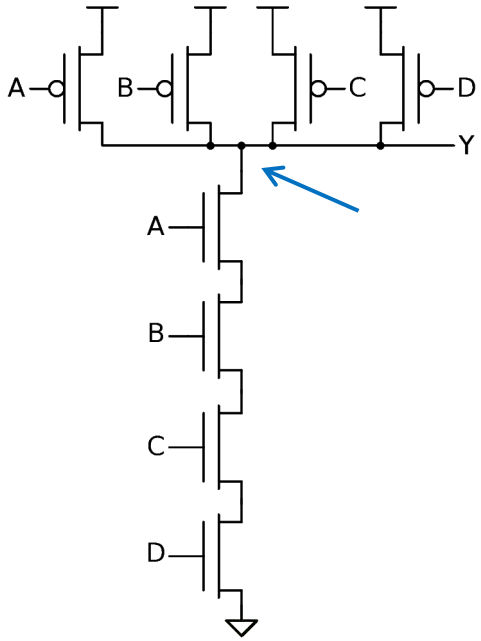
2 G terms
none D/S term

))

Gate Identification

- **Topology**-based gate **descriptor** proposal:

1. Total number of **transistors** and **nets**
2. Net classification: Number of **nets** with the same:
   a. Number of **PMOS** transistors
   b. Number of **NMOS** transistors
   c. Number of transistor **D/S-terms**
   d. Number of transistor **G-terms**
   e. Is gate **I/O term**?



1 nets with:
4 PMOS
1 NMOS
yes gate I/O

"8 8":(("**NAND4**" (3 0 2 2 0 0) (4 1 1 0 2 1) (1 4 1 5 0 1))

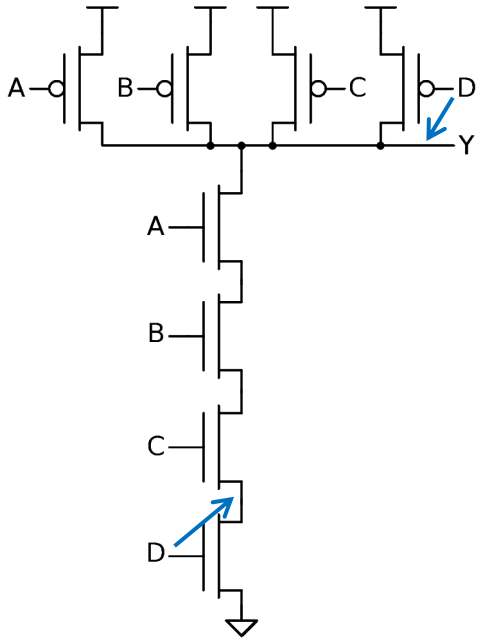
none G term
5 D/S terms

))

Gate Identification

- **Topology**-based gate **descriptor** proposal:

1. Total number of **transistors** and **nets**
2. Net classification: Number of **nets** with the same:
   a. Number of **PMOS** transistors
   b. Number of **NMOS** transistors
   c. Number of transistor **D/S-terms**
   d. Number of transistor **G-terms**
   e. Is gate **I/O term**?
3. For each I/O term, **shortest path** to reach the rest of I/O terms



net **classification**

I/O term **identification**

"8 8":(("**NAND4**" (3 0 2 2 0 0) (4 1 1 0 2 1) (1 4 1 5 0 1)) (("D" ((1 1 10)))

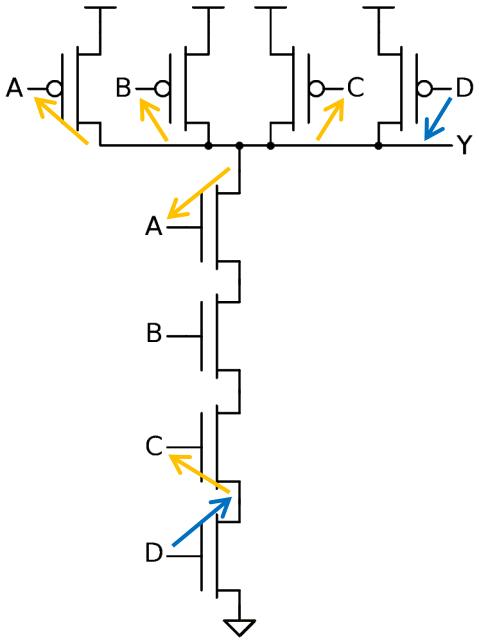
first iteration:
1 I/O term reached
through 1 PMOS
and none NMOS

))))

Gate Identification

- **Topology**-based gate **descriptor** proposal:

1. Total number of **transistors** and **nets**
2. Net classification: Number of **nets** with the same:
   a. Number of **PMOS** transistors
   b. Number of **NMOS** transistors
   c. Number of transistor **D/S-terms**
   d. Number of transistor **G-terms**
   e. Is gate **I/O term**?
3. For each I/O term, **shortest path** to reach the rest of I/O terms



net **classification**

I/O term **identification**

"8 8":(("**NAND4**" (3 0 2 2 0 0) (4 1 1 0 2 1) (1 4 1 5 0 1)) (("D" ((2 3 32) (1 1 10)))

second iteration:
3 I/O term reached
through 3 PMOS
and 2 NMOS

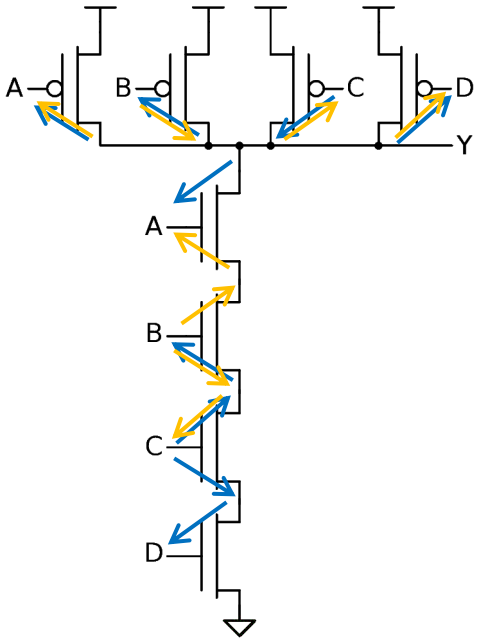
))

))

Gate Identification

- **Topology**-based gate **descriptor** proposal:

1. Total number of **transistors** and **nets**
2. Net classification: Number of **nets** with the same:
   a. Number of **PMOS** transistors
   b. Number of **NMOS** transistors
   c. Number of transistor **D/S-terms**
   d. Number of transistor **G-terms**
   e. Is gate **I/O term**?
3. For each I/O term, **shortest path** to reach the rest of I/O terms



net **classification**

I/O term **identification**

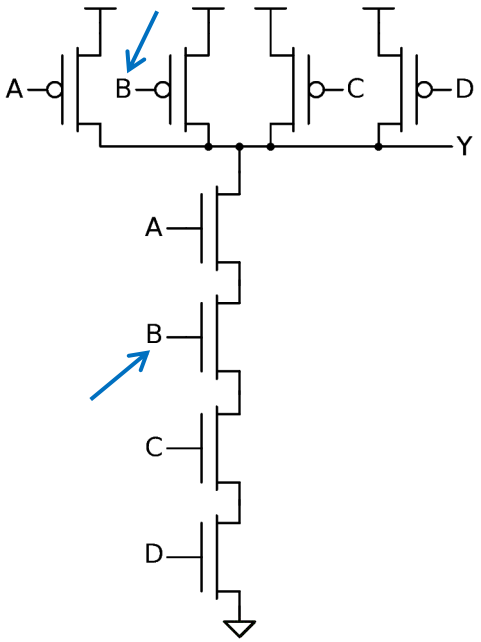
"8 8":(("**NAND4**" (3 0 2 2 0 0) (4 1 1 0 2 1) (1 4 1 5 0 1)) (("D" ((2 3 32) (1 1 10))) ("C" ((2 3 33) (1 1 10)))
("A" ((3 0 0) (2 3 31) (1 1 11))) ("B" ((2 3 33) (1 1 10))) ("Y" ((2 0 0) (1 4 41))))

))

Gate Identification

- **Topology**-based gate **descriptor** proposal:

1. Total number of **transistors** and **nets**
2. Net classification: Number of **nets** with the same:
   a. Number of **PMOS** transistors
   b. Number of **NMOS** transistors
   c. Number of transistor **D/S-terms**
   d. Number of transistor **G-terms**
   e. Is gate **I/O term**?
3. For each I/O term, **shortest path** to reach the rest of I/O terms
4. If identical I/O terms in 3, for each shortest path, extend **transistor connectivity** scanning



net **classification**

I/O term **identification**

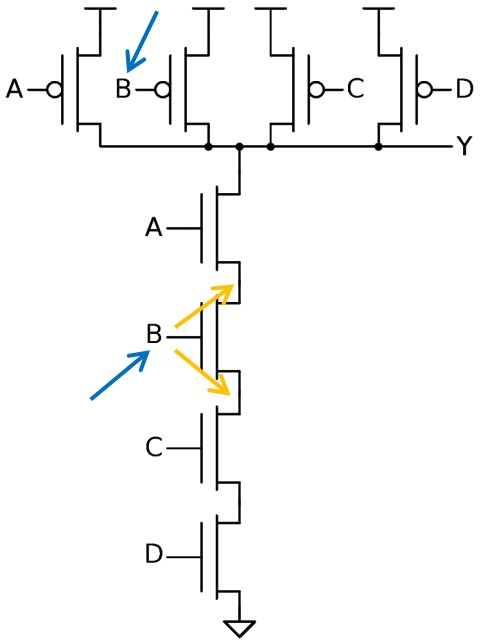
"8 8":(("**NAND4**" (3 0 2 2 0 0) (4 1 1 0 2 1) (1 4 1 5 0 1)) (("D" ((2 3 32) (1 1 10))) ("C" ((2 3 33) (1 1 10))) ("A" ((3 0 0) (2 3 31) (1 1 11))) ("B" ((2 3 33) (1 1 10))) ("Y" ((2 0 0) (1 4 41)))) (("D" (1 1 1 0 2) (1 1 1 0 2)) ("C" (1 1 1 0 2) (1 1 1 0 2)) ("A" (1 1 1 0 2) (1 1 1 0 2)) ("B" (1 1 1 0 2) (1 1 1 0 2)) ("Y" (1 4 1 5 0) (1 4 1 5 0))))

PMOS path NMOS path 1st iteration: 1 PMOS, 1 NMOS, 0 D/S, 2 G

Gate Identification

- **Topology**-based gate **descriptor** proposal:

1. Total number of **transistors** and **nets**
2. Net classification: Number of **nets** with the same:
   a. Number of **PMOS** transistors
   b. Number of **NMOS** transistors
   c. Number of transistor **D/S-terms**
   d. Number of transistor **G-terms**
   e. Is gate **I/O term**?
3. For each I/O term, **shortest path** to reach the rest of I/O terms
4. If identical I/O terms in 3, for each shortest path, extend **transistor connectivity** scanning



net **classification**

I/O term **identification**

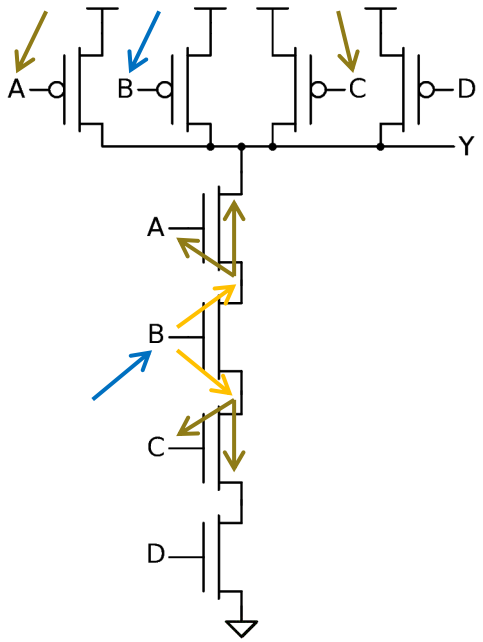
"8 8":(("**NAND4**" (3 0 2 2 0 0) (4 1 1 0 2 1) (1 4 1 5 0 1)) (("D" ((2 3 32) (1 1 10))) ("C" ((2 3 33) (1 1 10))) ("A" ((3 0 0) (2 3 31) (1 1 11))) ("B" ((2 3 33) (1 1 10))) ("Y" ((2 0 0) (1 4 41)))) (("D" (2 4 2 5 2) (2 1 2 2 2)) ("C" (2 4 2 5 2) (2 1 3 4 2)) ("A" (2 4 1 5 2) (2 4 2 7 2)) ("B" (2 4 2 5 2) (2 1 3 4 2)) ("Y" (1 4 1 5 0) (1 4 1 5 0))))

PMOS path NMOS path 2nd iteration: 1 PMOS, 3 NMOS, 4 D/S, 2 G

Gate Identification

- **Topology**-based gate **descriptor** proposal:

1. Total number of **transistors** and **nets**
2. Net classification: Number of **nets** with the same:
   a. Number of **PMOS** transistors
   b. Number of **NMOS** transistors
   c. Number of transistor **D/S-terms**
   d. Number of transistor **G-terms**
   e. Is gate **I/O term**?
3. For each I/O term, **shortest path** to reach the rest of I/O terms
4. If identical I/O terms in 3, for each shortest path, extend **transistor connectivity** scanning



net **classification**

I/O term **identification**

"8 8":(("**NAND4**" (3 0 2 2 0 0) (4 1 1 0 2 1) (1 4 1 5 0 1)) (("D" ((2 3 32) (1 1 10))) ("C" ((2 3 33) (1 1 10)))

("A" ((3 0 0) (2 3 31) (1 1 11))) ("B" ((2 3 33) (1 1 10))) ("Y" ((2 0 0) (1 4 41)))) (("D" (2 4 2 5 2)

(2 1 2 2 2)) ("C" (3 4 4 5 8) (3 3 4 6 6)) ("A" (2 4 1 5 2) (2 4 2 7 2)) ("B" (3 4 4 5 8) (3 4 4 11 6))

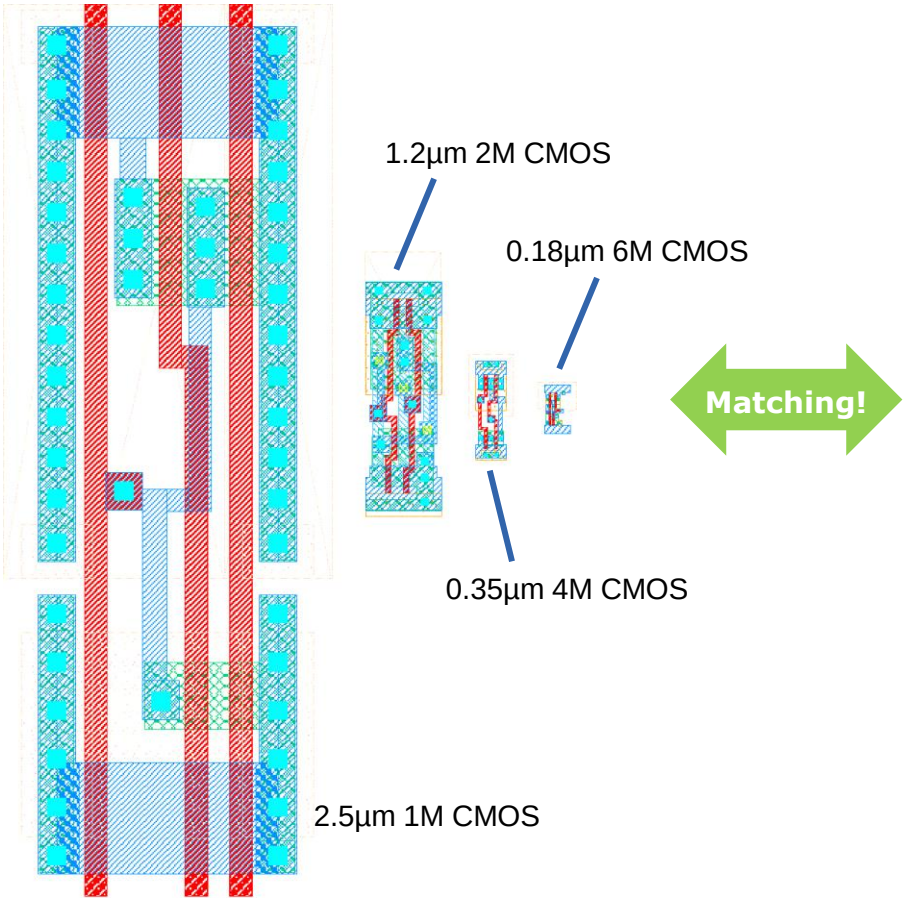
("Y" (1 4 1 5 0) (1 4 1 5 0))))

I/O term **disambiguation**

3rd iteration: 4 PMOS, 4 NMOS, 11 D/S, 6 G

Gate Identification

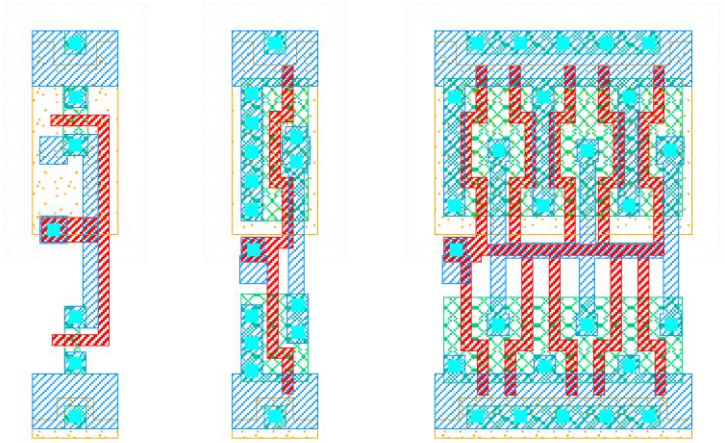
- **Multi-technology** methodology:



"4 4":(("**NAND2**" (1 0 2 2 0 0) (1 2 1 3 0 1)
(2 1 1 0 2 1)) (("**A**" ((2 1 11) (1 1 11)))
("**B**" ((2 1 12) (1 1 10))) ("**Y**" ((1 2 21)))) nil)

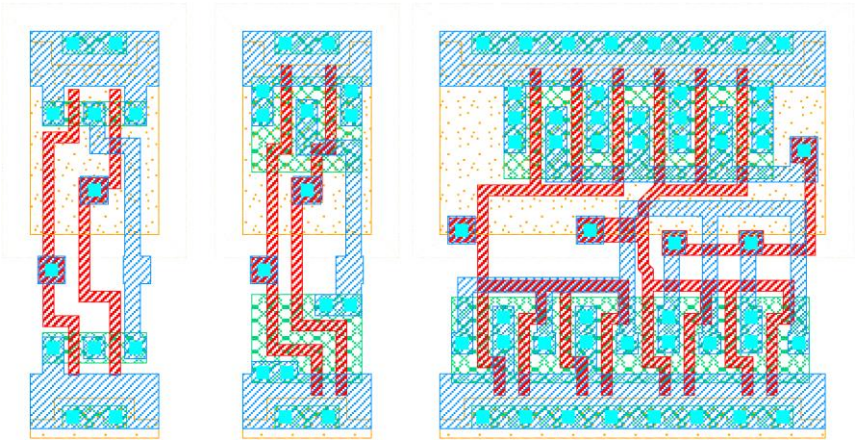
Gate Identification

- Independent from **transistor sizing** and **multiplicity**:



Matching!

"2 2":(("**INV**" (1 1 1 0 2 1) (1 1 1 2 0 1)
(("**A**" ((1 1 11))) ("**Y**" ((1 1 11))))
(("**Y**" (1 1 1 2 0) (1 1 1 2 0)) ("**A**" (1 1 1 0 2) (1 1 1 0 2))))

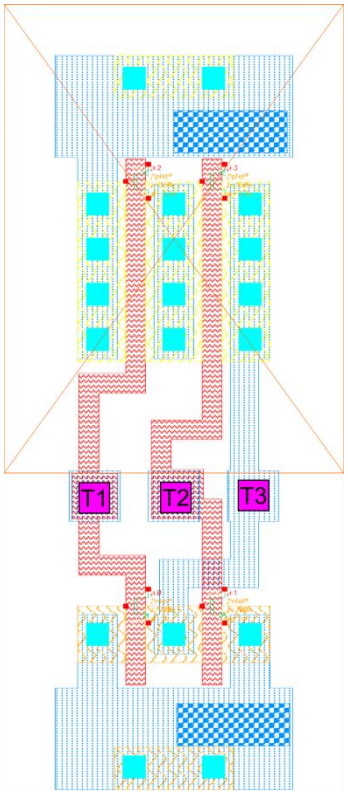


Matching!

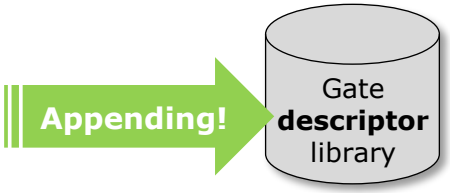
"4 4":(("**NAND2**" (1 0 2 2 0 0) (1 2 1 3 0 1)
(2 1 1 0 2 1)) (("**A**" ((2 1 11) (1 1 11)))
("**B**" ((2 1 12) (1 1 10))) ("**Y**" ((1 2 21)))) nil)

Analyses of New Topology

- If **unmatched** topology shows up:
  - New entry is **appended** to descriptor lib

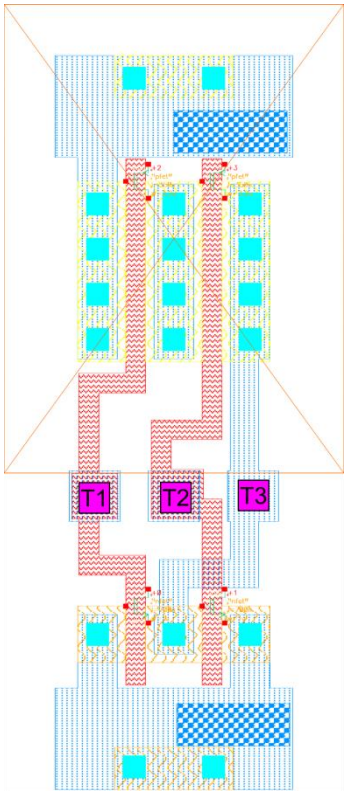


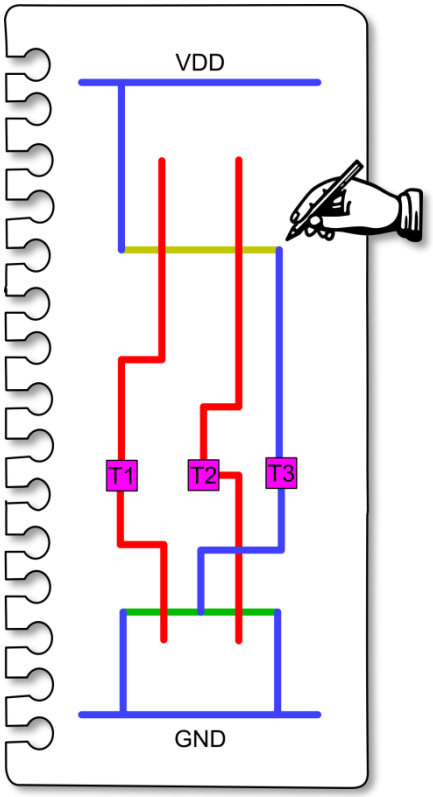
"4 4": (("**I256**" (2 1 1 0 2 1) (1 1 2 3 0 1)
(1 2 0 2 0 0)) (("**T2**" ((2 1 11) (1 1 11)))
("**T1**" ((2 1 21) (1 1 1))) ("**T3**" ((1 2 12)))) nil)

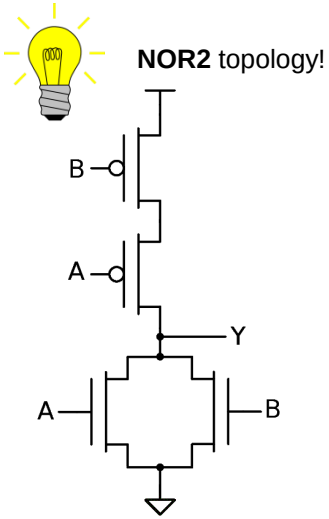


Analyses of New Topology

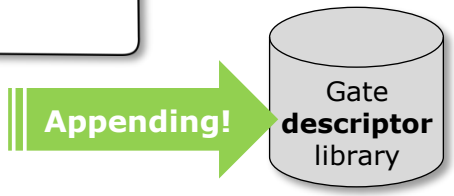
- If **unmatched** topology shows up:
  - New entry is **appended** to descriptor lib
  - **Manual analysis** of topology





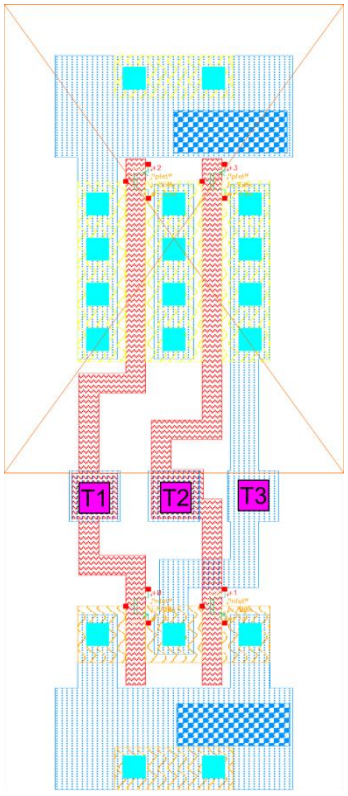


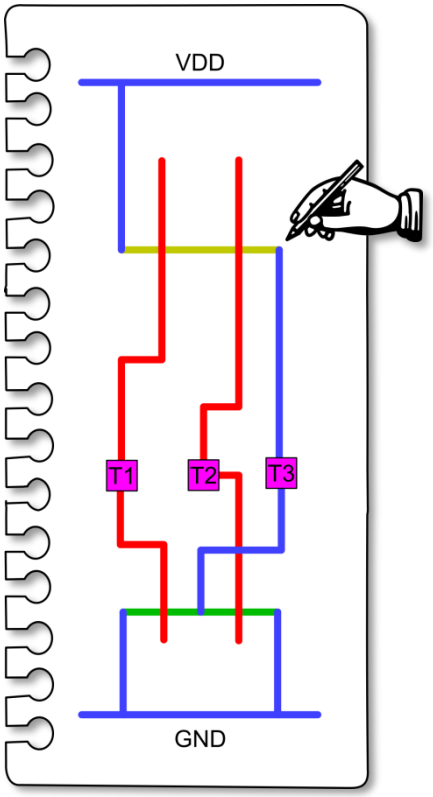
"4 4": (("**I256**" (2 1 1 0 2 1) (1 1 2 3 0 1)
(1 2 0 2 0 0)) (("**T2**" ((2 1 11) (1 1 11)))
("**T1**" ((2 1 21) (1 1 1))) ("**T3**" ((1 2 12)))) nil)

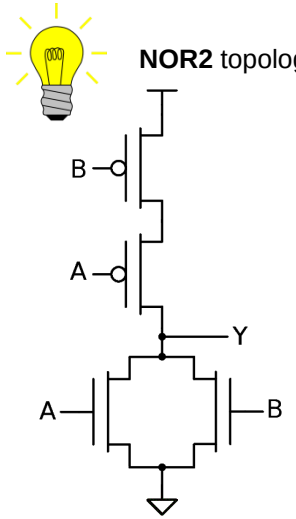


Analyses of New Topology

- If **unmatched** topology shows up:
  - New entry is **appended** to descriptor lib
  - **Manual analysis** of topology
  - Definition of descriptor-model **mapping**







"I256": ("NOR2" ("T1" "T2" "T3") ("B" "A" "Y"))

Gate **model** library

Mapping!

"4 4": (("**I256**" (2 1 1 0 2 1) (1 1 2 3 0 1)
(1 2 0 2 0 0)) (("**T2**" ((2 1 11) (1 1 11)))
("**T1**" ((2 1 21) (1 1 1))) ("**T3**" ((1 2 12)))) nil)

Appending!

Gate **descriptor** library

- To be done only ***one time*** per new topology ***ever***
- Descriptor lib **expands** project after project…