B4. Memory—2 Summary

Bei Yu

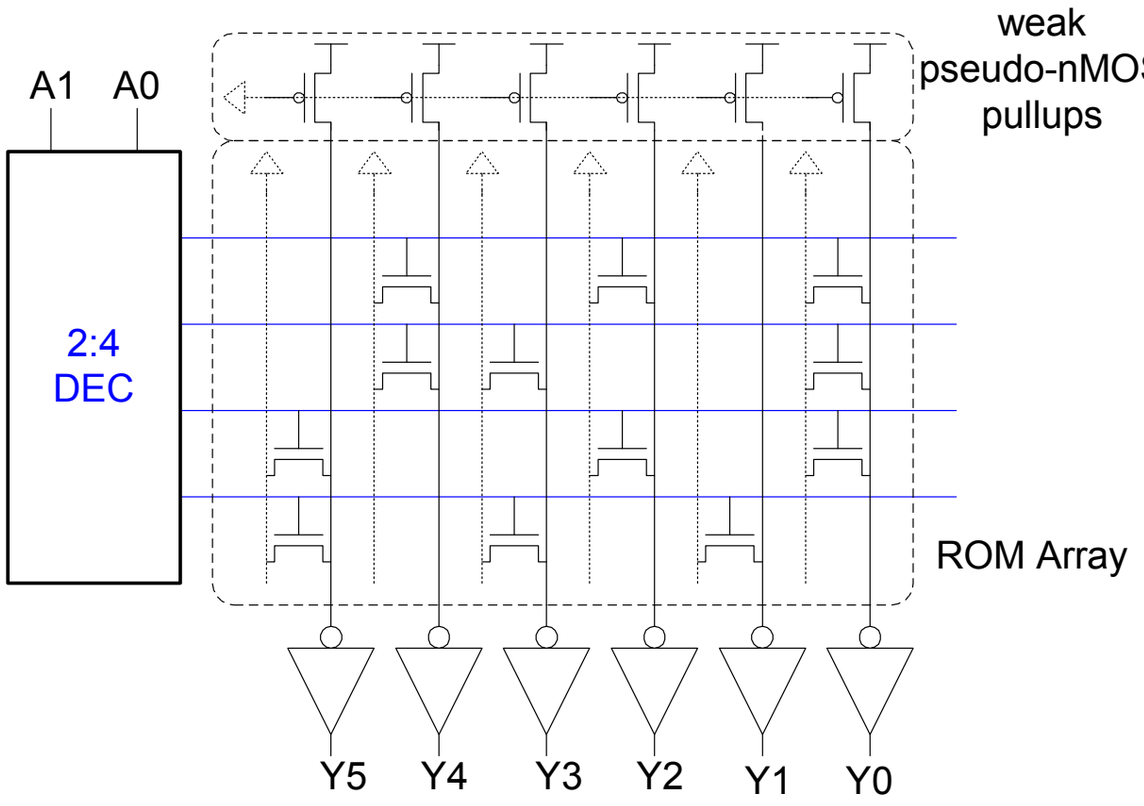
Reference:

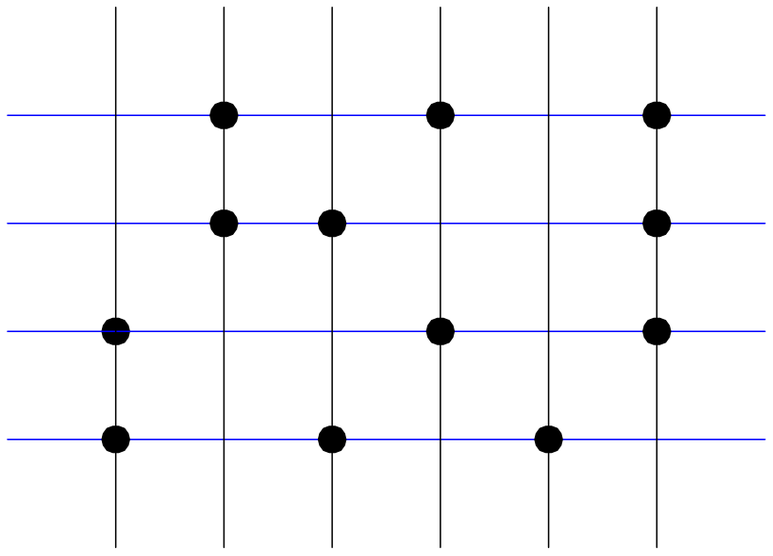
- Chapter 11 Memories
- **CMOS VLSI Design—A Circuits and Systems Perspective**
- by H.E.Weste and D.M.Harris

NOR ROM

- 4-word x 6-bit NOR-ROM
  - Selected word-line high
  - Represented with dot diagram

```
Word 0: 010101
Word 1: 011001
Word 2: 100101
Word 3: 101010
```





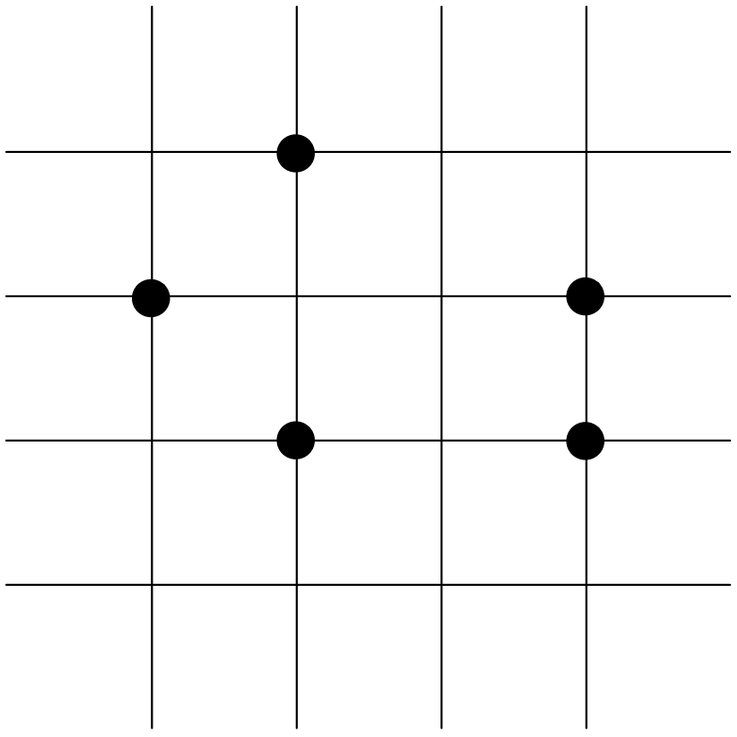
Looks like 6 4-input pseudo-nMOS NORs

EX: NOR ROM

- Draw 4-word 4-bit NOR-ROM structure and dot diagram

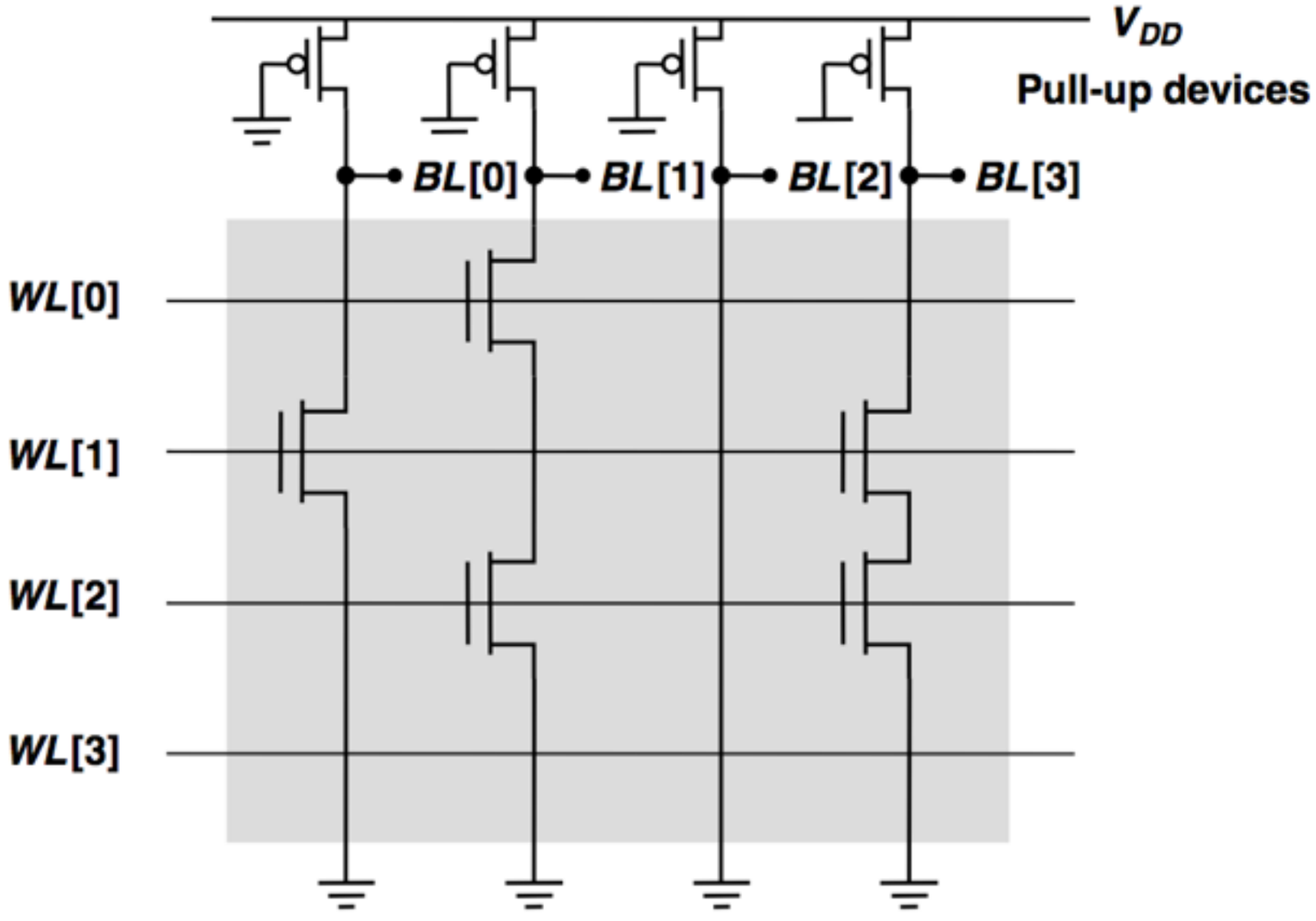
```
Word 0: 0100
Word 1: 1001
Word 2: 0101
Word 3: 0000
```

- Answer:



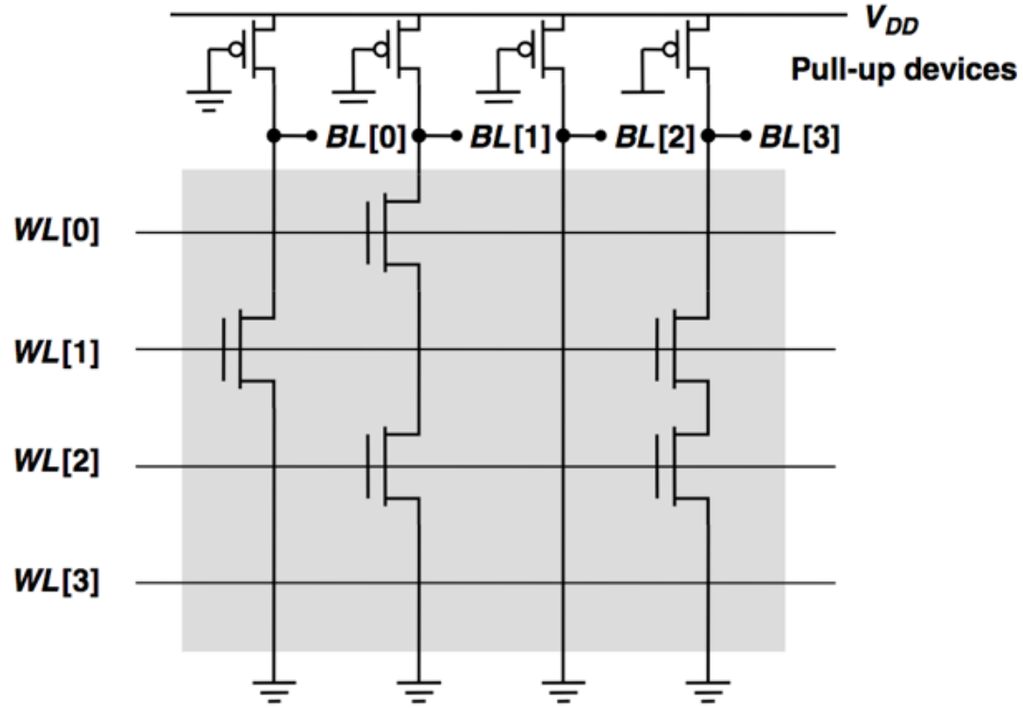
NAND ROM

- 4-word x 4-bit NAND-ROM
  - All word-lines high with exception of selected row



EX. NAND ROM

- Question: What's it function?



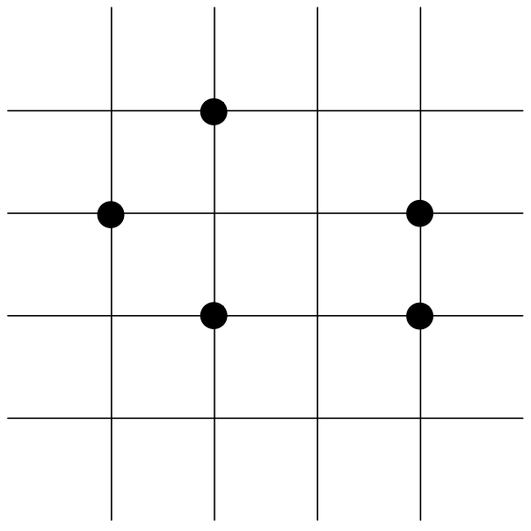
- Answer:

WL[0]=0:**0100**

WL[1]=0:**1001**

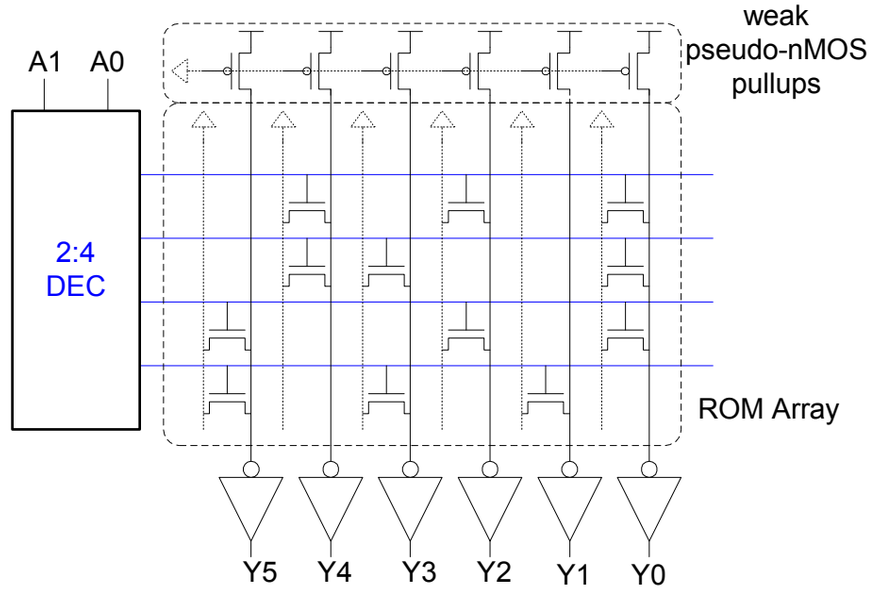
WL[2]=0:**0101**

WL[3]=0:**0000**



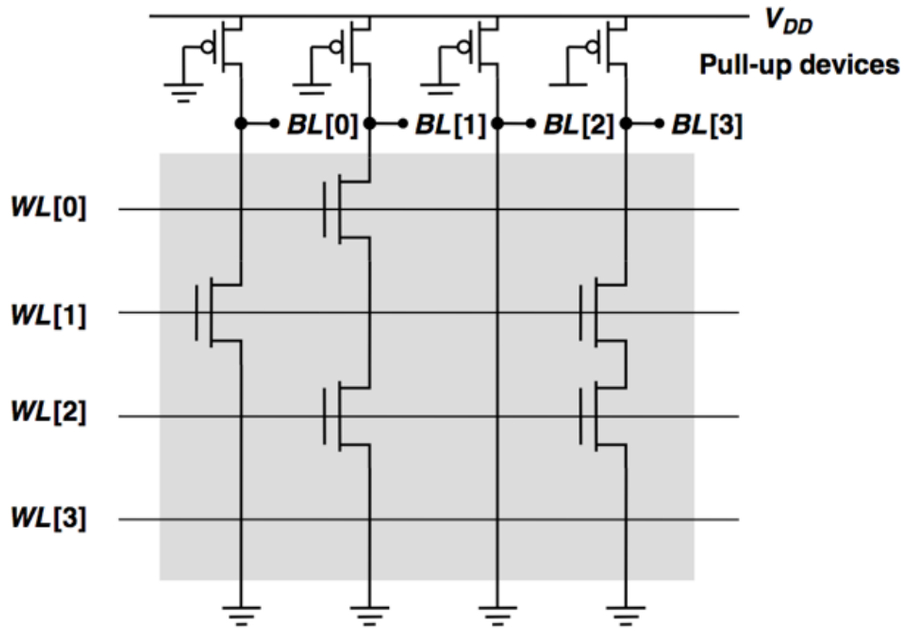
NOR ROM v.s. NAND ROM

- NOR ROM:
  - (+) Faster
  - (-) Larger Area (VDD lines)



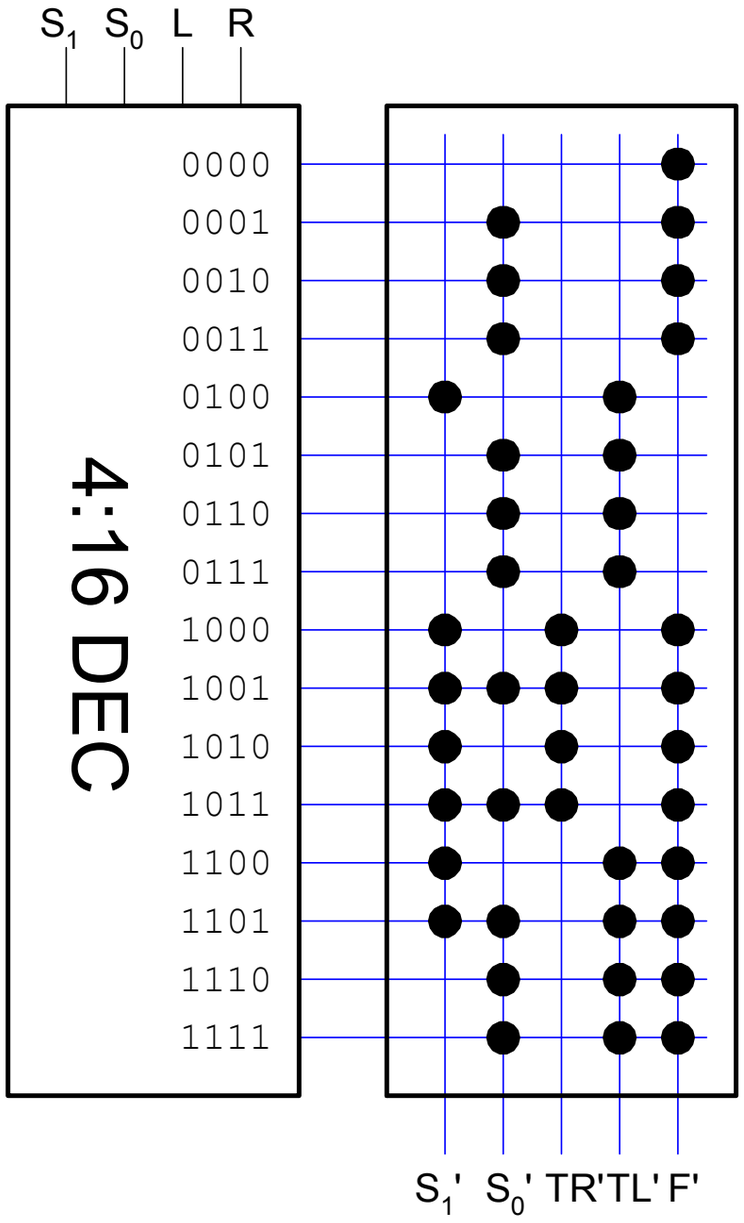
- NAND ROM:
  - (+) High density, small area
  - (-) Slower

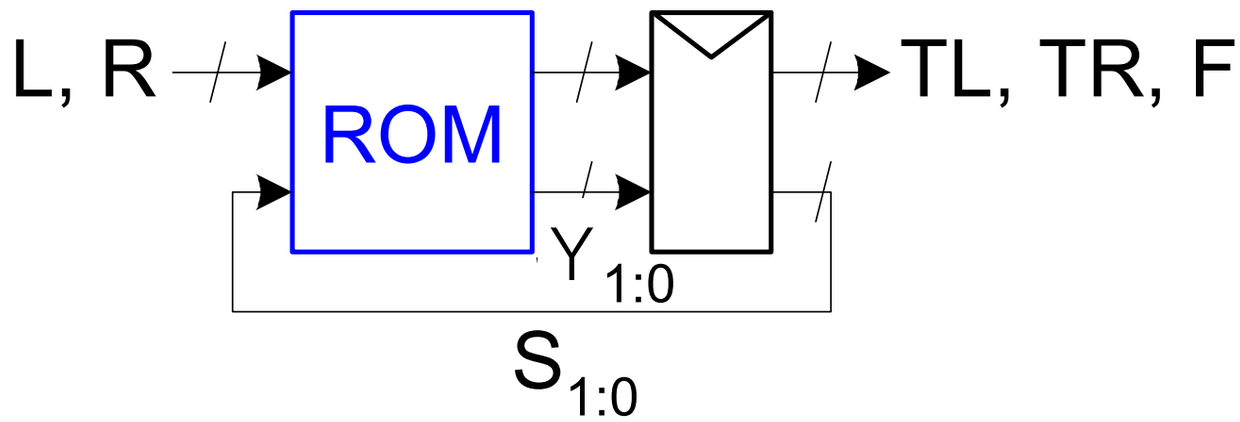
delay grows quadratically with the number of series transistors discharging the bitline.



ROM Implementation

- 16-word x 5 bit ROM





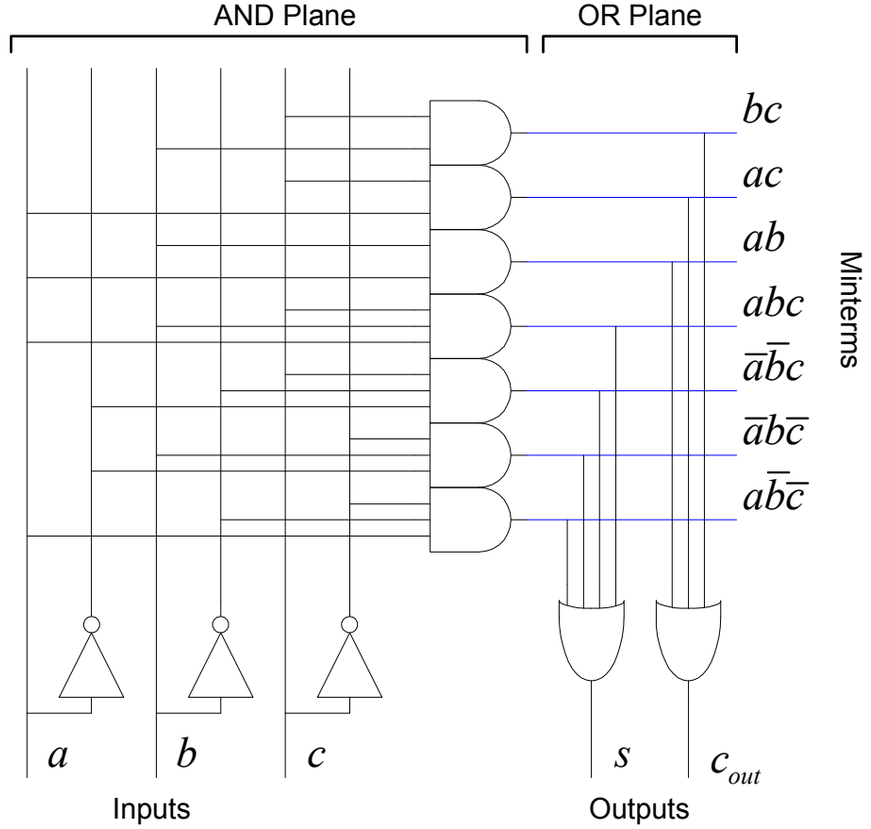
PLAs

- A *Programmable Logic Array* performs any function in sum-of-products form.
- *Literals*: inputs & complements
- *Products / Minterms*: AND of literals
- *Outputs*: OR of Minterms

- Example: Full Adder

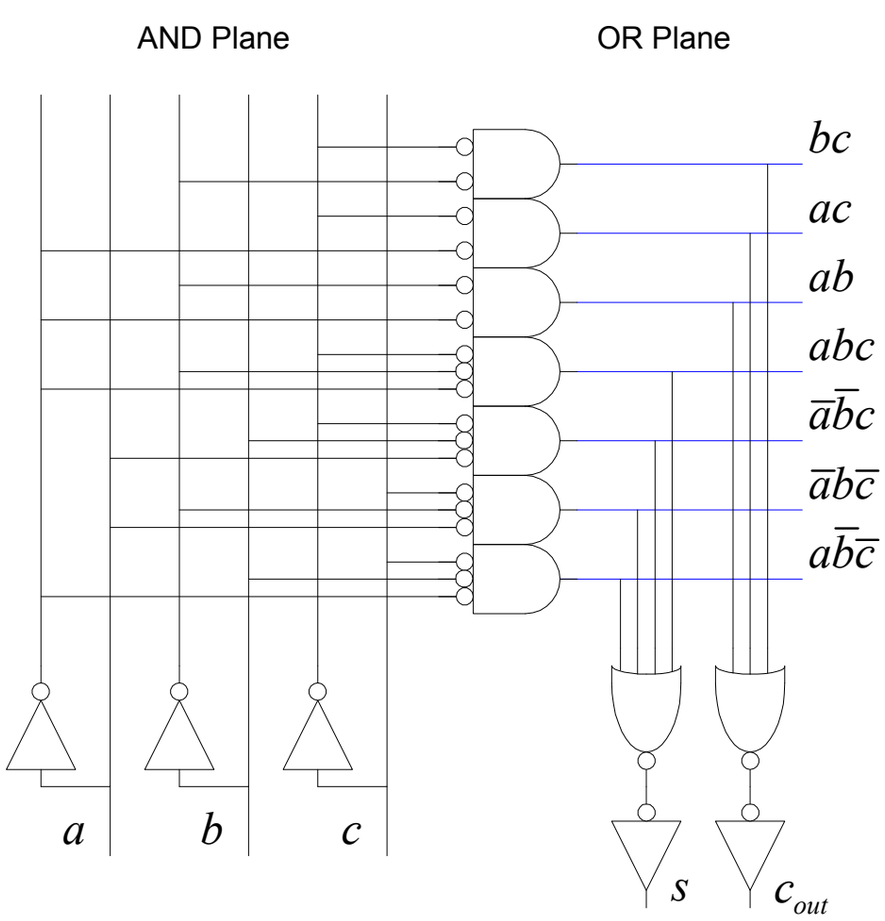
$$s = a\bar{b}\bar{c} + \bar{a}b\bar{c} + \bar{a}\bar{b}c + abc$$

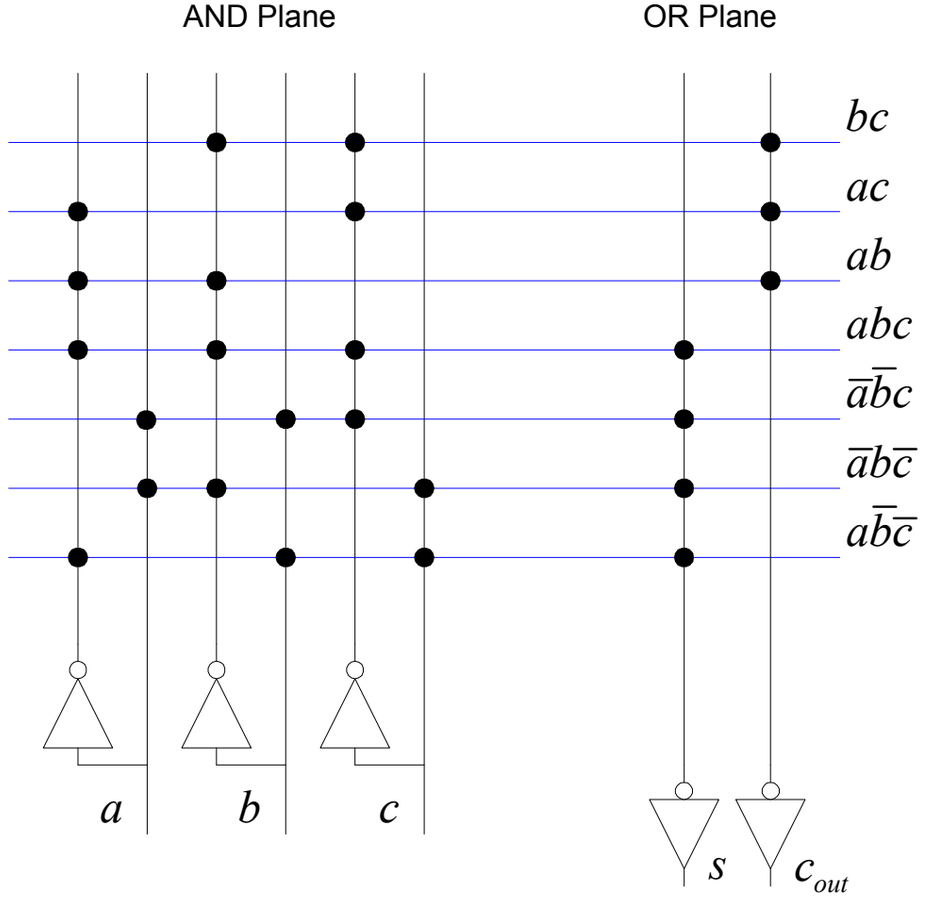
$$c_{\text{out}} = ab + bc + ac$$



NOR-NOR PLAs

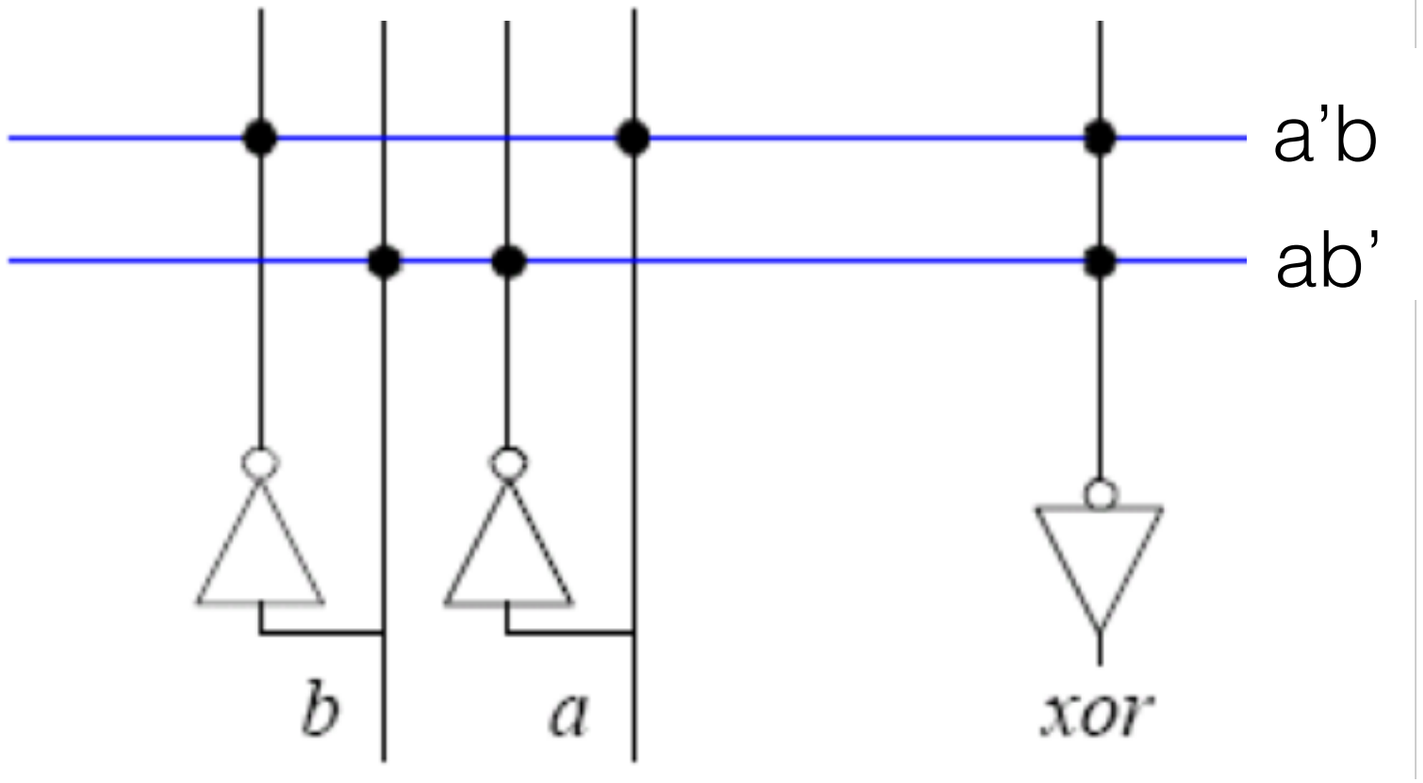
- ANDs and ORs not very efficient in CMOS
- Dynamic or Pseudo-nMOS NORs very efficient
- Use DeMorgan's Law to convert to all NORs





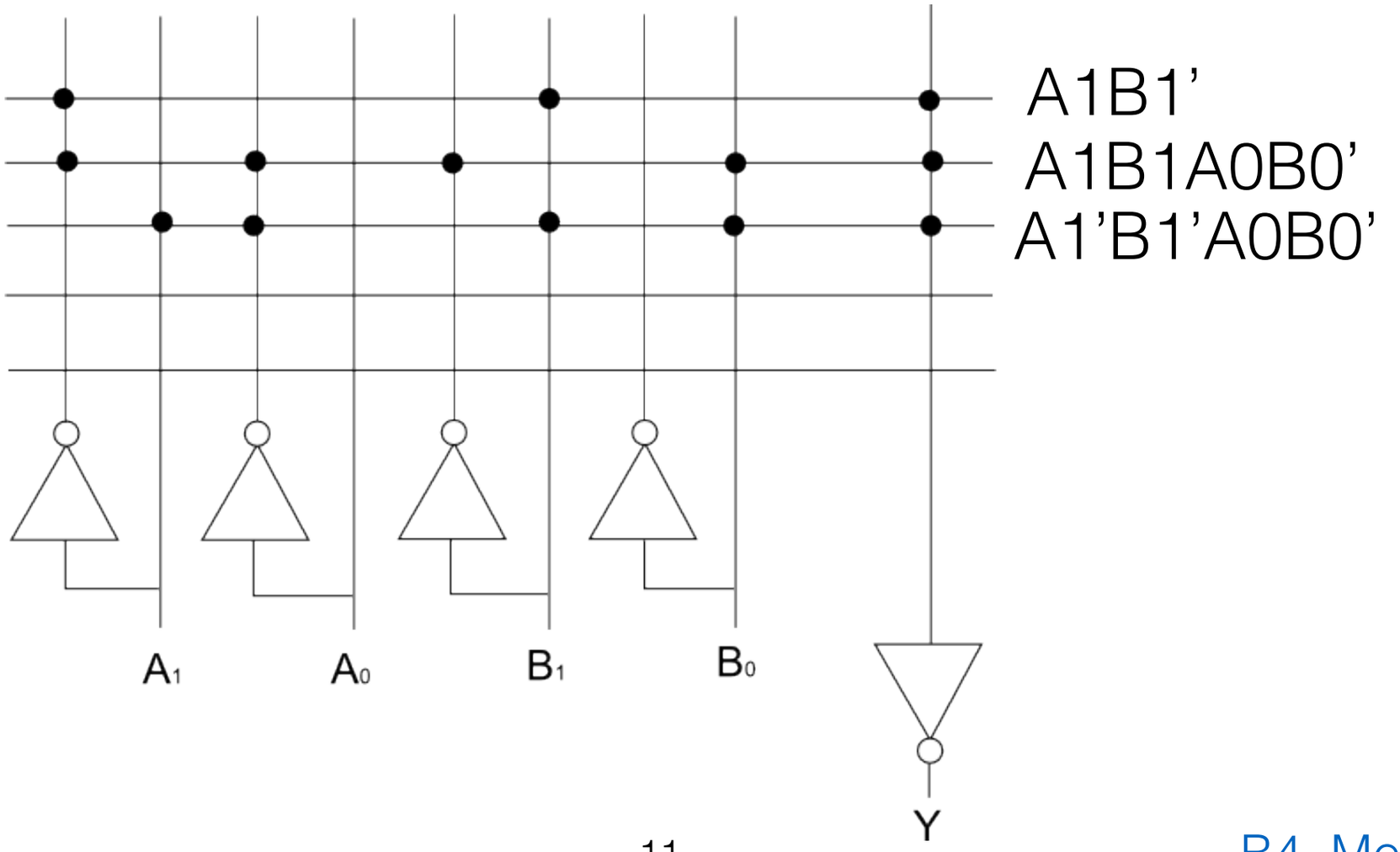
PLA Question 1

- Using PLA, design a comparator with function a≠b
- Answer: a’b + ab’



PLA Question 2

- Design PLA comparator with function A1A0 > B1B0
- Answer: A1B1’ + A1B1A0B0’ + A1’B1’A0B0’



RoboAnt PLA

- Convert state transition table to logic
- Karnaugh map

$S_{1:0}$	L	R	$Y_{1:0}$	TR	TL	F
00	0	0	00	0	0	1
00	1	X	01	0	0	1
00	0	1	01	0	0	1
01	1		01	0	1	0
01	0	1	01	0	1	0
01	0	0	10	0	1	0
10	X	0	10	1	0	1
10	X	1	11	1	0	1
11	1	X	01	0	1	1
11	0	0	10	0	1	1
11	0	1	11	0	1	1

S1′

LR \ S_1S_0	00	01	11	10
00	0	1	1	1
01	0	0	1	1
11	0	0	0	1
10	0	0	0	1

$$Y1 = S_1\overline{S_0} + \overline{L}S_1 + \overline{LR}S_0$$

S0′

LR \ S_1S_0	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	1	1	1	1
10	1	1	1	0

$$Y0 = R + L\overline{S_1} + LS_0$$

$$TR = S_1\overline{S_0}$$

$$TL = S_0$$

$$F = S_1 + \overline{S_0}$$

EX. RoboAnt Dot Diagram

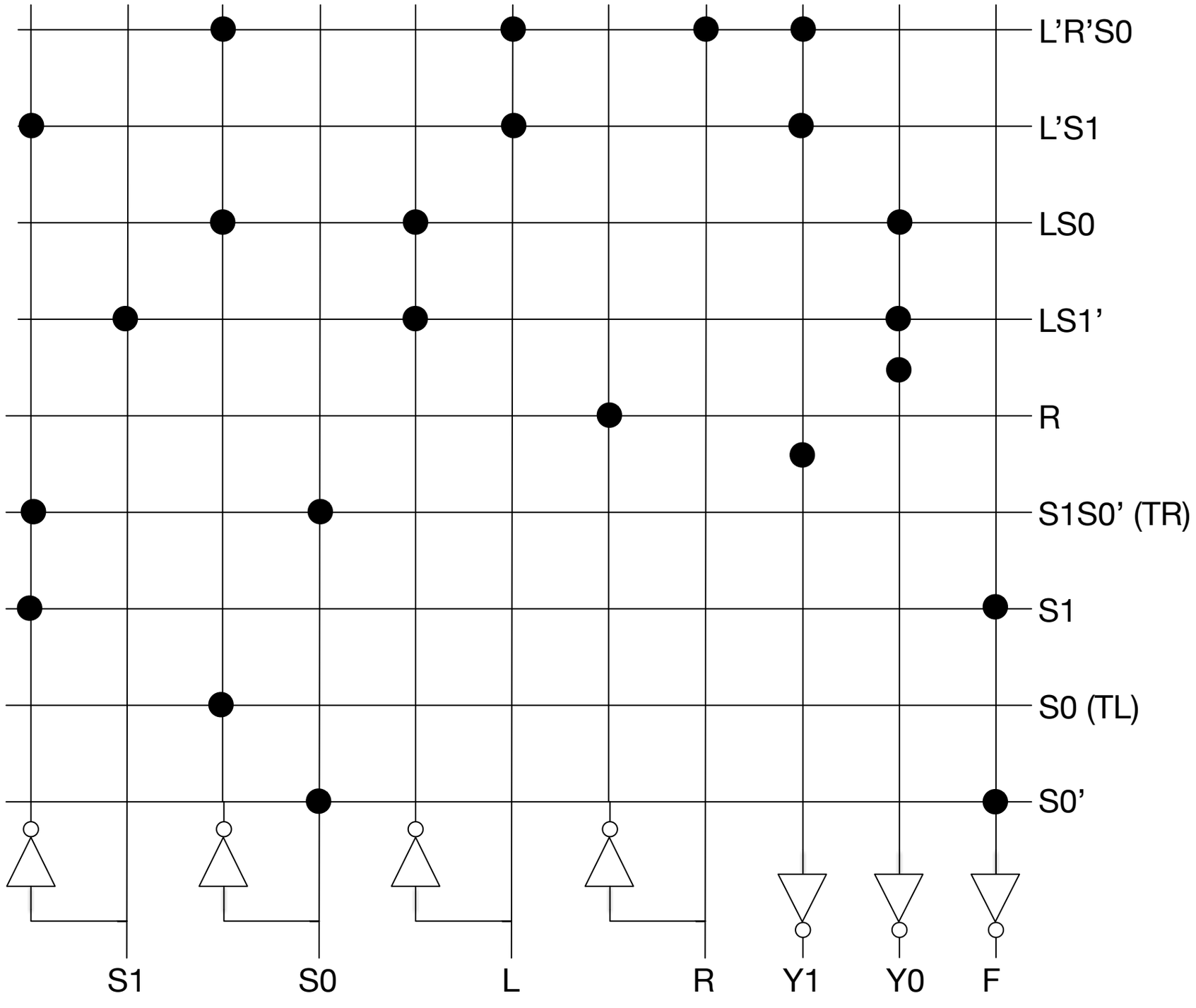
$$Y1 = S_1\overline{S_0} + \overline{L}S_1 + \overline{L}\,\overline{R}S_0$$

$$Y0 = R + L\overline{S_1} + LS_0$$

$$TR = S_1\overline{S_0}$$

$$TL = S_0$$

$$F = S_1 + \overline{S_0}$$



PLAs vs. ROMs

- The OR plane of the PLA is like the ROM array
- The AND plane of the PLA is like the ROM decoder
- PLAs are more flexible than ROMs
  - No need to have 2^n rows for n inputs
  - Only generate the minterms that are needed
  - Take advantage of logic simplification