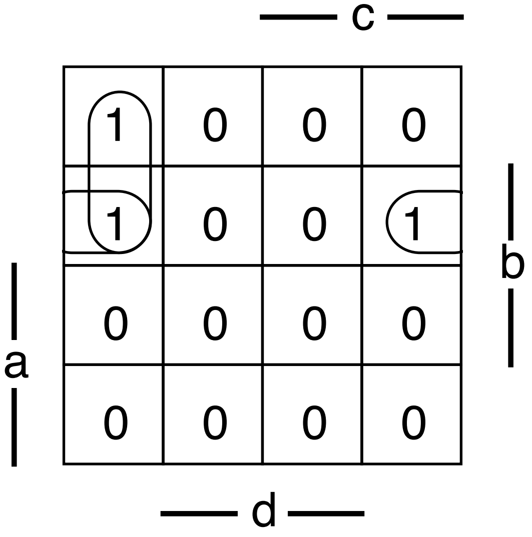
Implementing a Truth Table Using An And-Or Structure (Review)

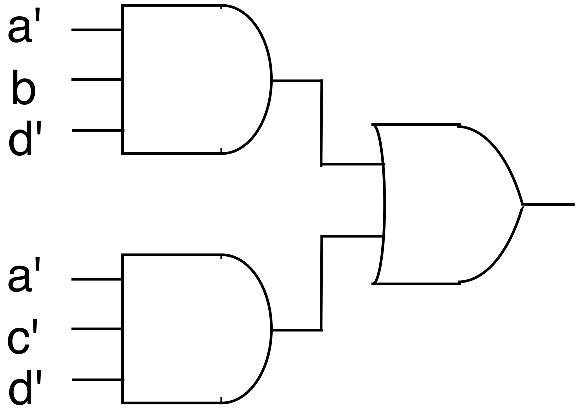
- Given a truth table, we can use a Karnaugh map to find the minimum 2-level SOP implementation

a	b	c	d	x
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0



1	0	0	0
1	0	0	1
0	0	0	0
0	0	0	0

$x = a'bd' + a'c'd'$



PLAs

- A 2-level *and-or* structure is replicated many times in a programmable array called a *PLA* (*programmable logic array*)
  - Parts of a CPU’s datapath or next-state logic can be built out of PLAs
  - Small circuits can be built out of PLAs
- At the input of each gate, there's a “fuse” which can be left whole, or broken
  - So the designer can control which inputs go to each **and** gate, and which outputs of the **and** gates go to each **or** gate
- A PLA can be either
  - Mask programmable — customer orders a programmed PLA from the manufacturer
  - Field programmable — customer can program PLA (once)

PLAs

- A 2-level *and-or* structure is replicated many times in a programmable array called a *PLA* (*programmable logic array*)

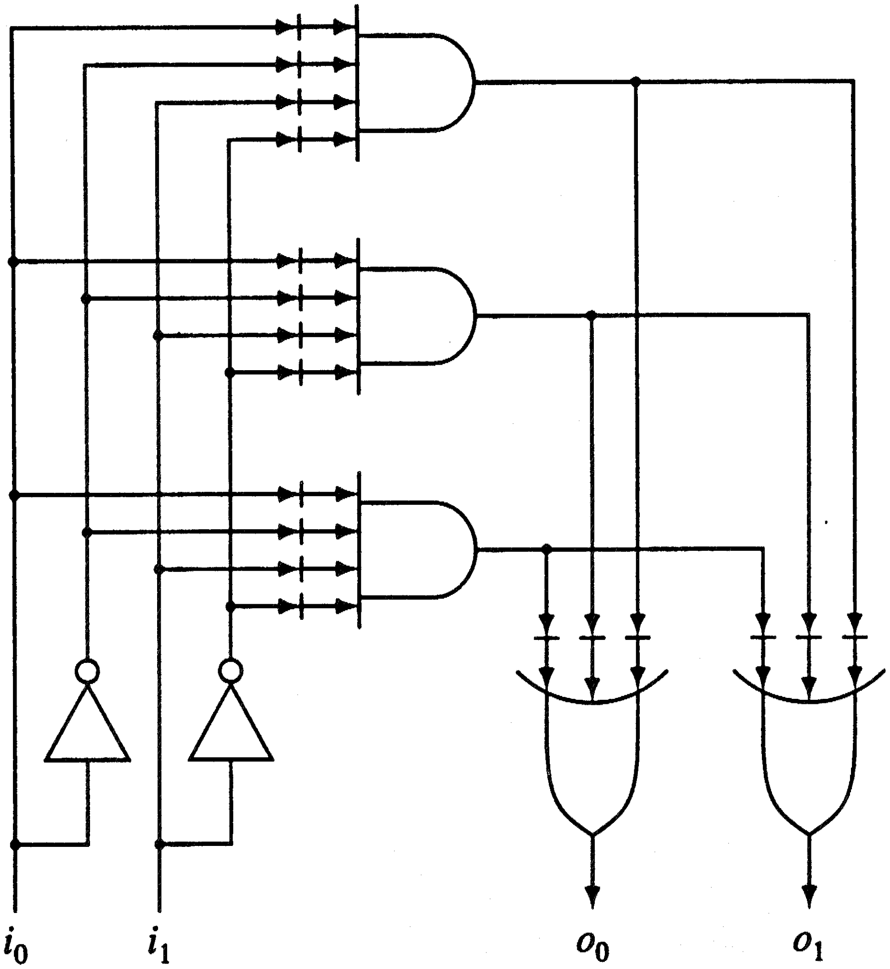


Diagram from *Computer Systems*, Maccabe, Irwin 1993

- This PLA has 2 inputs, 2 outputs, and can represent up to 3 product terms

PLA Example

- This is an *abstract* diagram of a PLA with 6 inputs, 4 outputs, which can represent up to 12 product terms

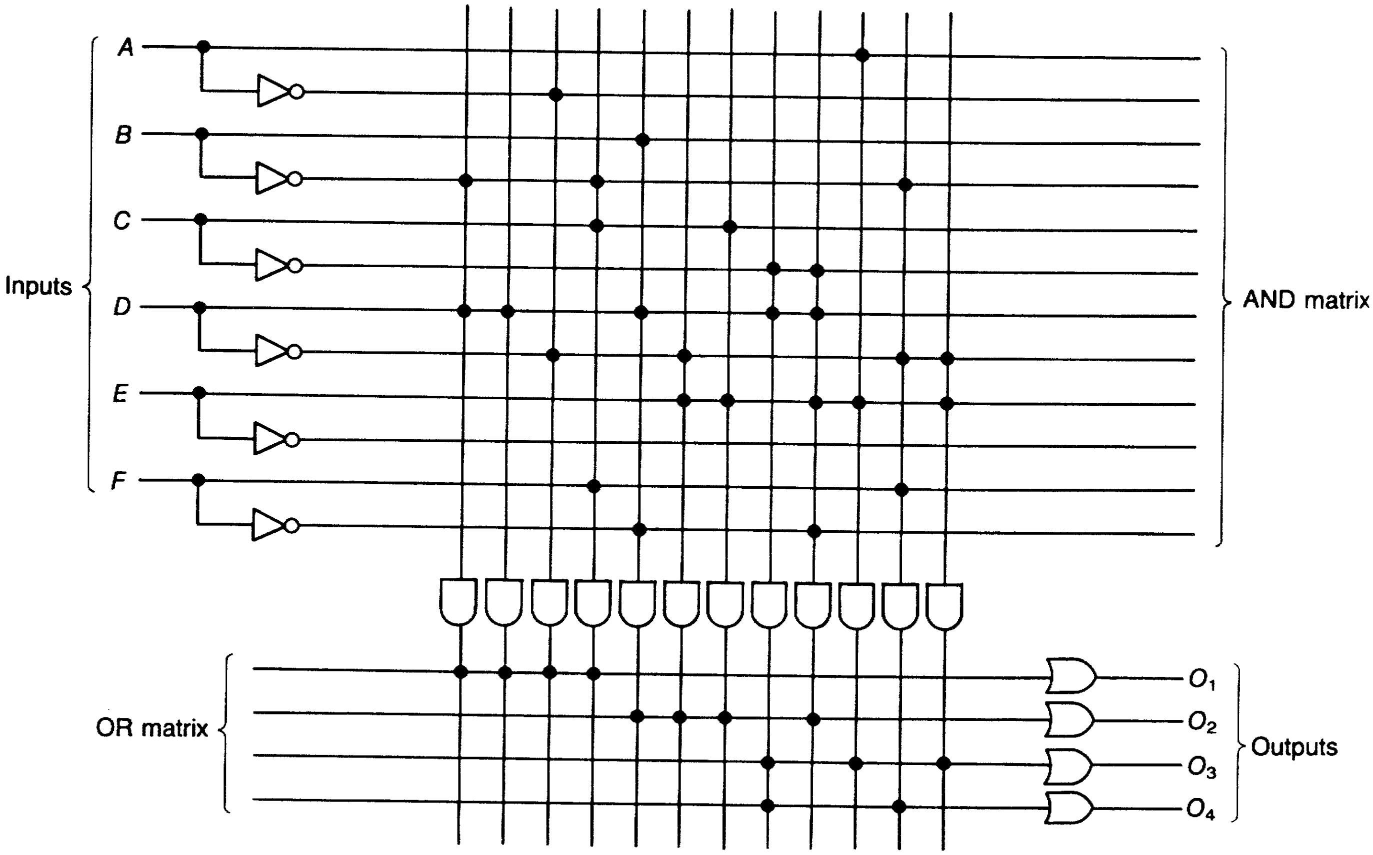


Diagram from *Digital Design*, Johnson & Karim, PWS-Kent 1987

- Try the Java KMap->PLA animation at http://tech-www.informatik.uni-hamburg.de/applets/kvd

Field-Programmable Logic Device

- The next evolutionary step beyond the PLA is the *field-programmable logic device* (FPLD), also called the:
  - Field-programmable gate array (FPGA)
  - Complex programmable logic device (CPLD)
- FPLD characteristics
  - Based on either an array of PLA-like *and-or* structures, or on look-up tables
  - May include D (or more complex) flip-flops, to more easily build sequential circuits, possibly even RAM
  - Many can be "programmed" repeatedly
    - Connect I/O to interconnect, connect interconnect to cells, control cell functions
  - Available in different sizes up to 500,000 gates (100MHz, 2.5 volt, 0.25μ, 5 metal)

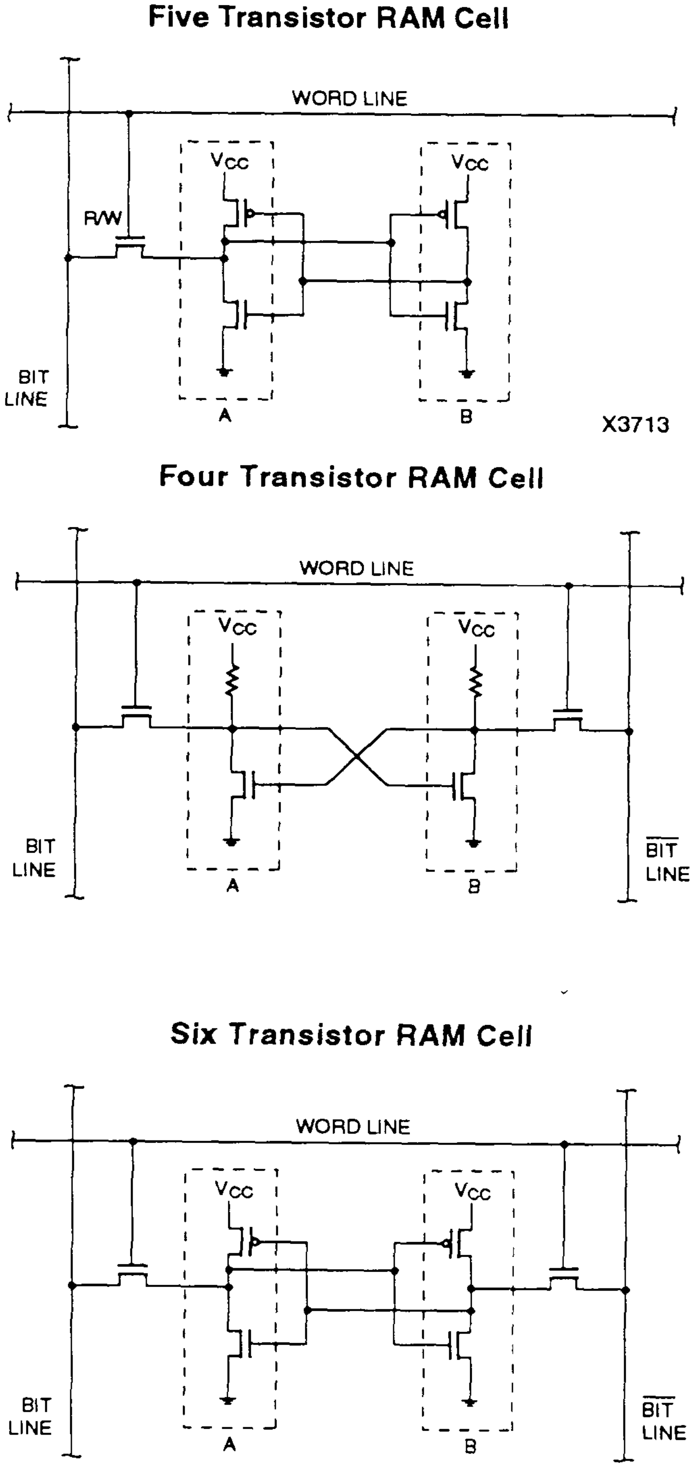
Programming Using Antifuses

- An *antifuse* is normally open (“off”); when enough current (5–15mA) passes through it it closes (“on”)
  - Current melts a thin insulating dielectric and forms a permanent silicon link
  - Disadvantage — can only program <u>once</u>
    - Programmed in a special hardware device
    - An antifuse FPLD may contain 750,000 antifuses, but only about 2% of them typically need to be programmed
    - Takes about 5-10 minutes for each chip
- Advantages:
  - Small — about the size of a via
  - Low resistance, low capacitance = fast

Programming Using EPROMs & EEPROMs (Floating Gates)

- EPROM programming & operation:
  - To program: a high programming voltage is applied, semi-permanently turning the transistor off
  - To erase: the transistor is exposed to UV light, which returns the transistor to normal operation
  - Can be reprogrammed many times
- EEPROMs are similar, but are erased electrically
  - Faster to erase than EPROM, and can be done "in-circuit"
  - Requires larger cell than EPROM
- Advantages
  - Can be programmed repeatedly, in-circuit
  - Fairly small — requires only 1 transistor

Programming Using Static RAMs (SRAMs)



Five Transistor RAM Cell

Four Transistor RAM Cell

Six Transistor RAM Cell

Figure from *Field-Programmable Gate Array Technology*, Trimberger, Kluwer, 1994

Programming Using Static RAMs (SRAMs) (cont.)

- Disadvantages:
  - Must load configuration from ROM, disk, etc. on power-up
  - Large — requires several transistors
- Advantages:
  - Can be programmed repeatedly, in-circuit
    - Can be programmed quickly (< 1ms)
  - Part has been 100% tested at factory
  - Same basic process as CMOS, so quickly takes advantage of new fab processes
    - CMOS also requires less power than circuits requiring pull-up resistors
- SRAMs can be used in FPLDs to :
  - Connect inputs to cells, or even to replace the cell if it's a LUT
  - Connect cells to interconnect

Types of FPLDs

Type of Base Cell

Programming Method	**Multiplexor**	**Look-Up Table (LUT)**	**AND-OR**
Antifuse	**Actel** ACT 1, ACT 2, ACT 3 **Quicklogic** **Crosspoint**		
EPROM			**Altera** MAX 5000, 7000 *(Salcic 2.1)* **Xilinx** EPLD
SRAM	**Plessy**	**Altera** Flex 8000, Flex 10K *(Salcic 2.2)* **Xilinx** LCA 2000, 3000, 4000 *(Salcic 2.3)*	

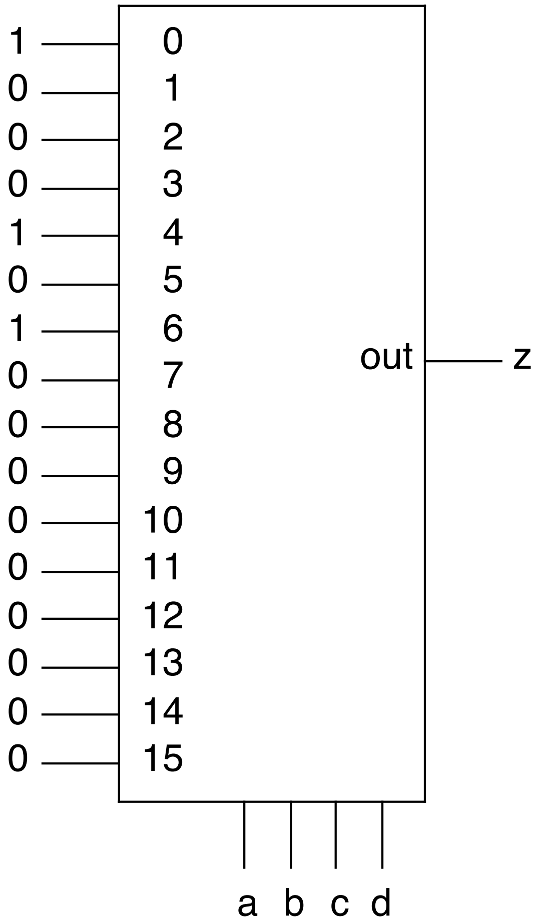
FPGAs **CPLDs**

- Layout / routing
  - Row-based: Actel
  - Matrix-based: Altera, Quicklogic, Xilinx

Implementing a Truth Table Using a Multiplexor

- Besides and-or structures, an alternative is to use a 16-input multiplexor

a	b	c	d	x
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

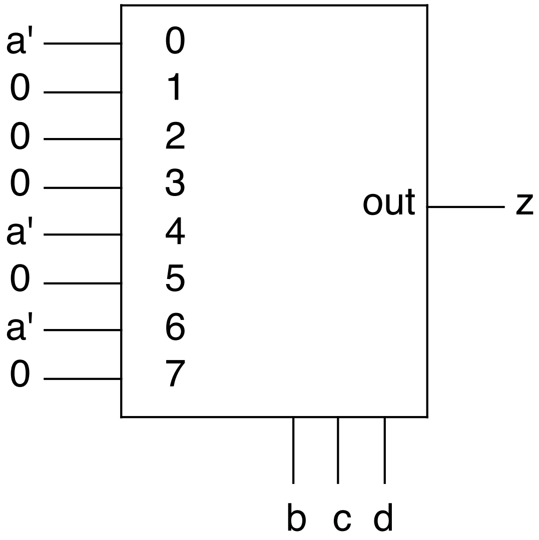


- Any function of N inputs can be implemented using a 2^N to 1 multiplexor

Implementing a Truth Table Using a Multiplexor (cont.)

- An alternative is to “fold” the truth table, and tie each input to either 1, 0, or the MSB, and only use a 8-input multiplexor

a	b	c	d	x
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

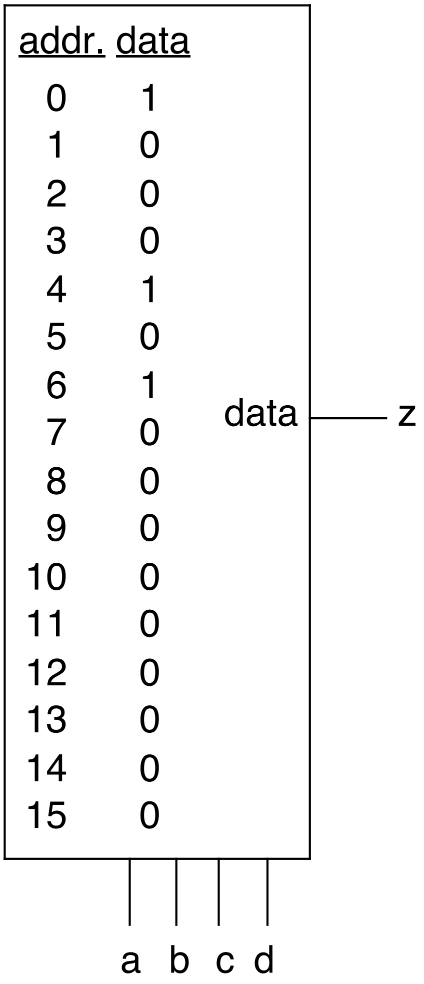


- Any function of N inputs can be implemented using a 2^{N-1} to 1 multiplexor
  - Some FPLDs are based on multiplexors, and attach simple gates to selector lines

Implementing a Truth Table Using a ROM

- Yet another alternative is to use a ROM

a	b	c	d	x
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0



addr.	data
0	1
1	0
2	0
3	0
4	1
5	0
6	1
7	0
8	0
9	0
10	0
11	0
12	0
13	0
14	0
15	0

- Any function of N inputs can be implemented using a 2^Nx 1 bit ROM
  - Some FPLDs are based on static RAMs (SRAMS) loaded at power-up; these are said to use *look-up tables* (LUTs)

Row-Based Layout

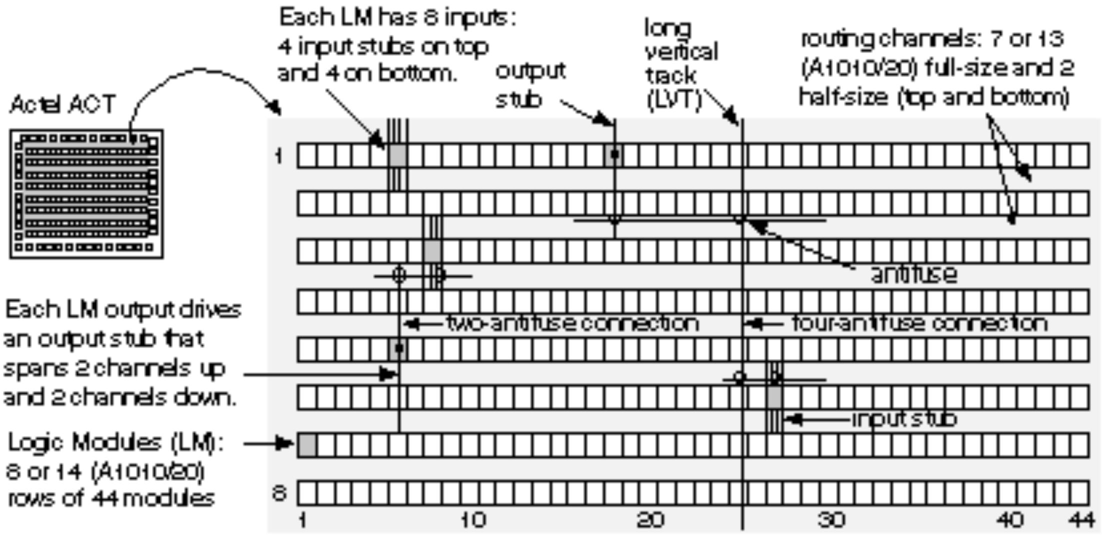

Actel ACT
Each LM has 8 inputs: 4 input stubs on top and 4 on bottom.
output stub
long vertical track (LVT)
routing channels: 7 or 13 (A1010/20) full-size and 2 half-size (top and bottom)
1
antifuse
Each LM output drives an output stub that spans 2 channels up and 2 channels down.
two-antifuse connection
four-antifuse connection
input stub
Logic Modules (LM): 8 or 14 (A1010/20) rows of 44 modules
8
1
10
20
30
40
44

Figure from *Application-Specific Integrated Circuits*, Smith, Addison-Wesley, 1997

- Cells are arranged in rows
  - Horizontal channels between rows
  - Vertical channels above cells: some short, some long
  - Each *channel* contains a fixed number of *tracks*, each track holds one wire
    - Wires may be divided into fixed-length *segments* within each track
  - In figure above, cell inputs connect to horizontal wires, outputs to vertical wires

Matrix-Based Layout

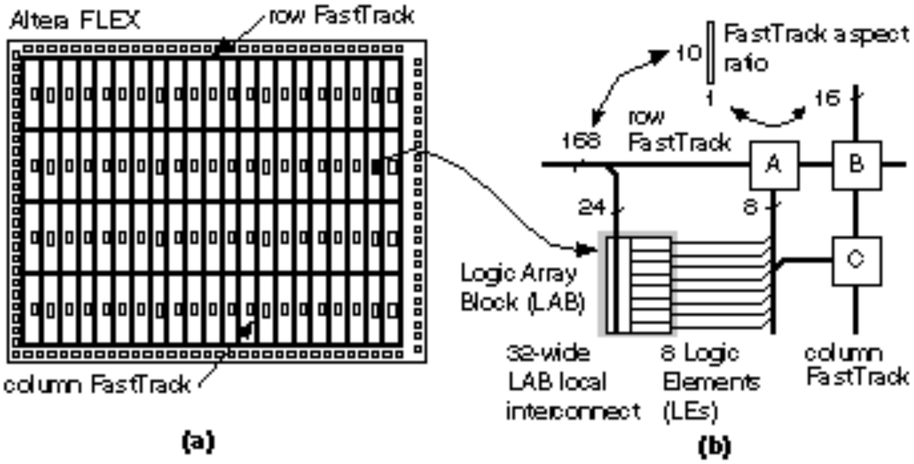
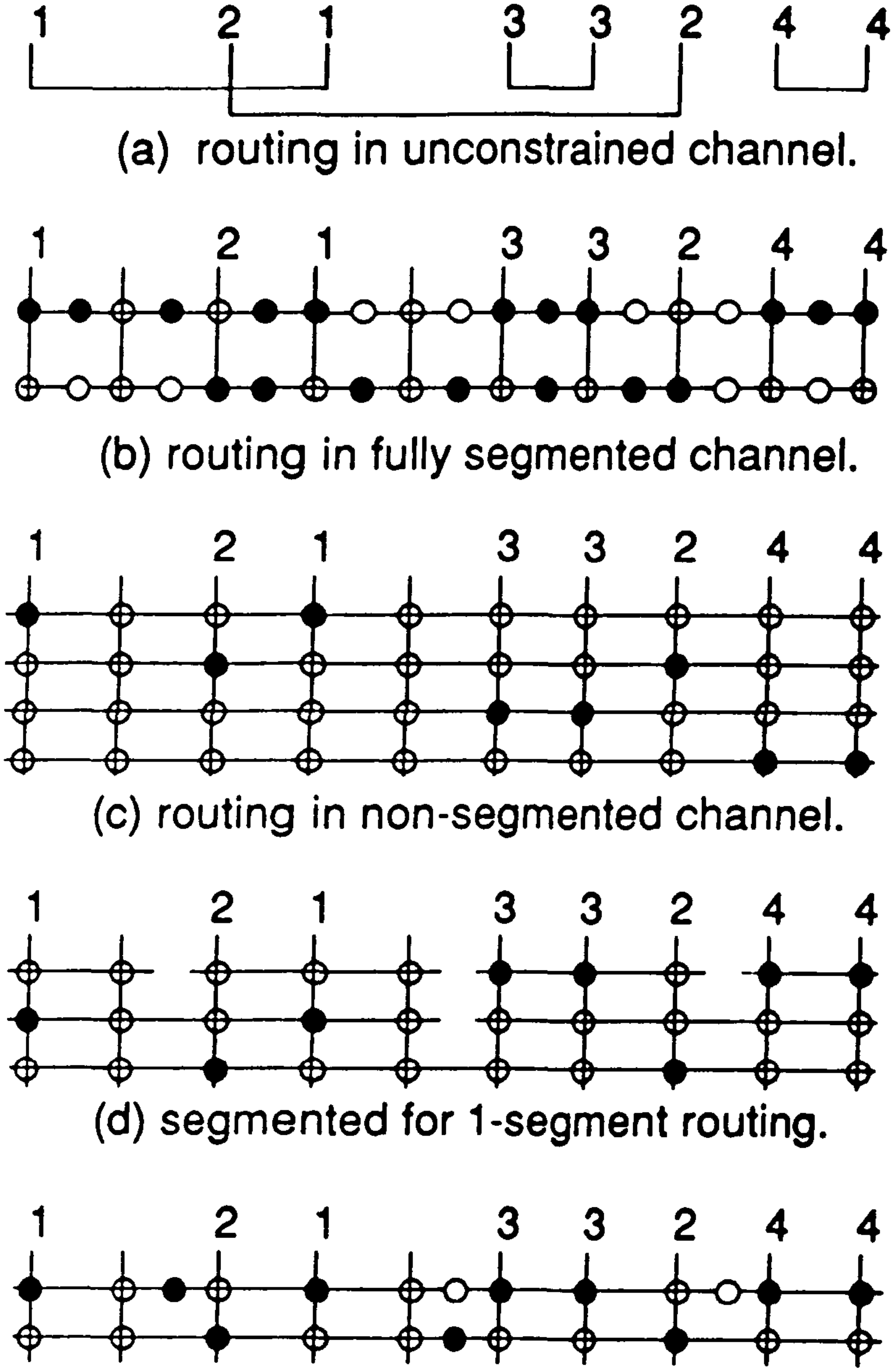


Figure from *Application-Specific Integrated Circuits*, Smith, Addison-Wesley, 1997

- Cells are arranged in an array (*matrix*)
  - Horizontal and vertical channels between cells
  - Each *channel* contains a fixed number of *tracks*, each track holds one wire
  - In figure above:
    - Cell inputs connect to horizontal tracks
    - Box A connects cell output(s) to horizontal tracks, and box C connects cell output(s) to vertical tracks
    - Box B acts as a switchbox between horizontal and vertical tracks

Antifuse Routing



(a) routing in unconstrained channel.

(b) routing in fully segmented channel.

(c) routing in non-segmented channel.

(d) segmented for 1-segment routing.

(e) segmented for 2-segment routing.

Figure from *Field-Programmable Gate Array Technology*, Trimberger, Kluwer, 1994

Antifuse Routing (cont.)

- Fully segmented
  - Switch at every cross point normally passes signals through vertically and horizontally, but can connect the vertical and horizontal tracks
  - Antifuse connects or disconnects the segments of the horizontal channel
- Non-segmented
  - Excessive area requirements
- 1-segment routing
  - Divides the tracks into segments of varying lengths, which allows each net to be routed in a track of more or less the appropriate size
- 2-segment routing
  - Allows track segments to be joined