

Digital Design

Chapter 7: Physical Implementation

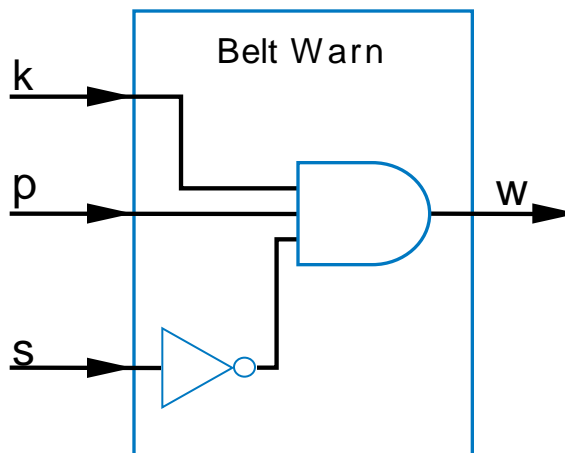
Slides to accompany the textbook *Digital Design*, First Edition,
by Frank Vahid, John Wiley and Sons Publishers, 2007.
<http://www.ddvahid.com>

Copyright © 2007 Frank Vahid

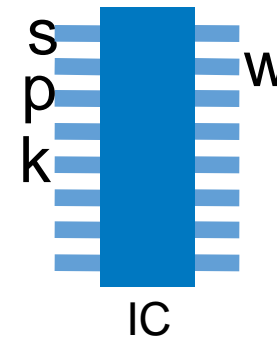
Instructors of courses requiring Vahid's *Digital Design* textbook (published by John Wiley and Sons) have permission to modify and use these slides for customary course-related activities, subject to keeping this copyright notice in place and unmodified. These slides may be posted as unanimated pdf versions on publicly-accessible course websites.. PowerPoint source (or pdf with animations) may not be posted to publicly-accessible websites, but may be posted for students on internal protected sites or distributed directly to students by other electronic means. Instructors may make printouts of the slides available to students for a reasonable photocopying charge, without incurring royalties. Any other use requires explicit permission. Instructors may obtain PowerPoint source or obtain special use permissions from Wiley – see <http://www.ddvahid.com> for information.

Introduction

- A digital circuit design is just an idea, perhaps drawn on paper
- We eventually need to implement the circuit on a physical device
 - How do we get from (a) to (b)?



(a) Digital circuit
design

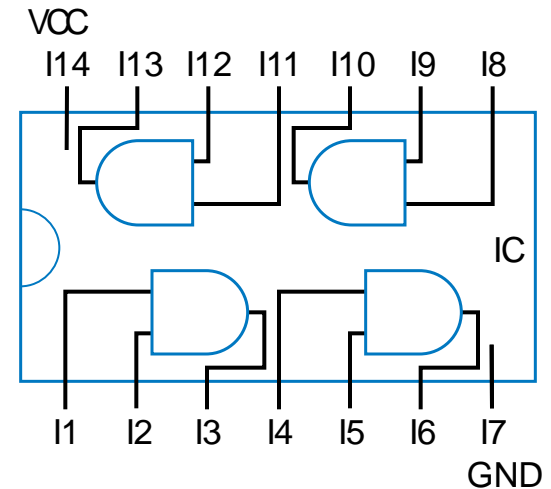


(b) Physical
implementation



Integrated Logic Circuits

- Off-the-shelf logic (SSI) IC
 - Logic IC has a few gates, connected to IC's pins
 - Known as Small Scale Integration (SSI)
 - Popular logic IC series: 7400
 - Originally developed 1960s
 - Back then, each IC cost \$1000
 - Today, costs just tens of cents



7400-Series Logic ICs

TABLE 7.1: Commonly used 7400-series ICs.

Part	Description	Pins
74LS00	Four 2-input NAND	14
74LS02	Four 2-input NOR	14
74LS04	Six inverters	14
74LS08	Four 2-input AND	14
74LS10	Three 3-input NAND	14
74LS11	Three 3-input AND	14
74LS14	Six inverters (Schmitt trigger)	14
74LS20	Two 4-input NAND	14
74LS27	Three 3-input NOR	14
74LS30	One 8-input NAND	14
74LS32	Four 2-input OR	14
74LS74	Two D flip-flop, positive edge triggered, with preset and reset	14
74LS83	4-bit binary full-adder	16
74LS85	4-bit magnitude comparator	16

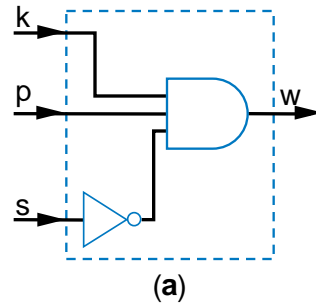
Source: www.digikey.com



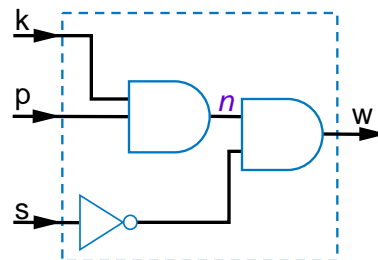
Using Logic ICs

- Example: Seat belt warning light using off-the-shelf 7400 ICs
 - Option 1: Use one 74LS08 IC having 2-input AND gates, and one 74LS04 IC having inverters

(a) Desired circuit

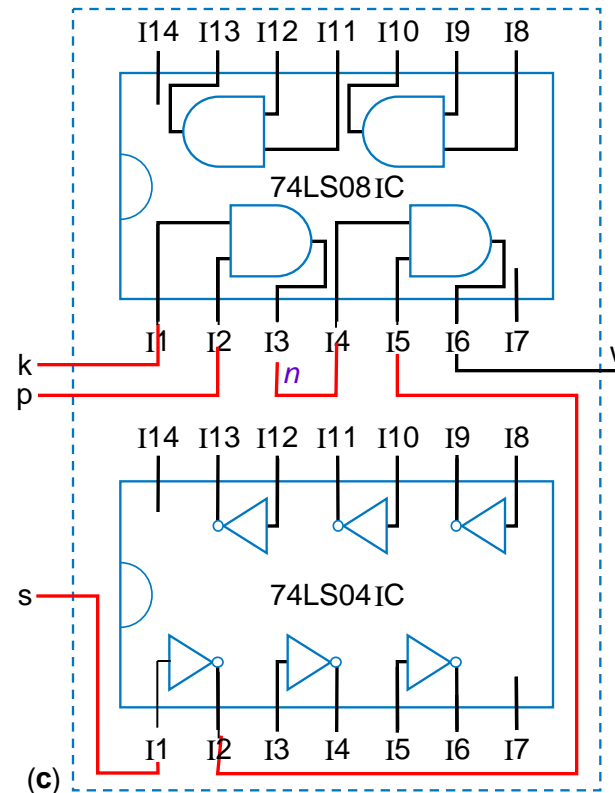


(a)



(b)

(b) Decompose into 2-input AND gates



(c) Connect ICs to create desired circuit

TABLE 7.1: Commonly used 7400-series ICs.

Part	Description	Pins
74LS00	Four 2-input NAND	14
74LS02	Four 2-input NOR	14
74LS04	Six inverters	14
74LS08	Four 2-input AND	14
74LS10	Three 3-input NAND	14
74LS11	Three 3-input AND	14
74LS14	Six inverters (Schmitt trigger)	14
74LS20	Two 4-input NAND	14
74LS27	Three 3-input NOR	14
74LS30	One 8-input NAND	14
74LS32	Four 2-input OR	14
74LS74	Two D flip-flop, positive edge triggered, with preset and reset	14
74LS83	4-bit binary full-adder	16
74LS85	4-bit magnitude comparator	16

Source: www.digikey.com



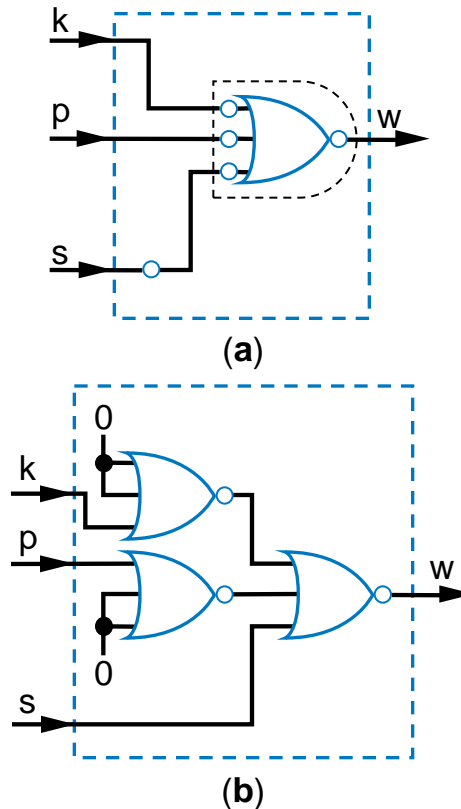
Using Logic ICs

- Example: Seat belt warning light using off-the-shelf 7400 ICs
 - Option 2: Use a single 74LS27 IC having 3-input NOR gates

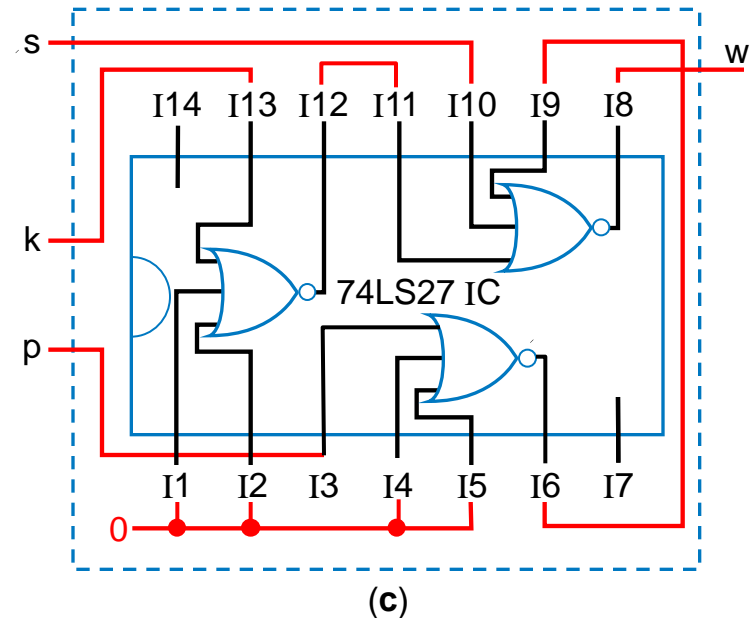
TABLE 7.1: Commonly used 7400-series ICs.

Part	Description	Pins
74LS00	Four 2-input NAND	14
74LS02	Four 2-input NOR	14
74LS04	Six inverters	14
74LS08	Four 2-input AND	14
74LS10	Three 3-input NAND	14
74LS11	Three 3-input AND	14
74LS14	Six inverters (Schmitt trigger)	14
74LS20	Two 4-input NAND	14
74LS27	Three 3-input NOR	14
74LS30	One 8-input NAND	14
74LS32	Four 2-input OR	14
74LS74	Two D flip-flop, positive edge triggered, with preset and reset	14
74LS83	4-bit binary full-adder	16
74LS85	4-bit magnitude comparator	16

Source: www.digikey.com



Converting to 3-input NOR gates

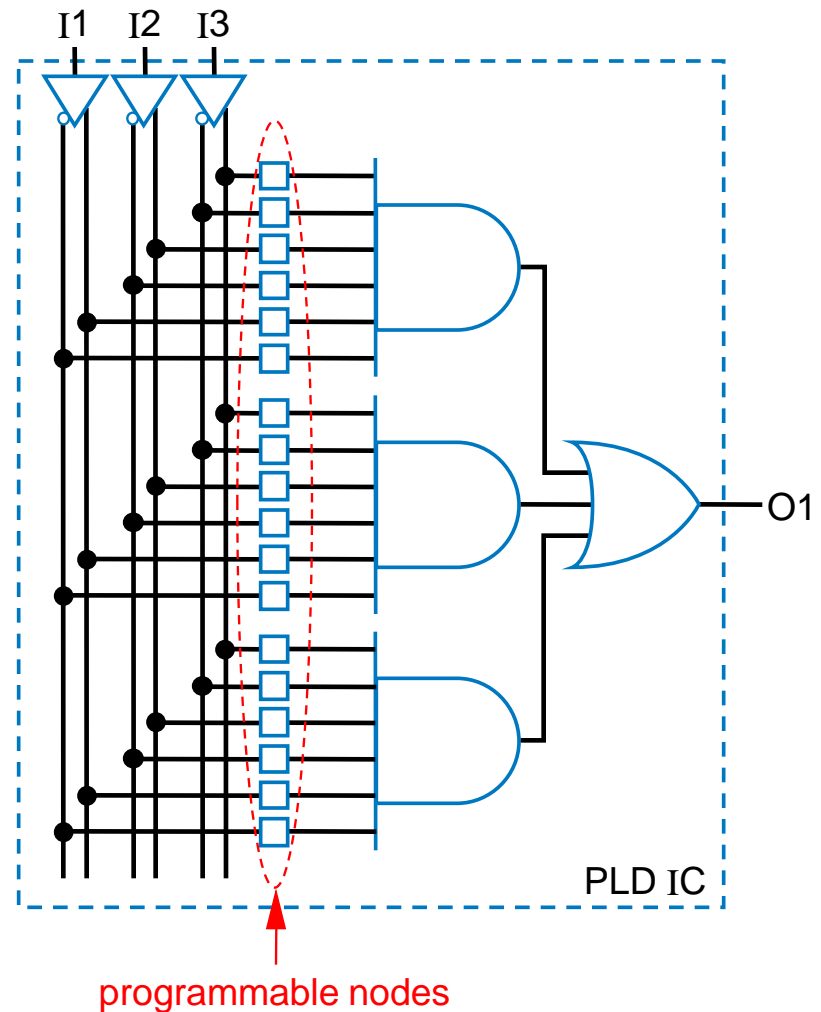


Connecting the pins to create the desired circuit



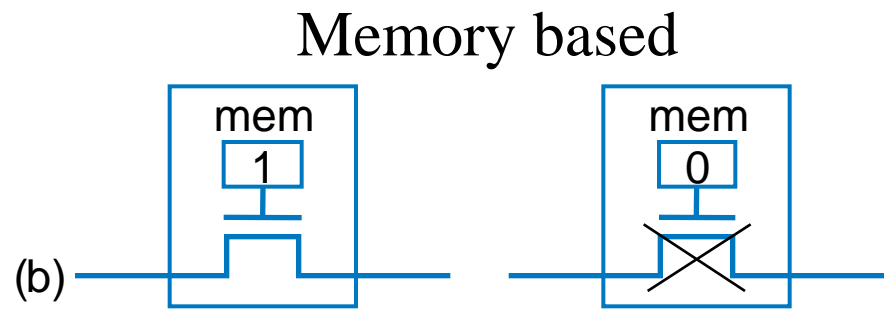
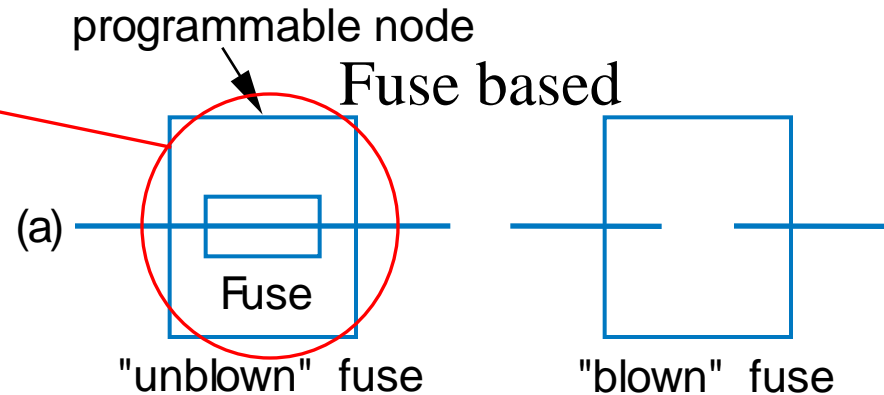
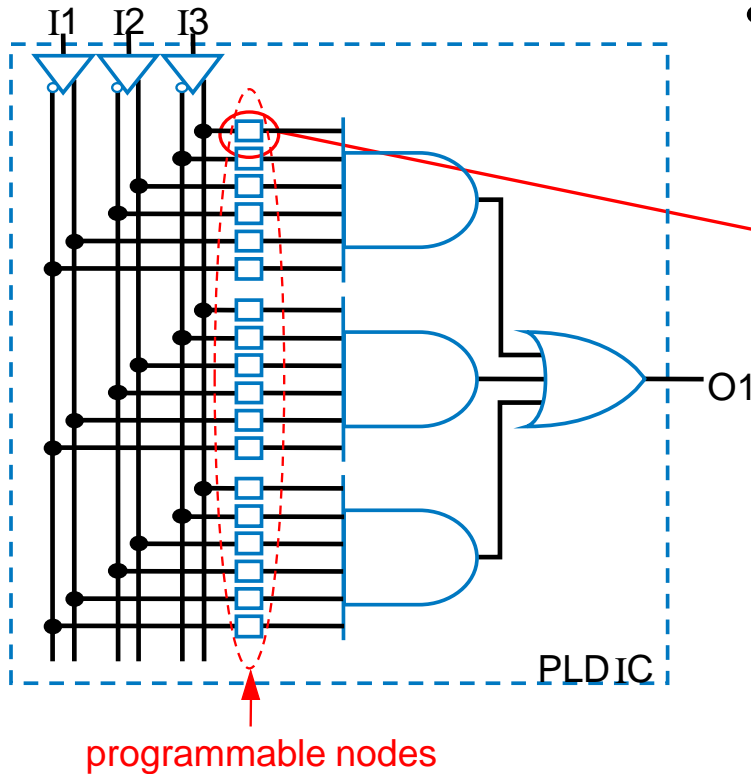
Other Technologies

- Simple Programmable Logic Devices (SPLDs)
 - Developed 1970s (thus, pre-dates FPGAs)
 - Prefabricated IC with large AND-OR structure
 - Connections can be "programmed" to create custom circuit
 - Circuit shown can implement any 3-input function of up to 3 terms
 - e.g., $F = abc + a'c'$



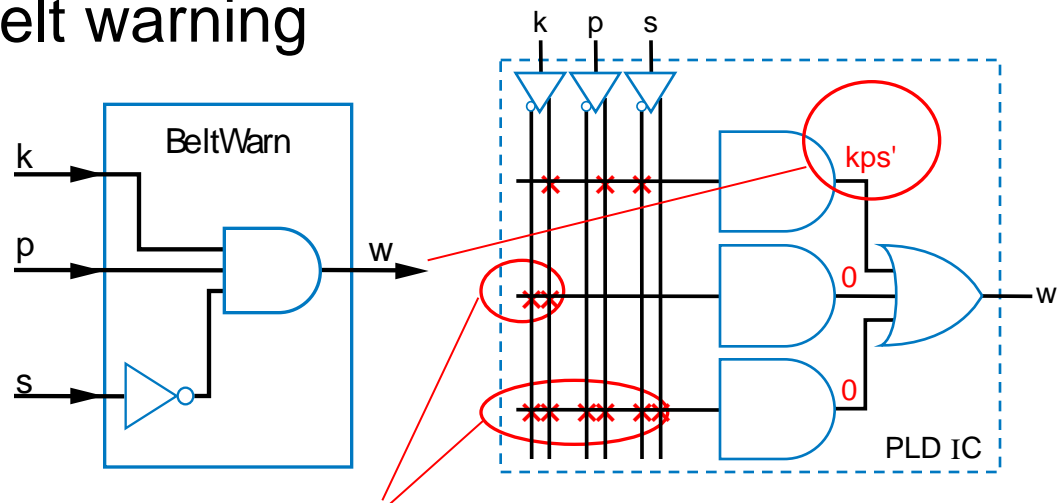
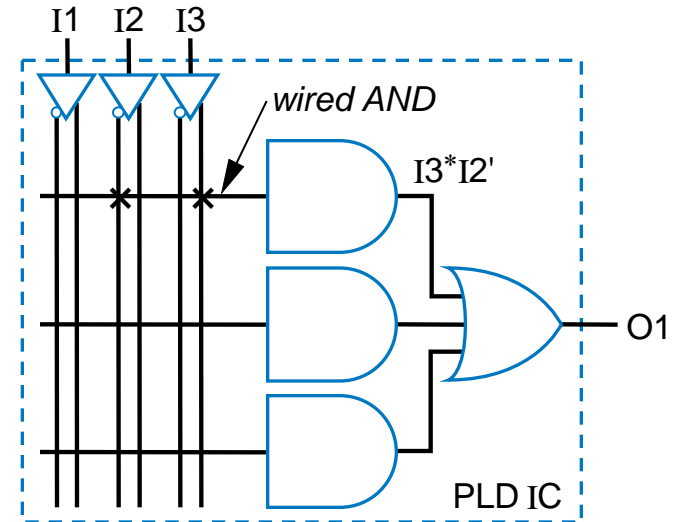
Programmable Nodes in an SPLD

- Fuse based – "blown" fuse removes connection
- Memory based – 1 creates connection



PLD Drawings and PLD Implementation Example

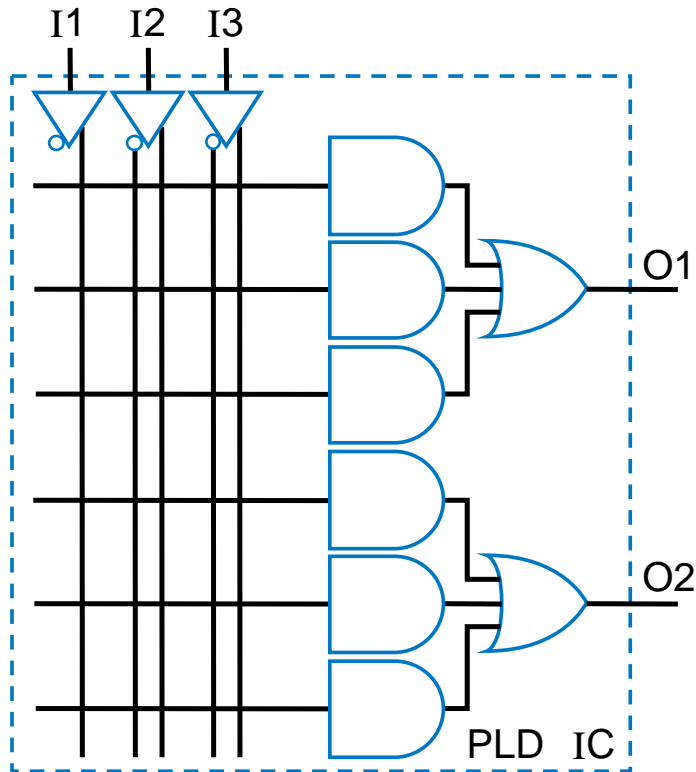
- Common way of drawing PLD connections:
 - Uses one wire to represent all inputs of an AND
 - Uses "x" to represent connection
 - Crossing wires are not connected unless "x" is present
- Example: Seat belt warning light using SPLD



Two ways to generate a 0 term

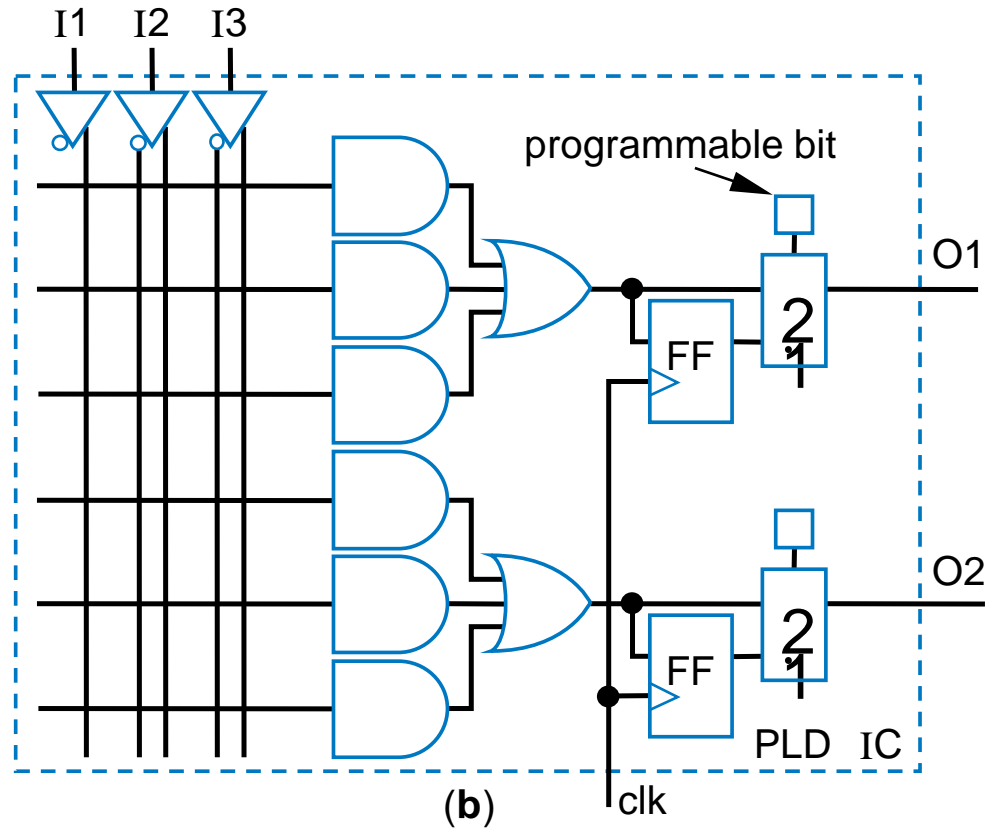


PLD Extensions



(a)

Two-output PLD



(b)

PLD with programmable registered outputs



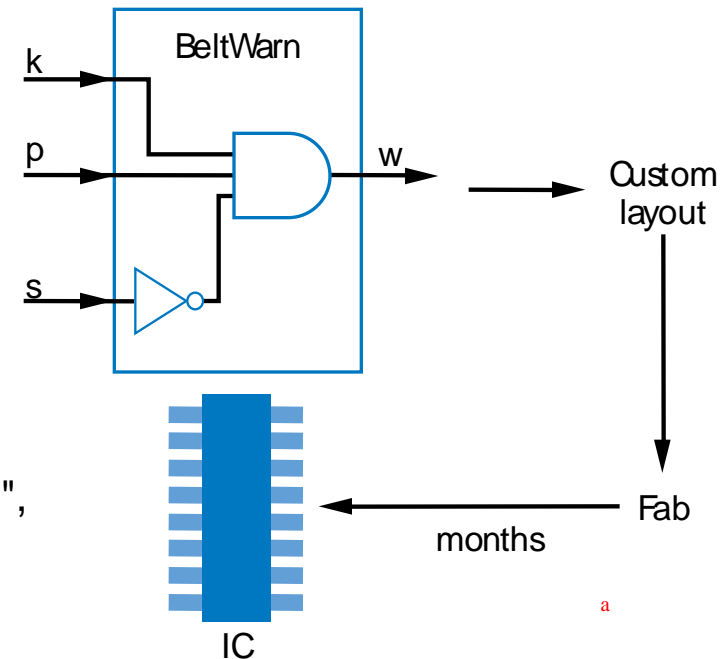
More on PLDs

- Originally (1970s) known as Programmable Logic Array – *PLA*
 - Had programmable AND and OR arrays
- AMD created "Programmable Array Logic" – "*PAL*" (trademark)
 - Only AND array was programmable (fuse based)
- Lattice Semiconductor Corp. created "Generic Array Logic – "*GAL*" (trademark)
 - Memory based
- As IC capacities increased, companies put multiple PLD structures on one chip, interconnecting them
 - Become known as Complex PLDs (CPLD), and older PLDs became known as Simple PLDs (SPLD)
- GENERALLY SPEAKING, difference of SPLDs vs. CPLDs vs. FPGAs:
 - SPLD: tens to hundreds of gates, and usually non-volatile (saves bits without power)
 - CPLD: thousands of gates, and usually non-volatile
 - FPGA: tens of thousands of gates and more, and usually volatile (but no reason why couldn't be non-volatile)



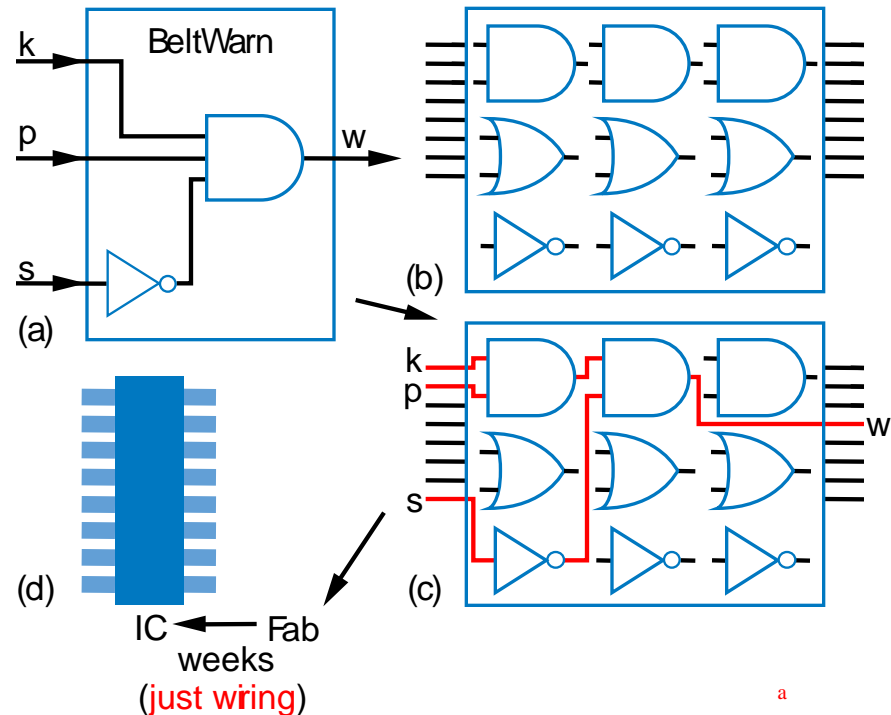
Manufactured IC Technologies

- We can manufacture our own IC
 - Months of time and millions of dollars
 - (1) Full-custom or (2) semicustom
- (1) Full-custom IC
 - We make a *full custom layout*
 - Using CAD tools
 - Layout describes the location and size of every transistor and wire
 - A *fab* (fabrication plant) builds IC for layout
 - Hard!
 - Fab setup costs ("non-recurring engineering", or NRE, costs) high
 - Error prone (several "respins")
 - Fairly uncommon
 - Reserved for special ICs that demand the very best performance or the very smallest size/power



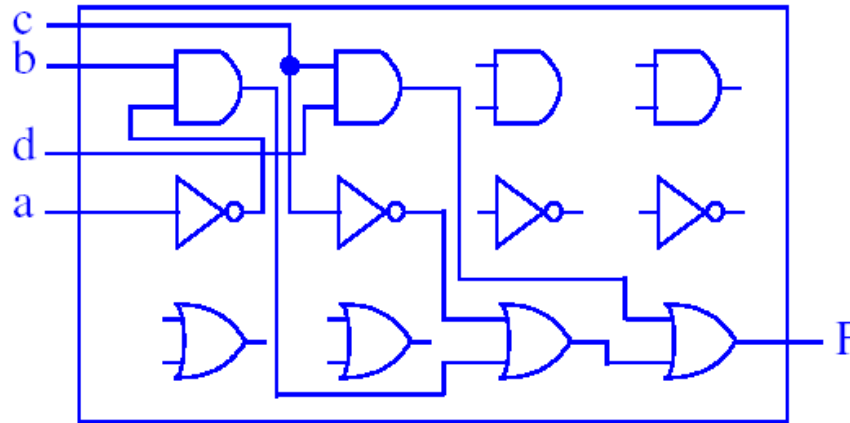
Manufactured IC Technologies – Gate Array ASIC

- (2) Semicustom IC
 - "Application-specific IC" (ASIC)
 - (a) Gate array or (b) standard cell
- (2a) Gate array
 - Series of gates already layed out on chip
 - We just wire them together
 - Using CAD tools
 - Vs. full-custom
 - Cheaper and quicker to design
 - But worse performance, size, power
 - Very popular



Gate Array Homework Example

- 7.5 Draw a gate array IC having three rows, the first row having four 2-input AND gates, the second row having four 2-input OR gates, and the third having row four NOT gates. Show how to instantiate wires to the gate array to implement the function $F(a, b, c, d) = a'b + cd + c'$.



Manufactured IC Technologies – Gate Array ASIC

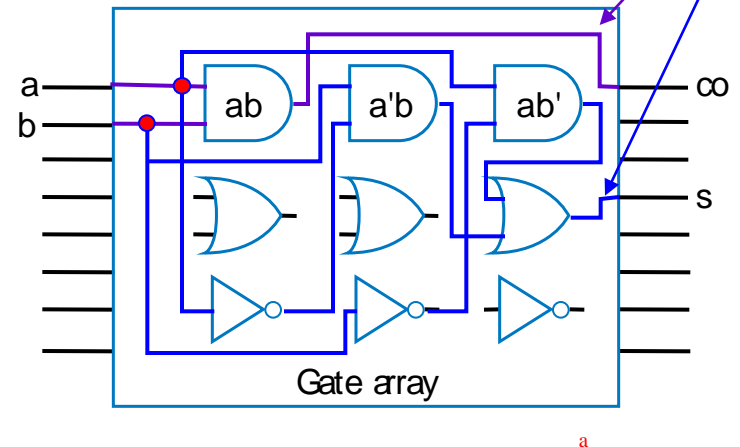
Function Table for half-adder

a	b	SUM	Carry Out
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Half-adder equations:

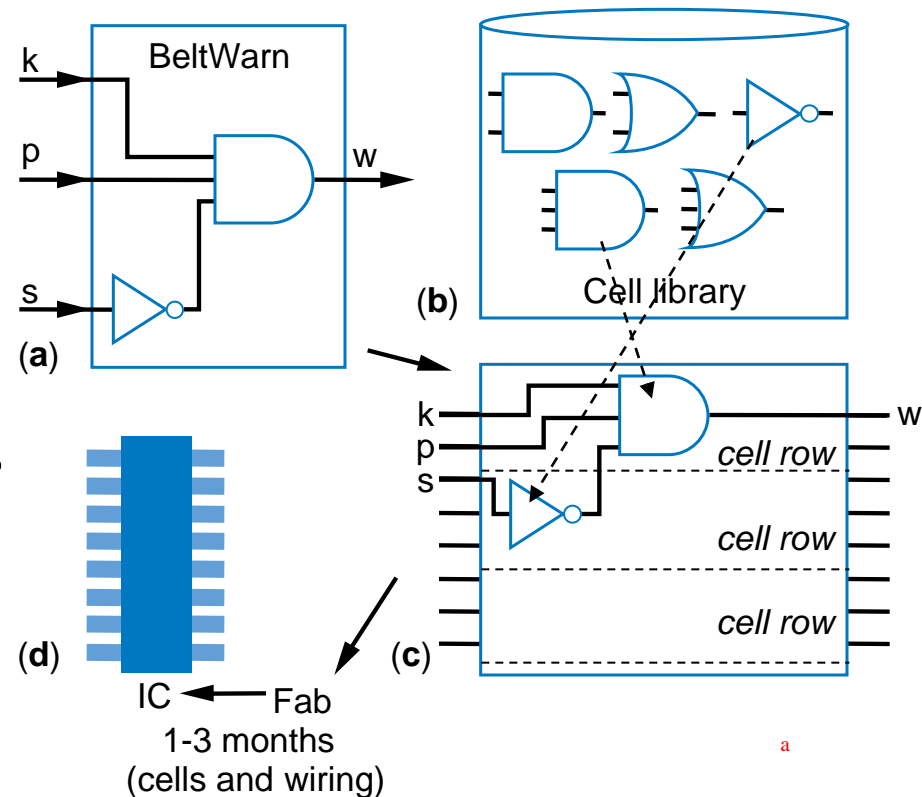
$$s = a'b + ab'$$

$$co = ab$$



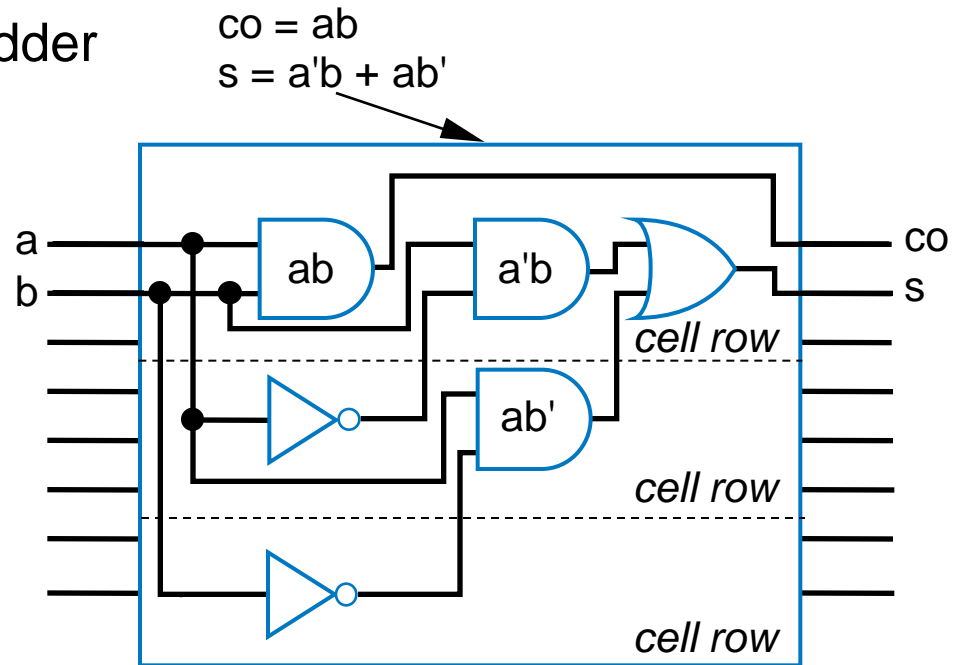
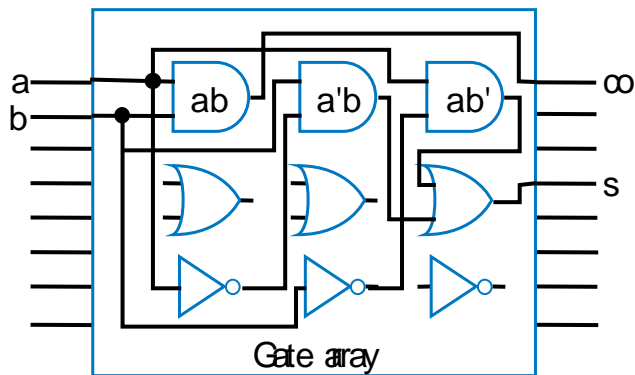
Manufactured IC Technologies – Standard Cell ASIC

- (2) Semicustom IC
 - "Application-specific IC" (ASIC)
 - (a) Gate array or (b) standard cell
- (2b) Standard cell
 - Pre-layed-out "cells" exist in library, not on chip
 - Designer instantiates cells into pre-defined rows, and connects
 - Vs. gate array
 - Better performance/power/size
 - A bit harder to design
 - Vs. full custom
 - Not as good of circuit, but still far easier to design



Manufactured IC Technologies – Standard Cell ASIC

- (2b) Standard cell
 - Example: Mapping a half-adder to standard cells

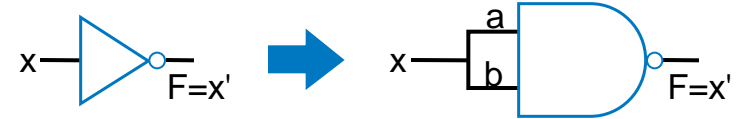
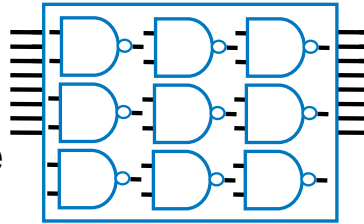


Notice fewer gates and shorter wires for standard cells versus gate array, but at cost of more design effort

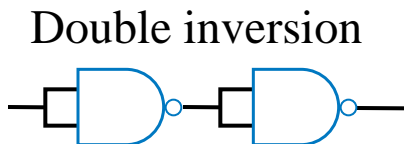
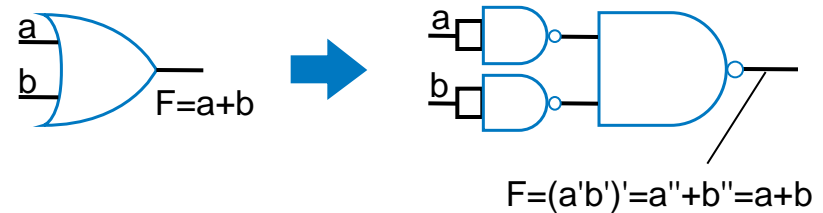
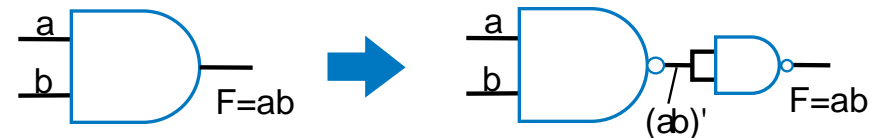


Implementing Circuits Using NAND Gates Only

- Gate array may have NAND gates only
 - NAND is universal gate
 - Any circuit can be mapped to NANDs only
- Convert AND/OR/NOT circuit to NAND-only circuit using mapping rules
 - After converting, remove double inversions

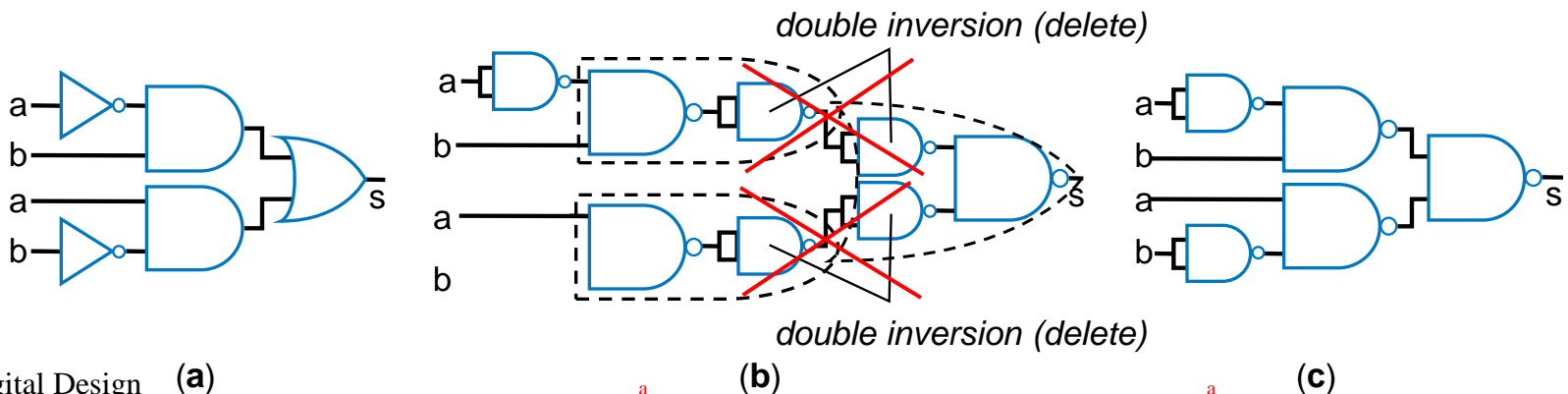
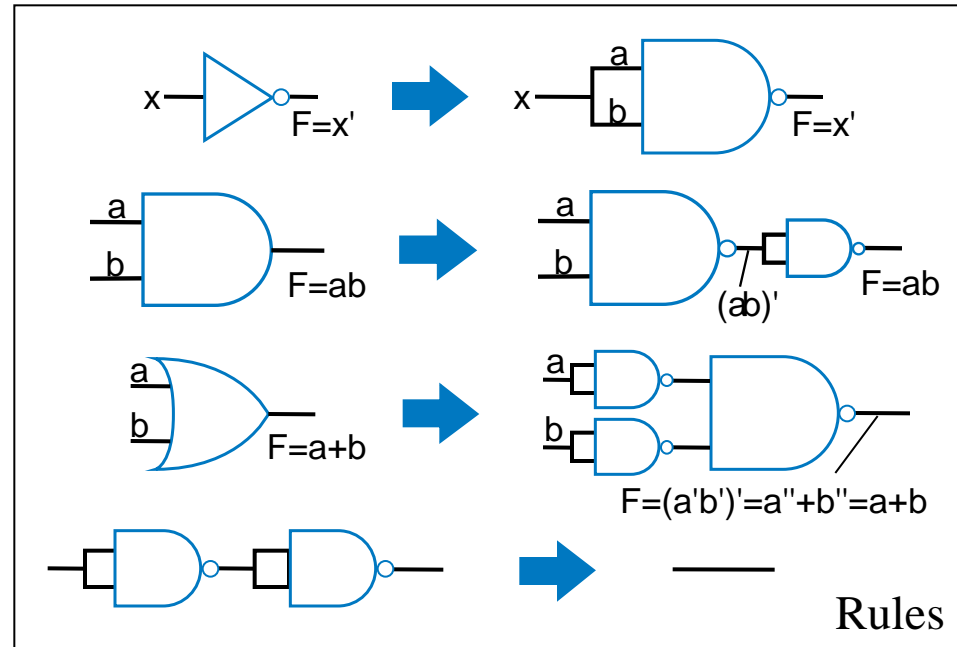


Inputs		Output
x	F	
0	1	
1	0	



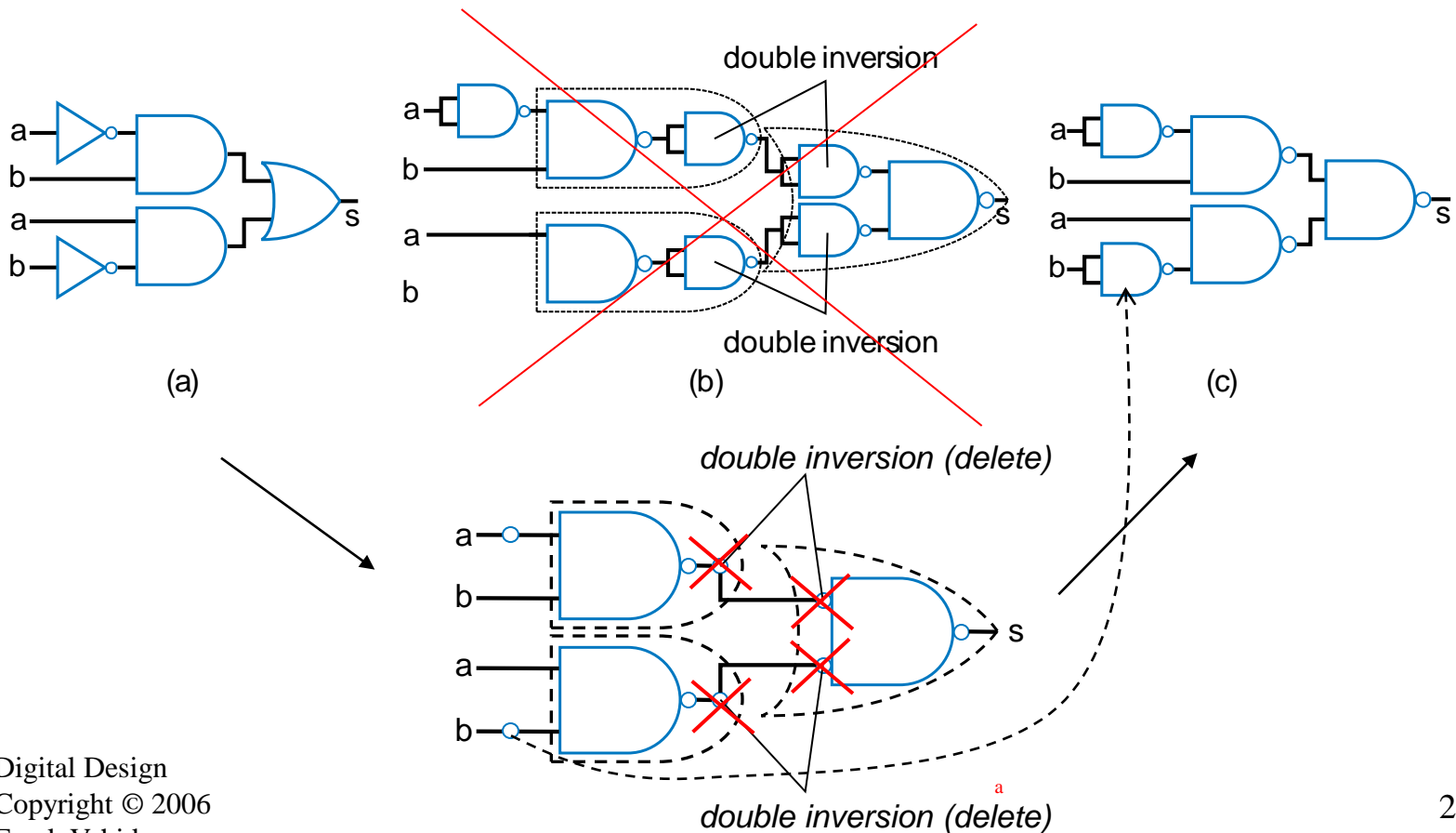
Implementing Circuits Using NAND Gates Only

- Example: Half-adder



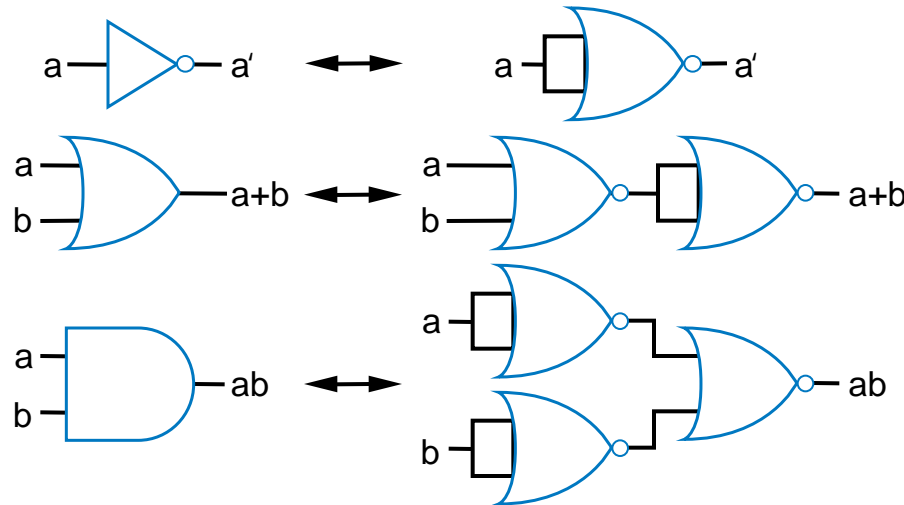
Implementing Circuits Using NAND Gates Only

- Shortcut when converting by hand
 - Use inversion bubbles rather than drawing inverters as 2-input NAND
 - Then remove double inversions as before



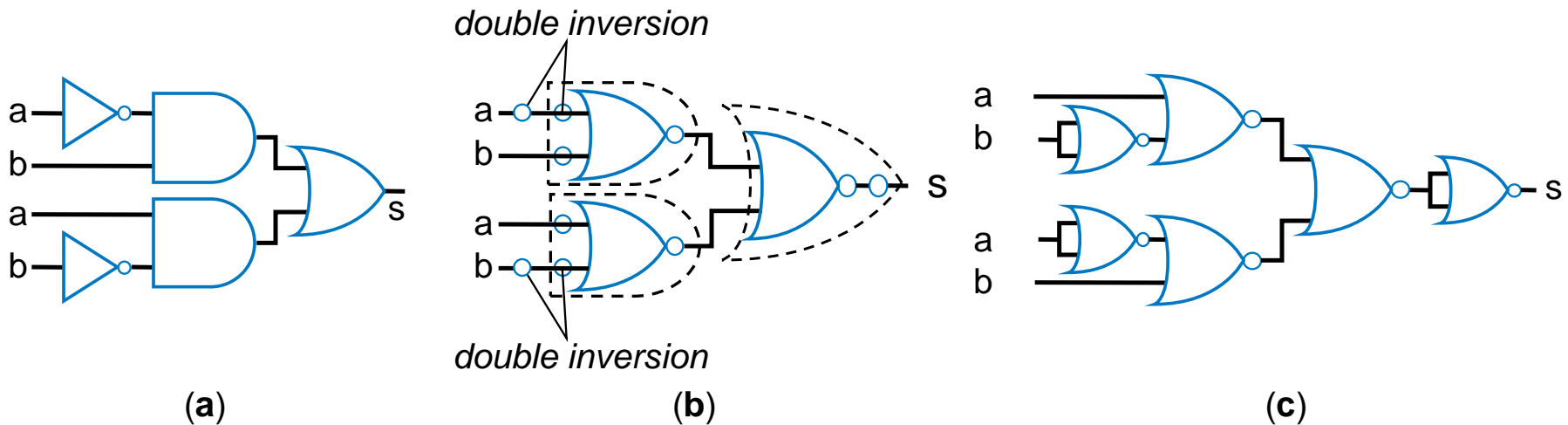
Implementing Circuits Using NOR Gates Only

- NOR gate is also universal
- Converting AND/OR/NOT to NOR is done using similar rules



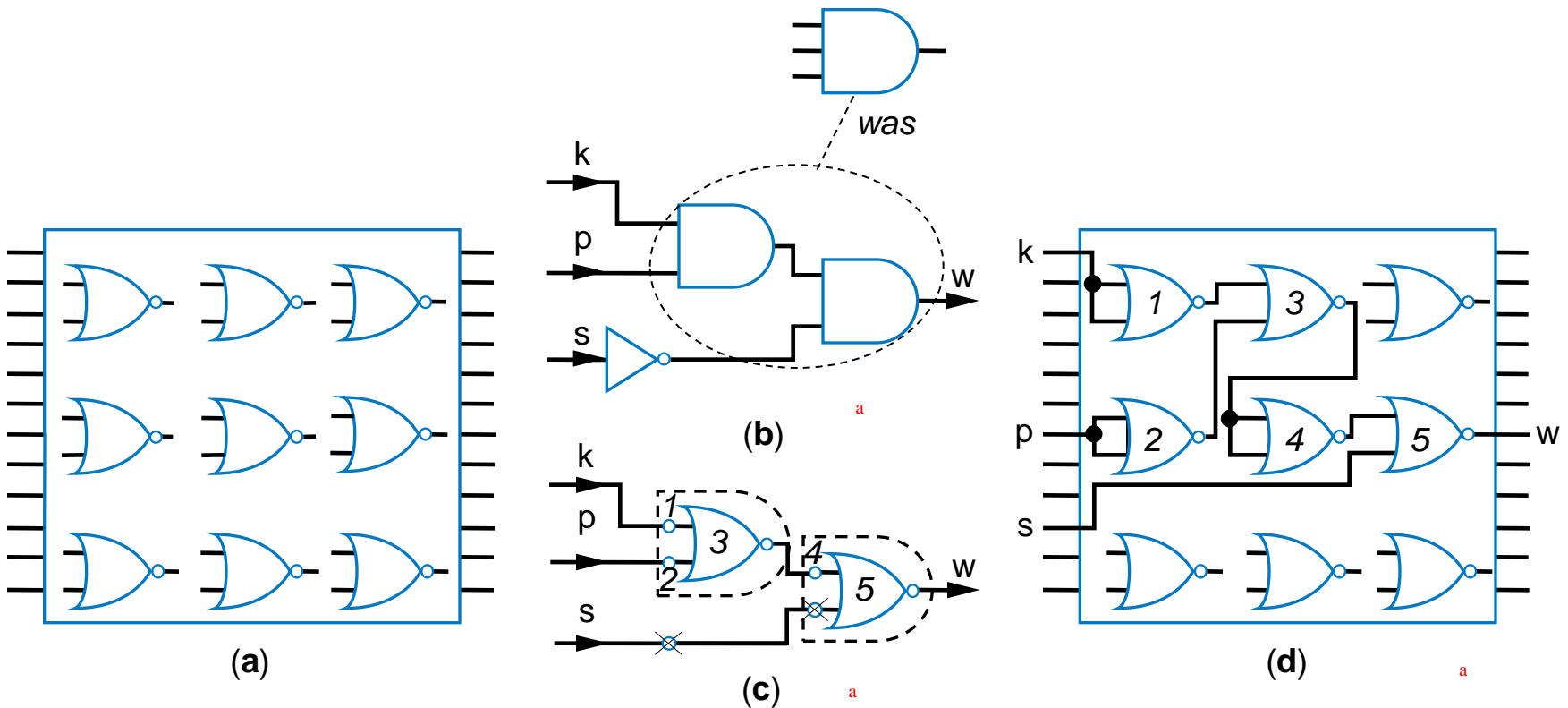
Implementing Circuits Using NOR Gates Only

- Example: Half adder



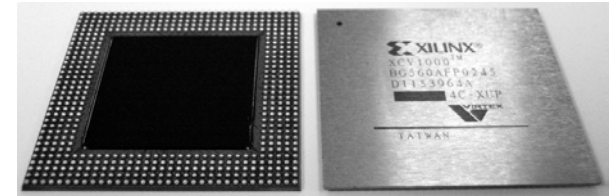
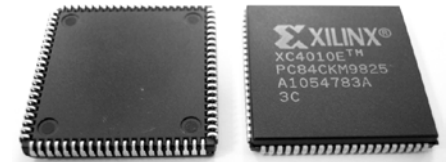
Implementing Circuits Using NOR Gates Only

- Example: Seat belt warning light on a NOR-based gate array
 - Note: if using 2-input NOR gates, first convert AND/OR gates to 2-inputs



Programmable IC Technology – FPGA

- Manufactured IC technologies require weeks to months to fabricate
 - And have large (hundred thousand to million dollar) initial costs
- Programmable ICs are pre-manufactured
 - Can implement circuit *today*
 - Just download bits into device
 - Slower/bigger/more-power than manufactured ICs
 - But get it today, and no fabrication costs
- Popular programmable IC – FPGA
 - "Field-programmable gate array"
 - Developed late 1980s
 - Though no "gate array" inside
 - Named when gate arrays were very popular in the 1980s
 - Programmable in seconds

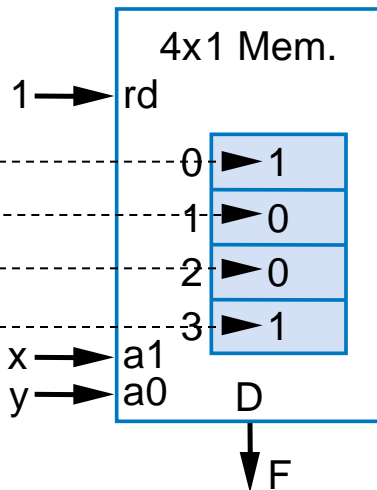


FPGA Internals: Lookup Tables (LUTs)

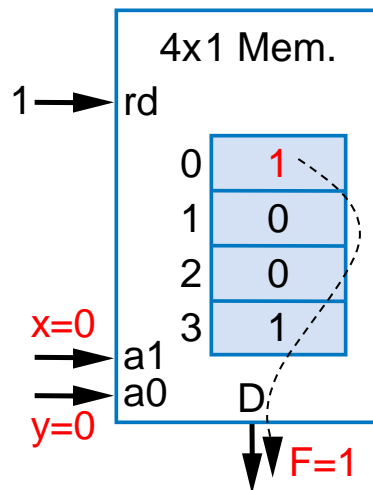
- Basic idea: Memory can implement combinational logic
 - e.g., 2-address memory can implement 2-input logic
 - 1-bit wide memory – 1 function; 2-bits wide – 2 functions
- Such memory in FPGA known as Lookup Table (LUT)

$$F = x'y' + xy$$

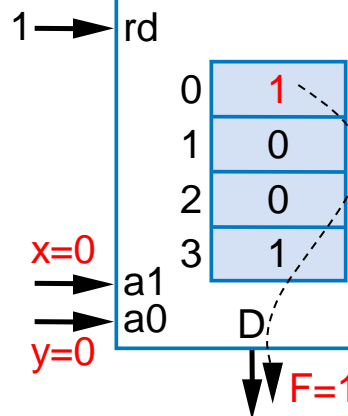
x	y	F
0	0	1
0	1	0
1	0	0
1	1	1



(a)



(b)

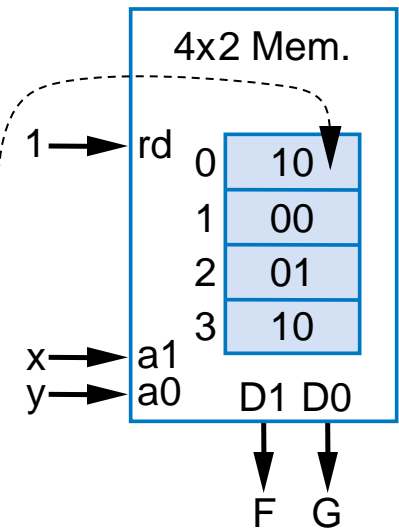


(c)

$$F = x'y' + xy$$

$$G = xy'$$

x	y	F	G
0	0	1	0
0	1	0	0
1	0	0	1
1	1	1	0



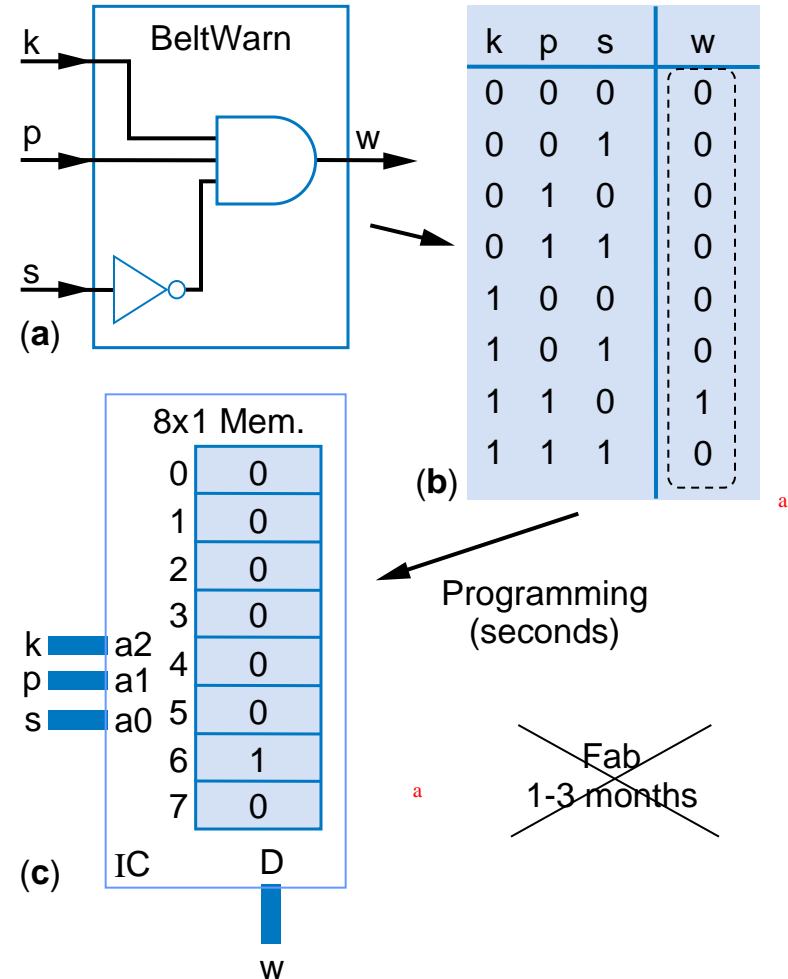
(d)

(e)



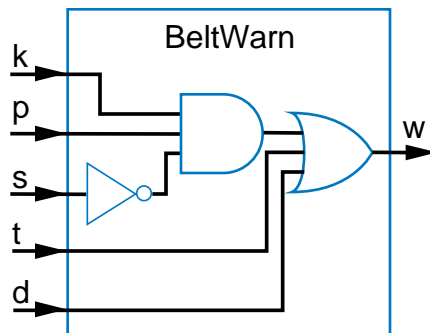
FPGA Internals: Lookup Tables (LUTs)

- Example: Seat-belt warning light (again)



FPGA Internals: Lookup Tables (LUTs)

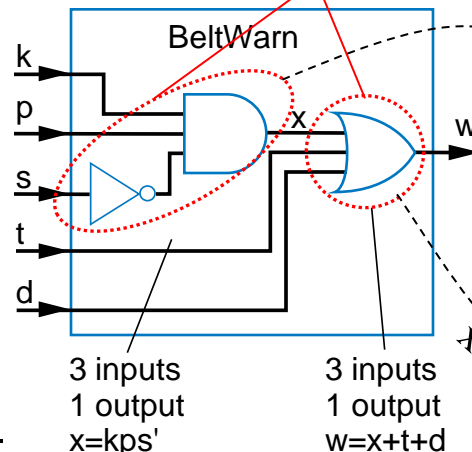
- Lookup tables become inefficient for more inputs
 - 3 inputs \rightarrow only 8 words
 - 8 inputs \rightarrow 256 words; 16 inputs \rightarrow 65,536 words!
- FPGAs thus have numerous small (3, 4, 5, or even 6-input) LUTs
 - If circuit has more inputs, must partition circuit among LUTs
 - Example: Extended seat-belt warning light system:



(a)

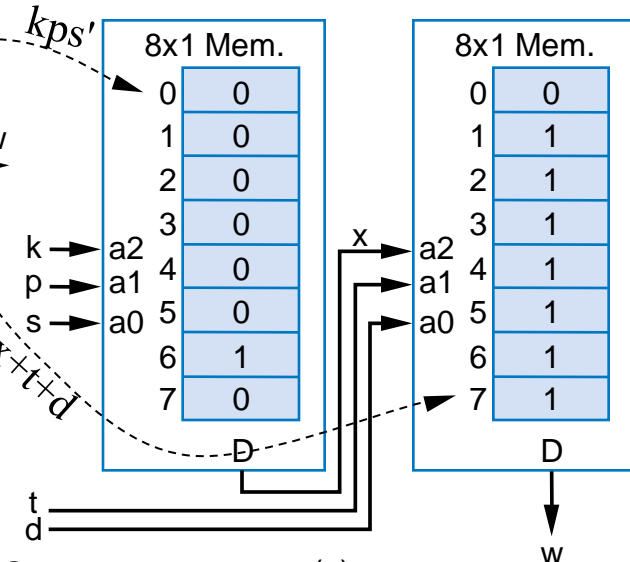
5-input circuit, but 3-input LUTs available

Sub-circuits have only 3-inputs each



(b)

Partition circuit into 3-input sub-circuits



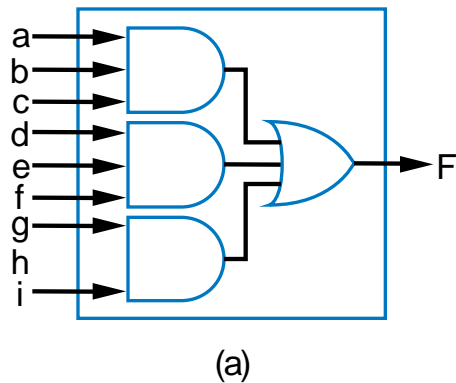
(c)

Map to 3-input LUTs

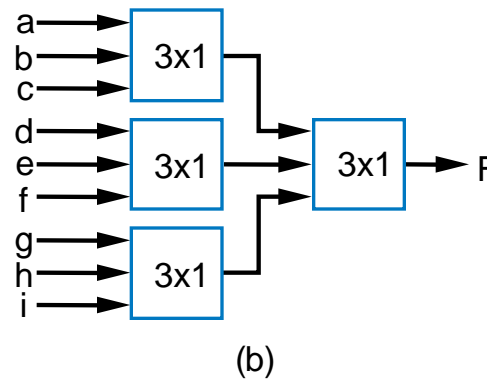


FPGA Internals: Lookup Tables (LUTs)

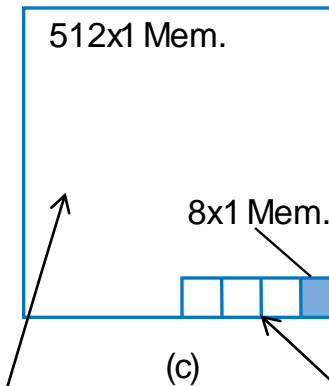
- Partitioning among smaller LUTs is more size efficient
 - Example: 9-input circuit



Original 9-input circuit



Partitioned among
3x1 LUTs

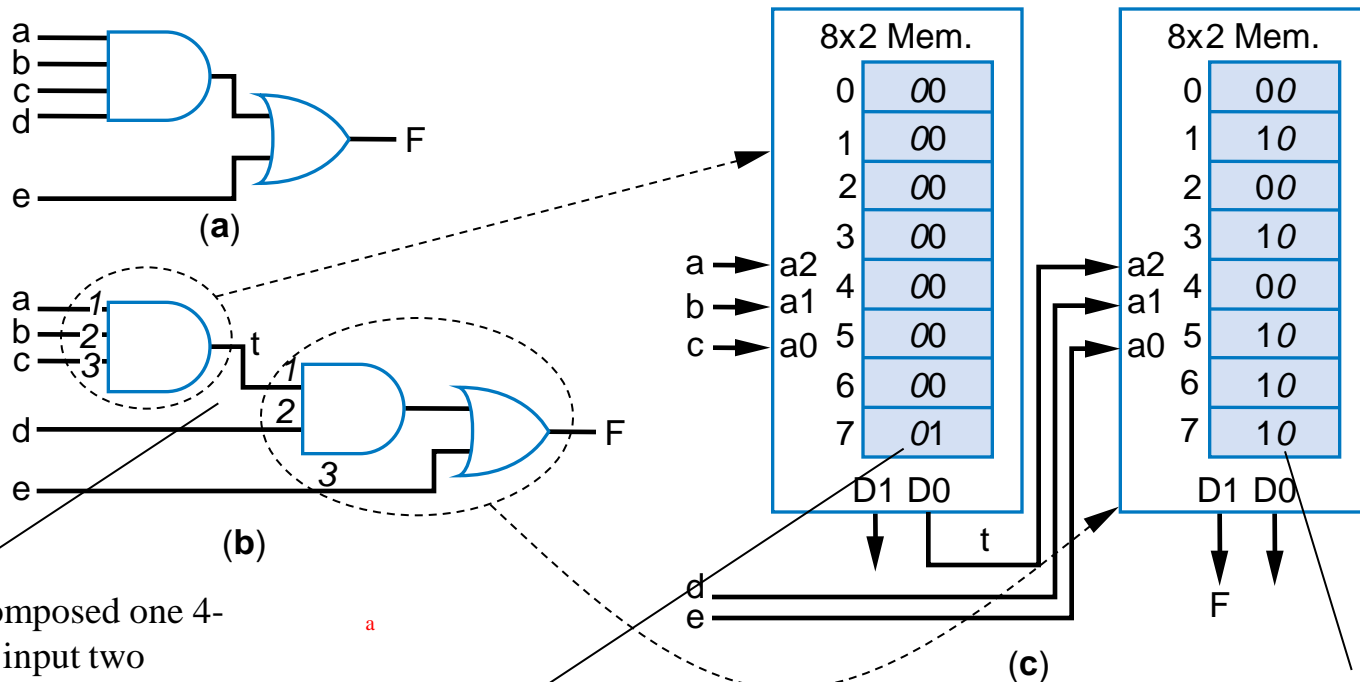


Requires only 4
3-input LUTs
(8x1 memories) –
much smaller than
a 9-input LUT
(512x1 memory)



FPGA Internals: Lookup Tables (LUTs)

- LUT typically has 2 (or more) outputs, not just one
- Example: Partitioning a circuit among 3-input 2-output lookup tables



(Note: decomposed one 4-input AND into two smaller ANDs to enable partitioning into 3-input sub-circuits)

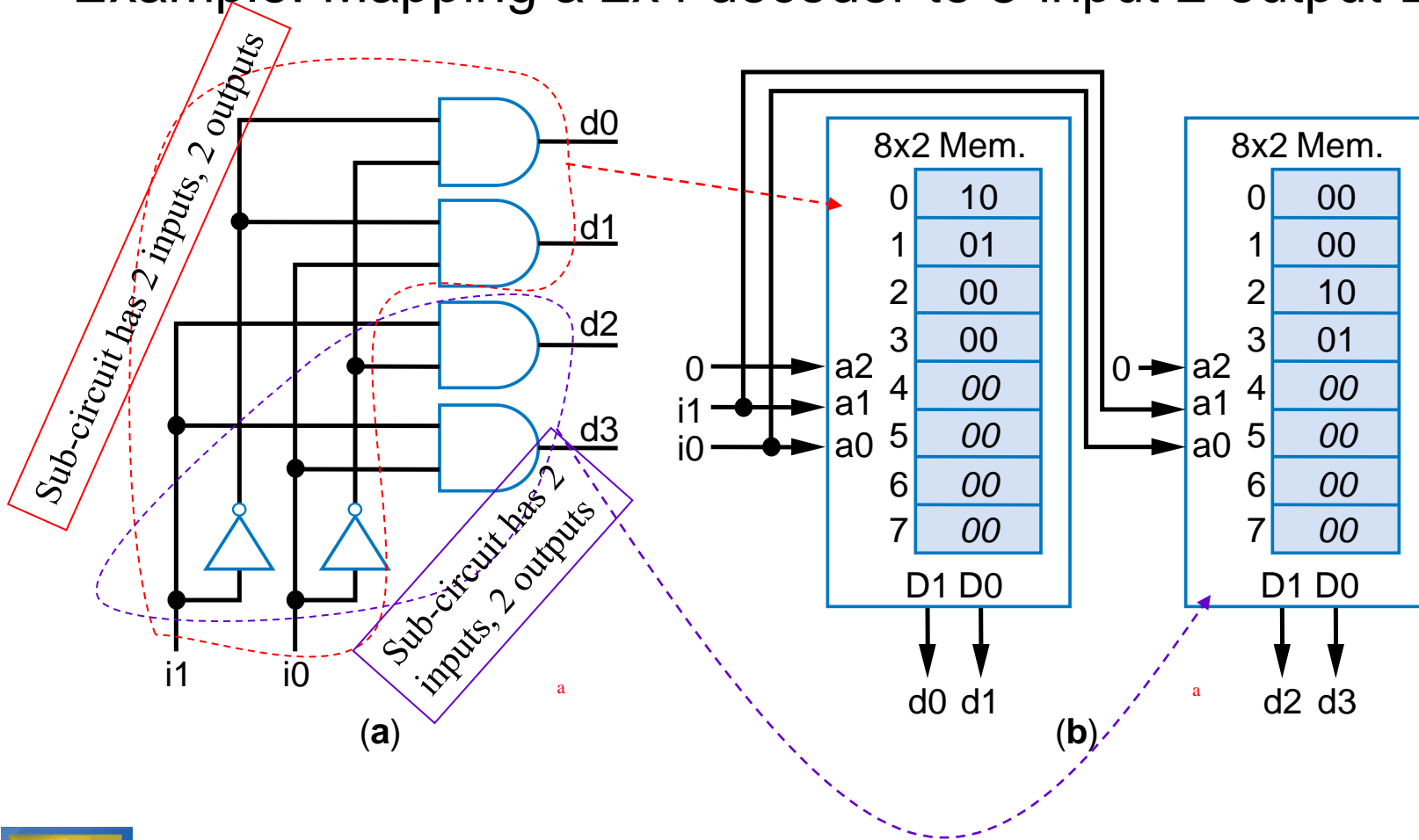
First column unused;
second column
implements AND

Second column unused;
first column implements
AND/OR sub-circuit



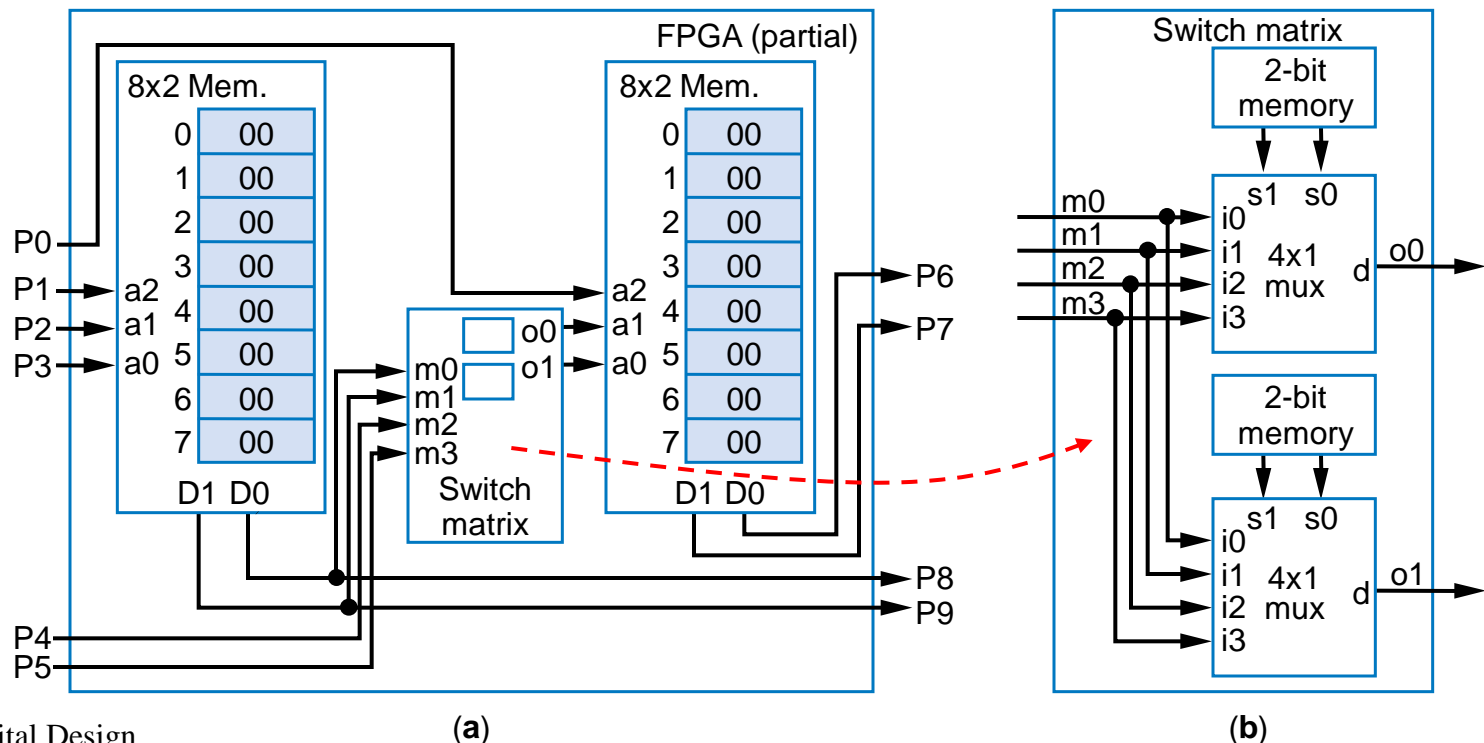
FPGA Internals: Lookup Tables (LUTs)

- Example: Mapping a 2x4 decoder to 3-input 2-output LUTs



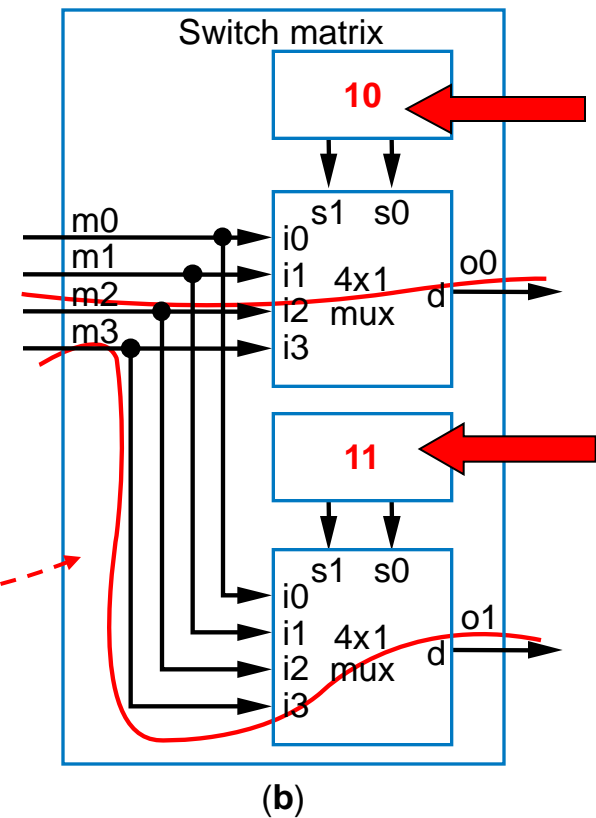
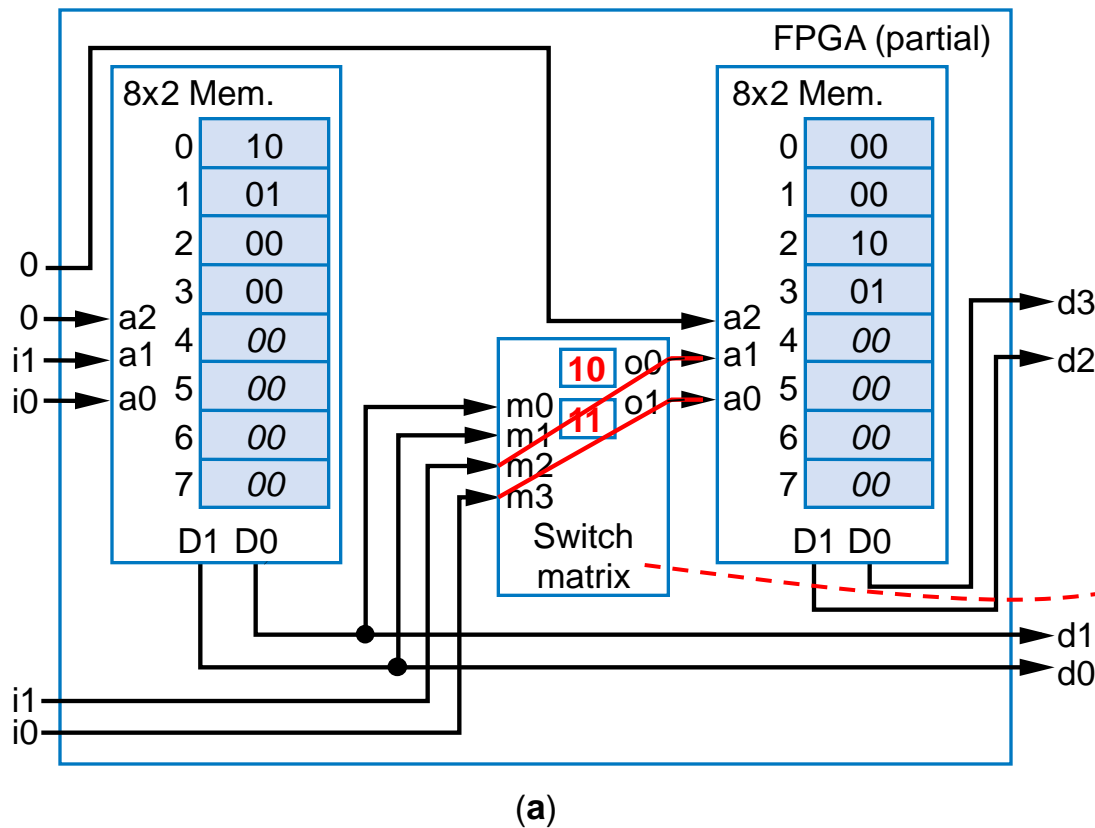
FPGA Internals: Switch Matrices

- Previous slides had hardwired connections between LUTs
- Instead, want to program the connections too
- Use switch matrices (also known as programmable interconnect)
 - Simple mux-based version – each output can be set to any of the four inputs just by programming its 2-bit configuration memory



FPGA Internals: Switch Matrices

- Mapping a 2x4 decoder onto an FPGA with a switch matrix

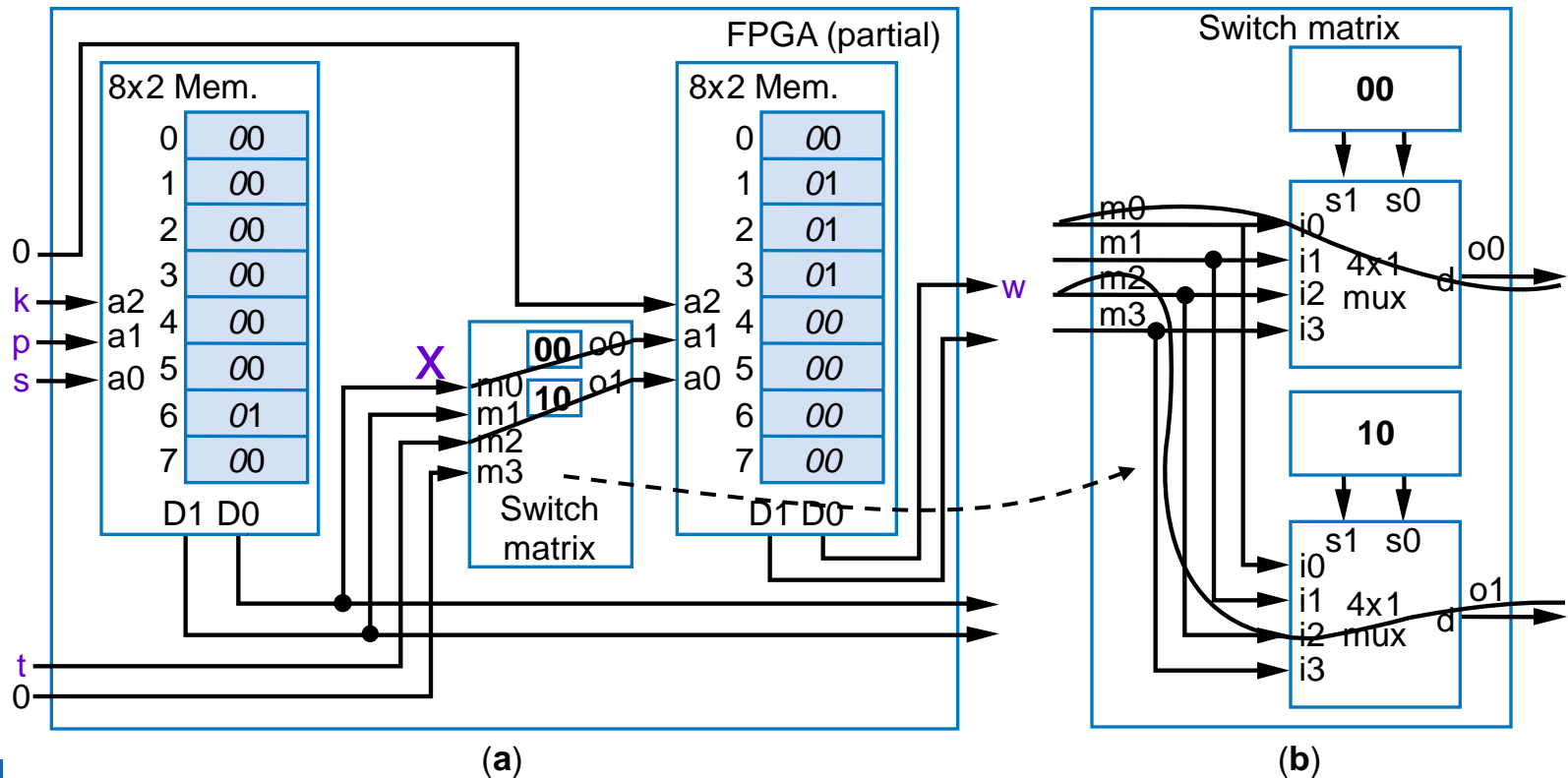
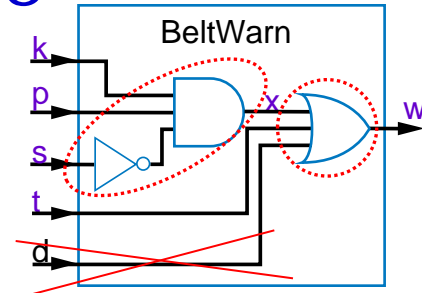


These bits establish the desired connections



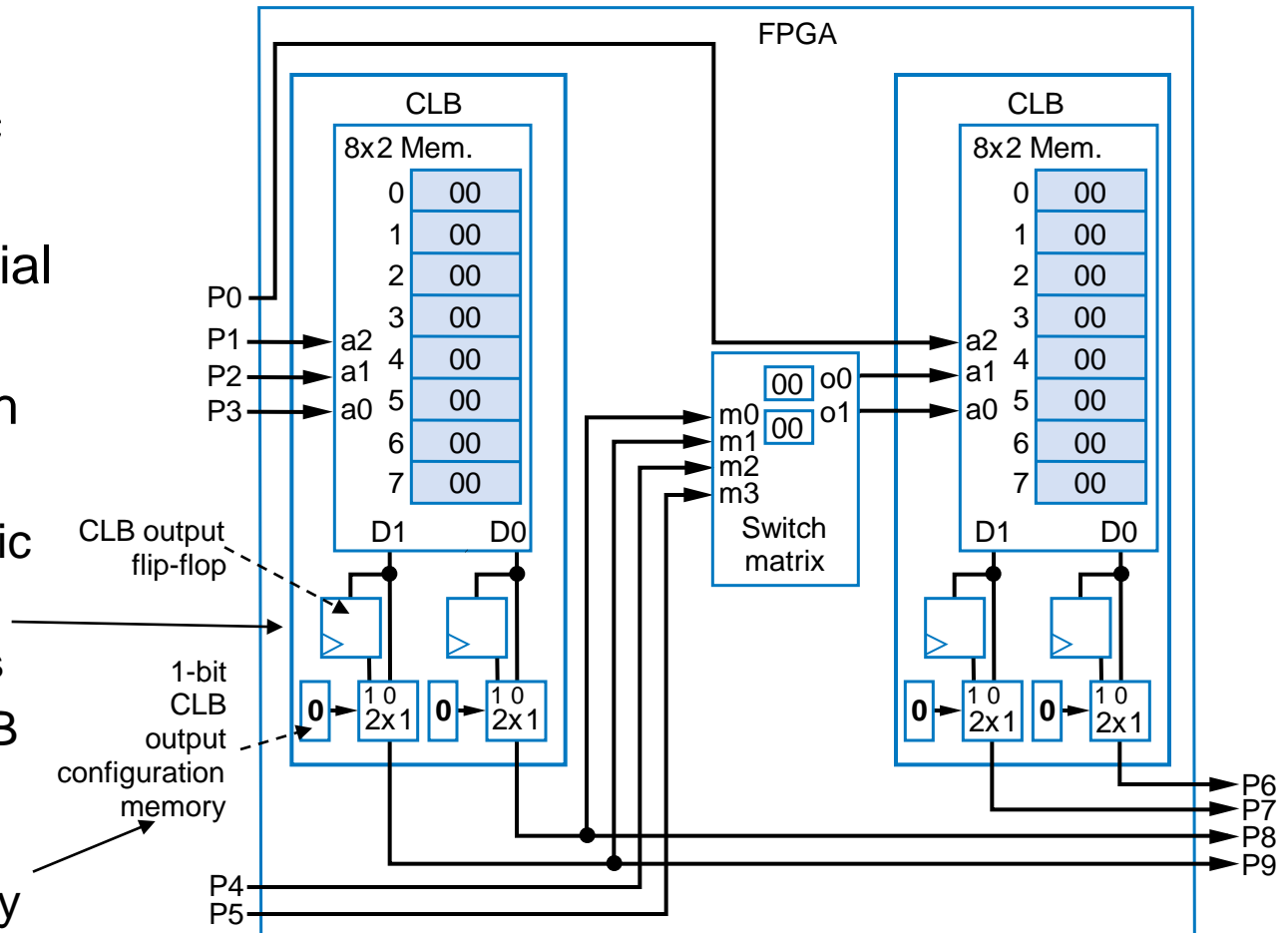
FPGA Internals: Switch Matrices

- Mapping the extended seatbelt warning light onto an FPGA with a switch matrix
 - Recall earlier example (let's ignore d input for simplicity)

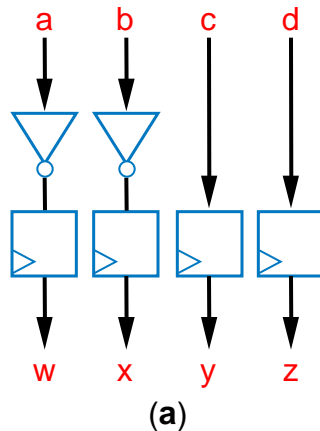


FPGA Internals: Configurable Logic Blocks (CLBs)

- LUTs can only implement combinational logic
- Need flip-flops to implement sequential logic
- Add flip-flop to each LUT output
 - Configurable Logic Block (CLB)
 - LUT + flip-flops
 - Can program CLB outputs to come from flip-flops or from LUTs directly



FPGA Internals: Sequential Circuit Example using CLBs

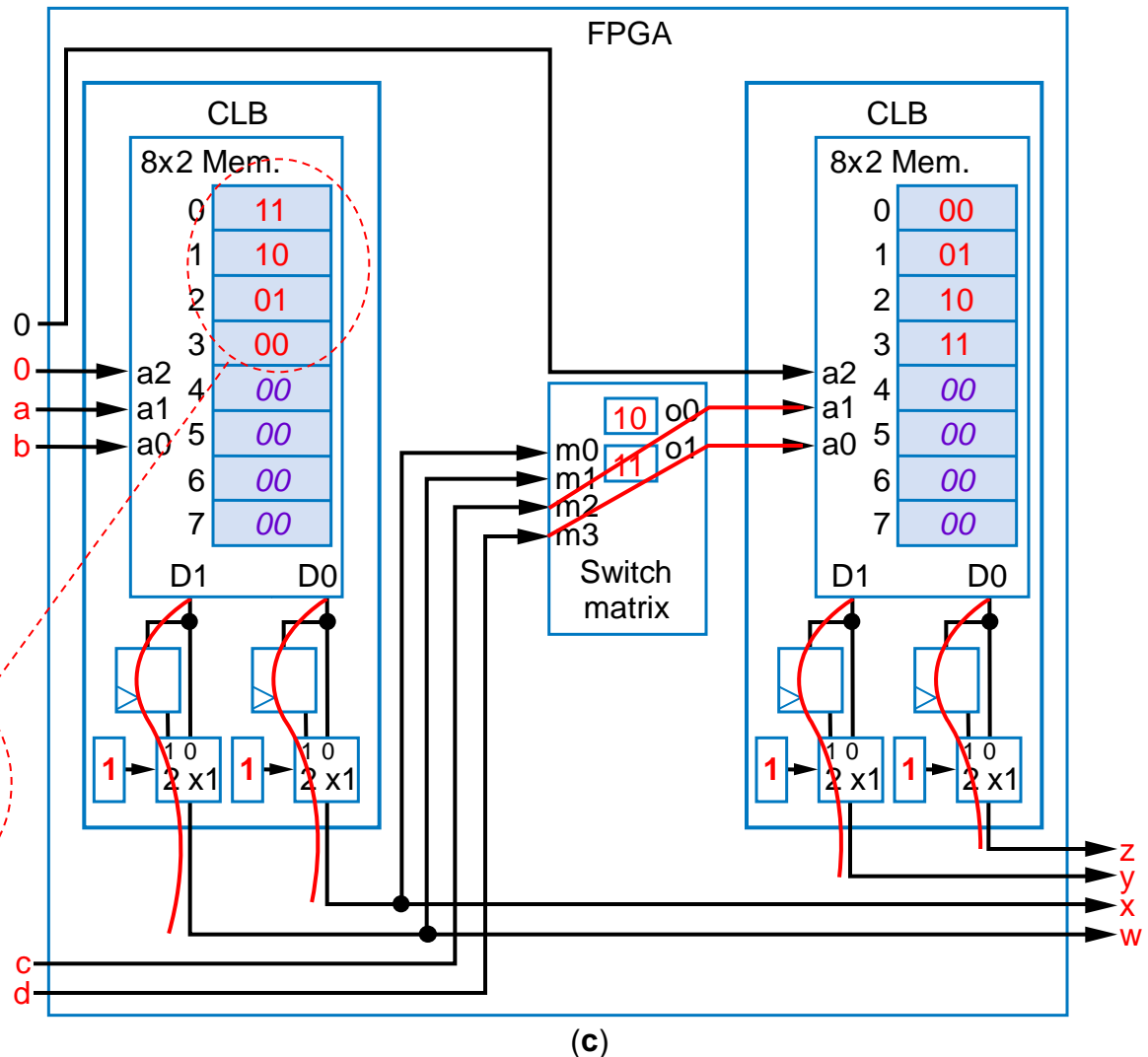


Left lookup table

a2	a1	a0	D1	D0
0	a	b	$w=a'$ $x=b'$	
0	0	0	1	1
0	0	1	1	0
0	1	0	0	1
0	1	1	0	0

below unused

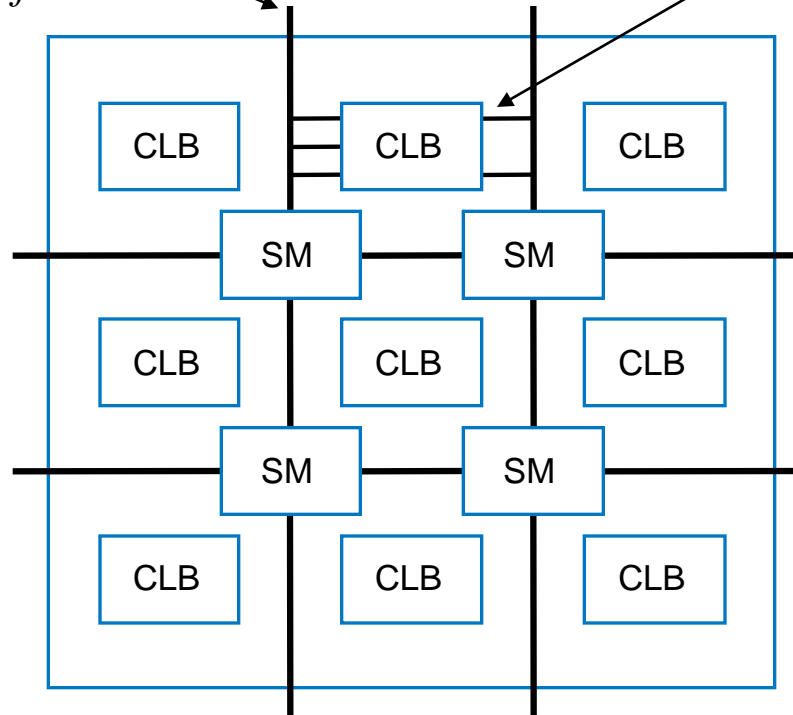
(b)



FPGA Internals: Overall Architecture

- Consists of hundreds or thousands of CLBs and switch matrices (SMs) arranged in regular pattern on a chip

*Represents channel with
tens of wires*

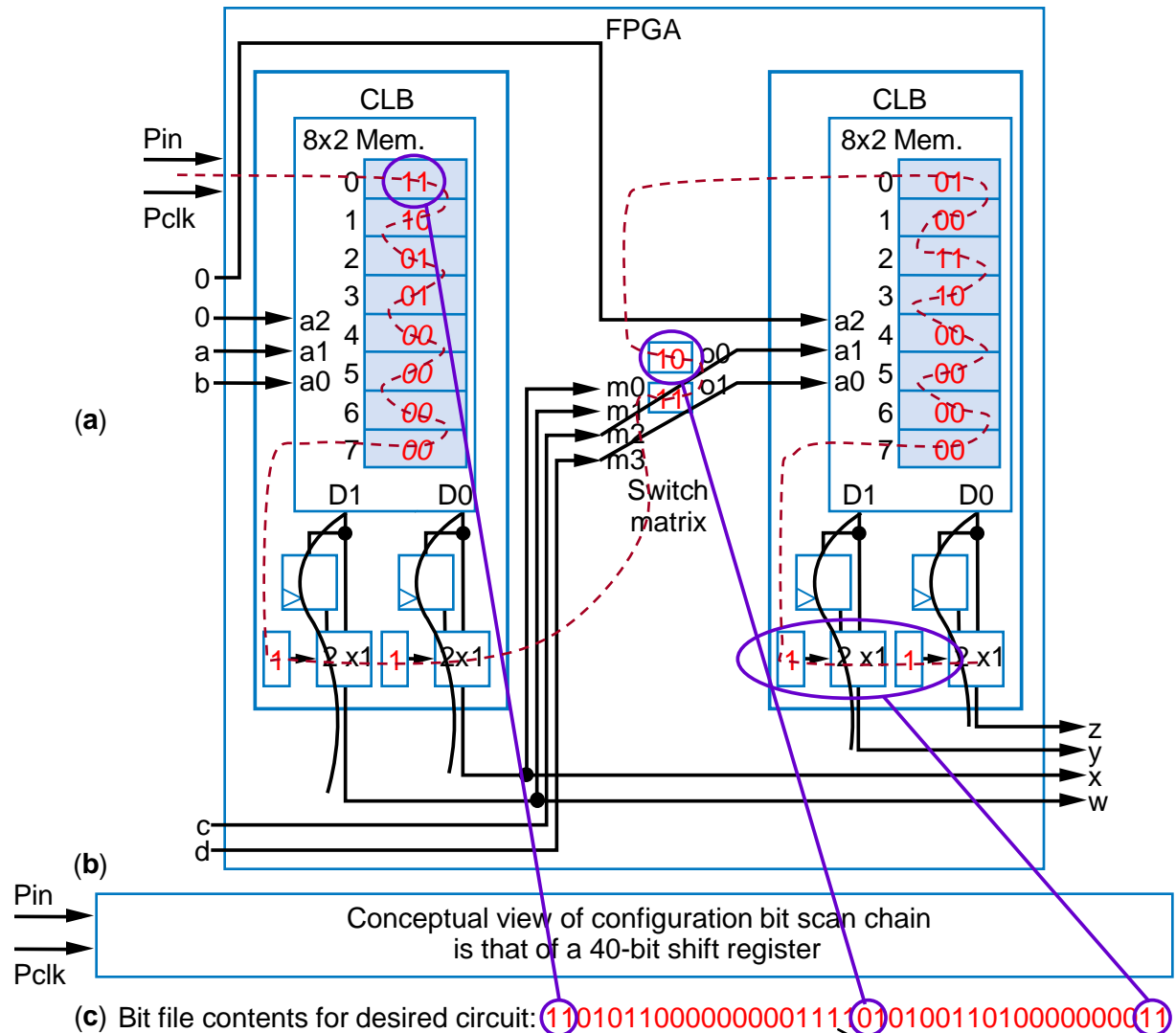


*Connections for just one
CLB shown, but all
CLBs are obviously
connected to channels*



FPGA Internals: Programming an FPGA

- All configuration memory bits are connected as one big shift register
 - Known as scan chain
- Shift in "bit file" of desired circuit



a

This isn't wrong. Although the bits appear as "10" above, note that the scan chain passes through those bits from right to left – so "01" is correct here.

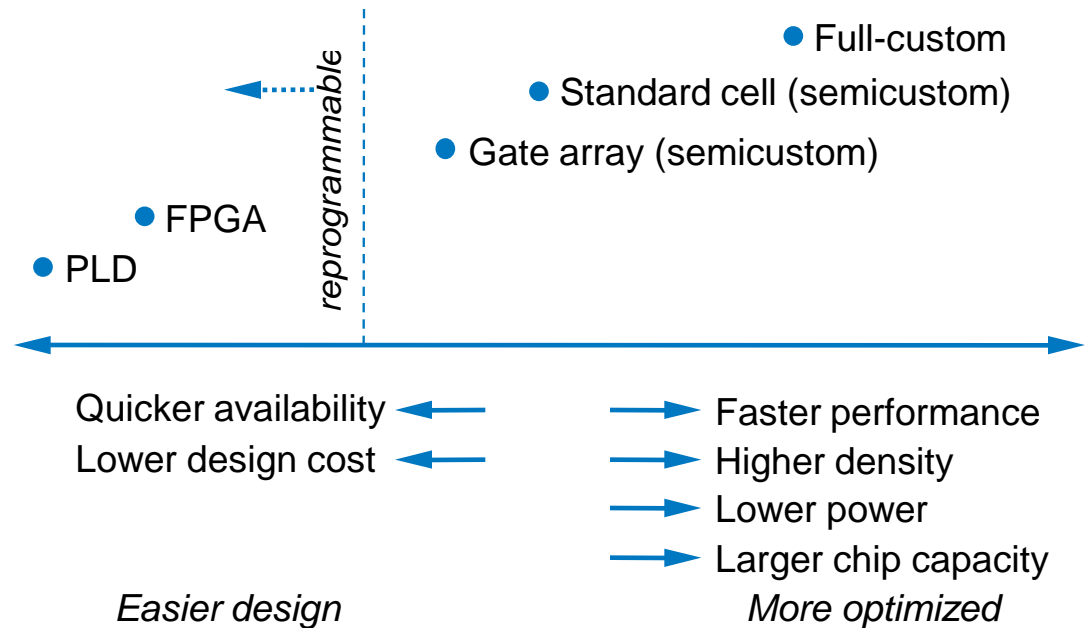


Technology Comparisons

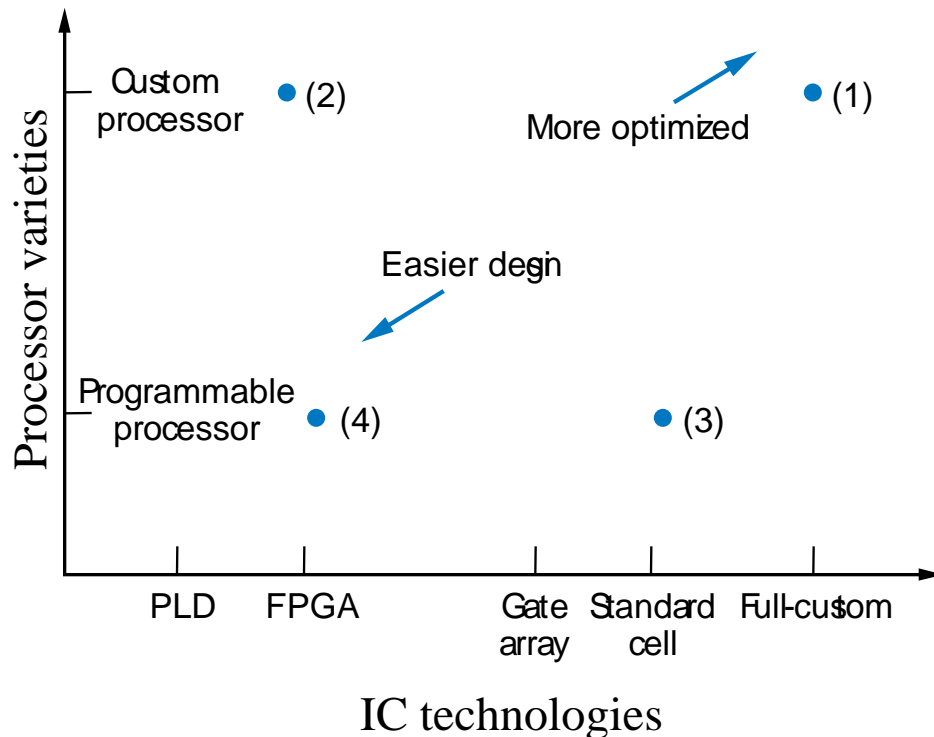
TABLE 7.2: Sample % of new implementations in various technologies. Total is more than 100% due to overlap among categories.

Technology	%
Standard cell	55%
Gate array	5%
System-on-a-Chip	30%
Full-custom	10%
CPLD/FPGA	10%
Other	5%

Source: Synopsys, DAC 2002 panel.



Technology Comparisons



(1): Custom processor in full-custom IC
Highly optimized

(2): Custom processor in FPGA
Parallelized circuit, slower IC technology but programmable

(3): Programmable processor in standard cell IC

Program runs (mostly) sequentially on moderate-costing IC

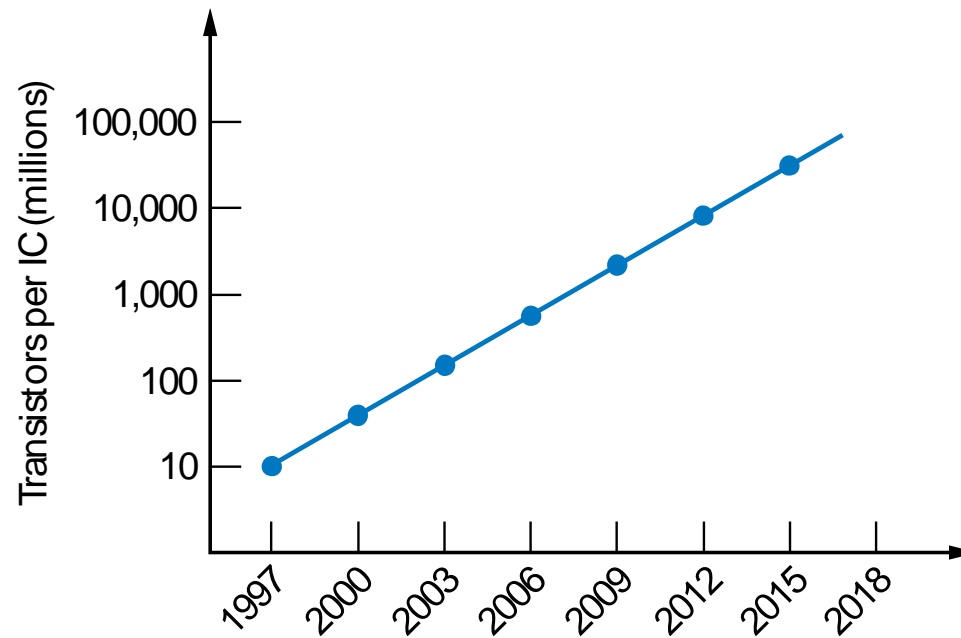
(4): Programmable processor in FPGA

Not only can processor be programmed, but FPGA can be programmed to implement multiple processors/coprocessors



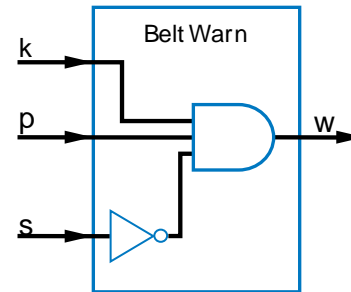
Key Trend in Implementation Technologies

- Transistors per IC doubling every 18 months for past three decades
 - Known as "Moore's Law"
 - Tremendous implications – applications infeasible at one time due to outrageous processing requirements become feasible a few years later
 - Can Moore's Law continue?



Chapter Summary

- Many ways to get from design to physical implementation
 - Manufactured IC technologies
 - Full-custom IC
 - Decide on every transistor and wire
 - Semi-custom IC
 - Transistor details pre-designed
 - Standard cell: Place cells and wire them
 - Gate array: Just wire existing gates
 - FPGAs
 - Fully programmable
 - Other technologies
 - Logic ICs, PLDs



(a) Digital circuit design



(b) Physical implementation



Homework

- Chapter 7: 3, 4, 8, 10, 20
- Homework is due on Thursday, February 11