ECE 274 Digital Logic

Implementation

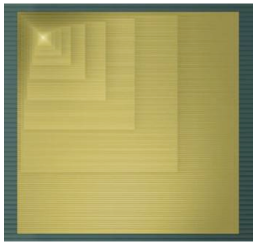
Digital Design 7

The University of Arizona
Tucson Arizona

Digital Design

Chapter 7: Implementation

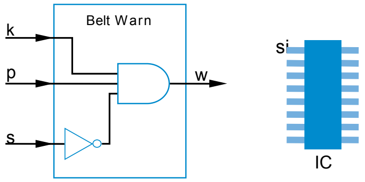
Slides to accompany the textbook *Digital Design*, First Edition,
by Frank Vahid, John Wiley and Sons Publishers, 2007.
http://www.ddvahid.com



Introduction

7.1

- 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

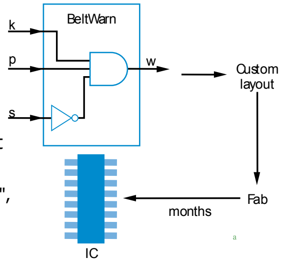
(b) Physical implementation

Note: Slides with animation are denoted with a small red "a" near the animated items

Manufactured IC Technologies

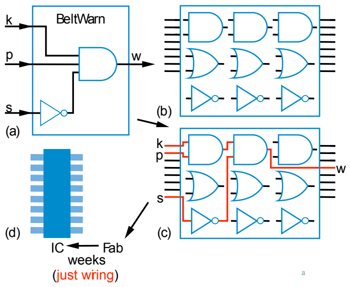
7.2

- 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



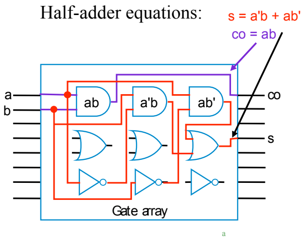
Manufactured IC Technologies – Gate Array ASIC

- (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



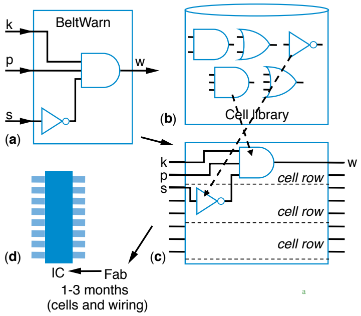
Manufactured IC Technologies – Gate Array ASIC

- (2a) Gate array
  - Example: Mapping a half-adder to a gate array



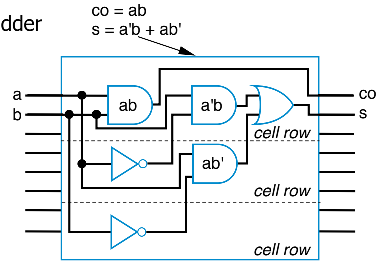
Manufactured IC Technologies – Standard Cell ASIC

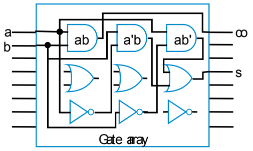
- (2) Semicustom IC
  - "Application-specific IC" (ASIC)
  - (a) Gate array or (b) standard cell
- (2b) Standard cell
  - Pre-layed-out "cells" exist in library, not on chip
  - Designer instantiates cells into pre-defined rows, and connects
  - Vs. gate array
    - Better performance/power/size
    - 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

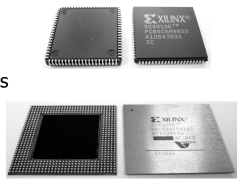




Notice fewer gates and shorter wires for standard cells versus gate array, but at cost of more design effort

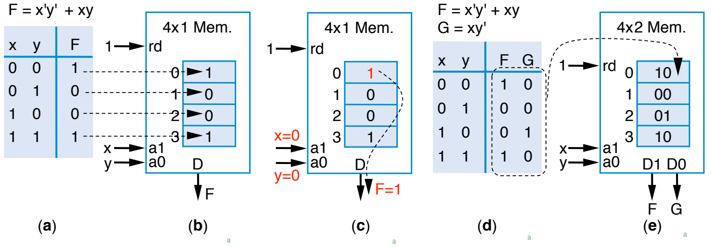
Programmable IC Technology – FPGA

- 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)



(a)

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

(b), (c) 4x1 Mem.

0	1
1	0
2	0
3	1

(d)

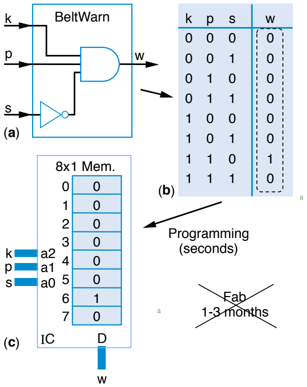
x	y	F	G
0	0	1	0
0	1	0	0
1	0	0	1
1	1	1	0

(e) 4x2 Mem.

0	10
1	00
2	01
3	10

FPGA Internals: Lookup Tables (LUTs)

- Example: Seat-belt warning light (again)



(b)

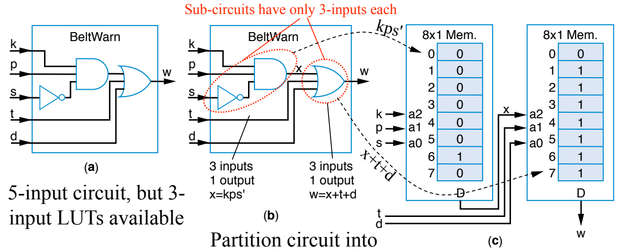
k	p	s	w
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

(c) 8x1 Mem.

0	0
1	0
2	0
3	0
4	0
5	0
6	1
7	0

FPGA Internals: Lookup Tables (LUTs)

- Lookup tables become inefficient for more inputs
  - 3 inputs → only 8 words
  - 8 inputs → 256 words; 16 inputs → 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:



5-input circuit, but 3-input LUTs available

Partition circuit into 3-input sub-circuits

Map to 3-input LUTs

First 8x1 Mem.

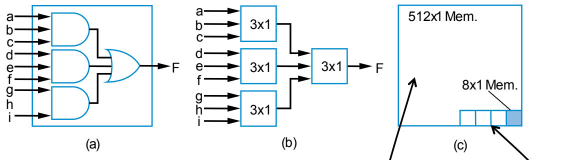
0	0
1	0
2	0
3	0
4	0
5	0
6	1
7	0

Second 8x1 Mem.

0	0
1	1
2	1
3	1
4	1
5	1
6	1
7	1

FPGA Internals: Lookup Tables (LUTs)

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



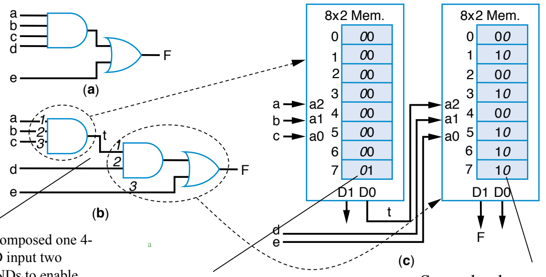
Original 9-input circuit

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



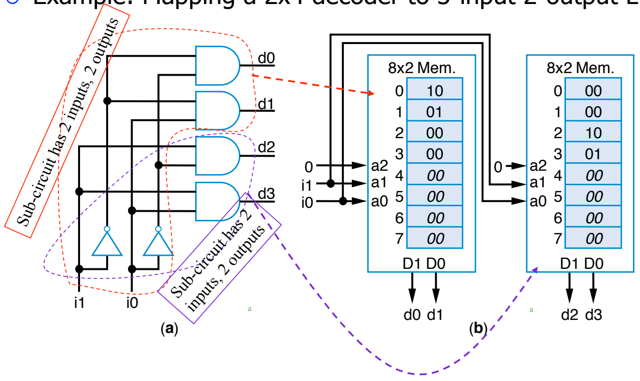
(Note: decomposed one 4-input AND input two smaller ANDs to enable partitioning into 3-input sub-circuits)

First column unused; second column implements AND

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

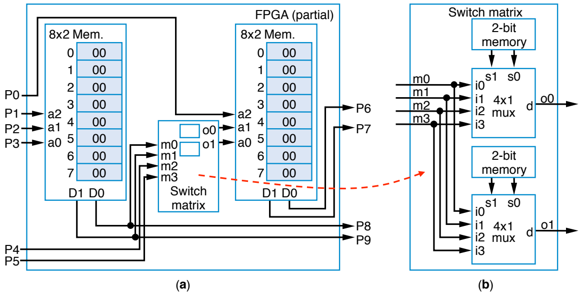
FPGA Internals: Lookup Tables (LUTs)

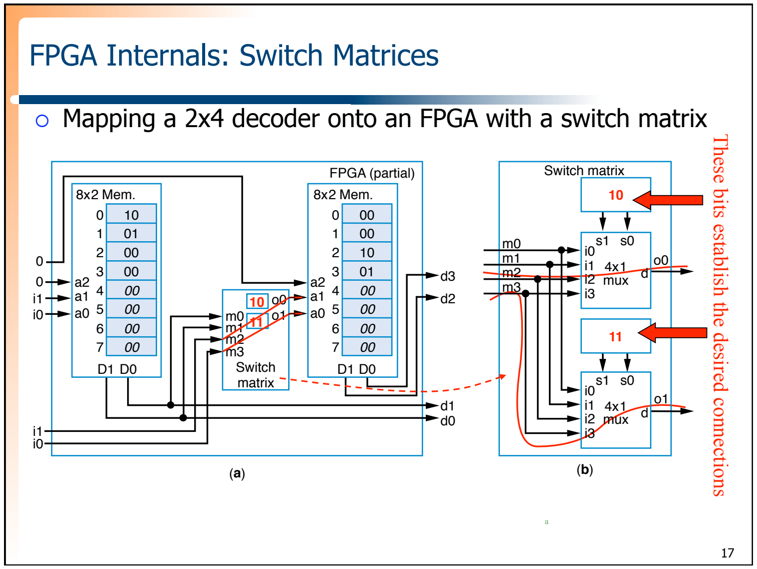
- Example: Mapping a 2x4 decoder to 3-input 2-output LUTs



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



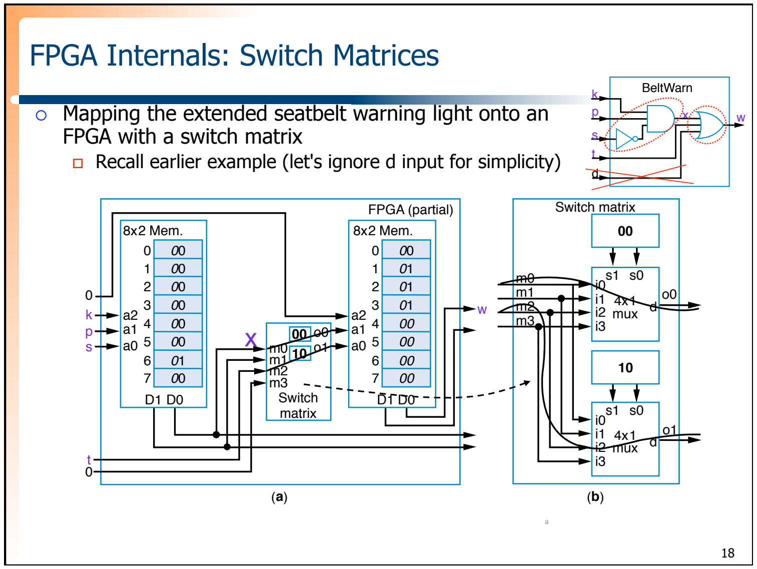


FPGA Internals: Switch Matrices

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

These bits establish the desired connections

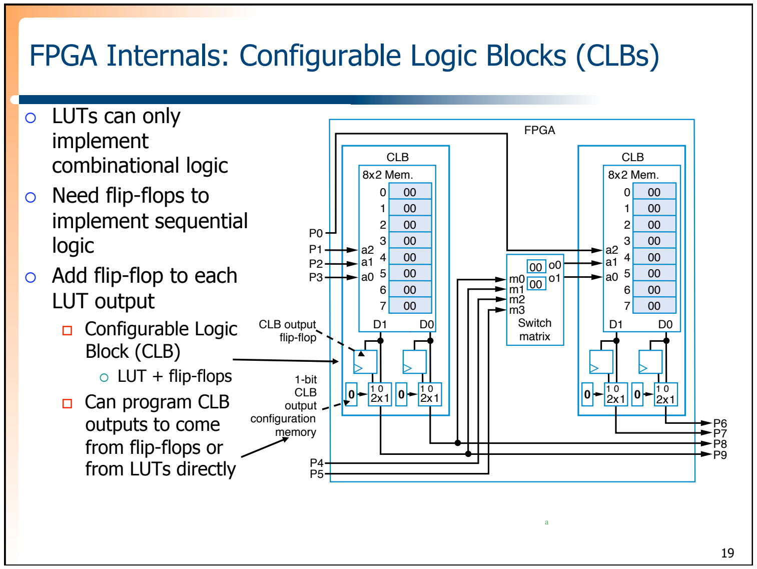
17



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)

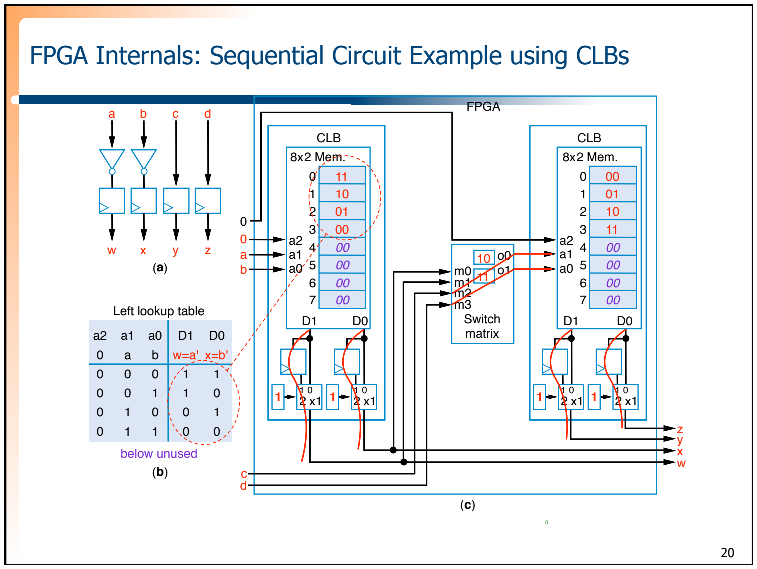
18



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

19



FPGA Internals: Sequential Circuit Example using CLBs

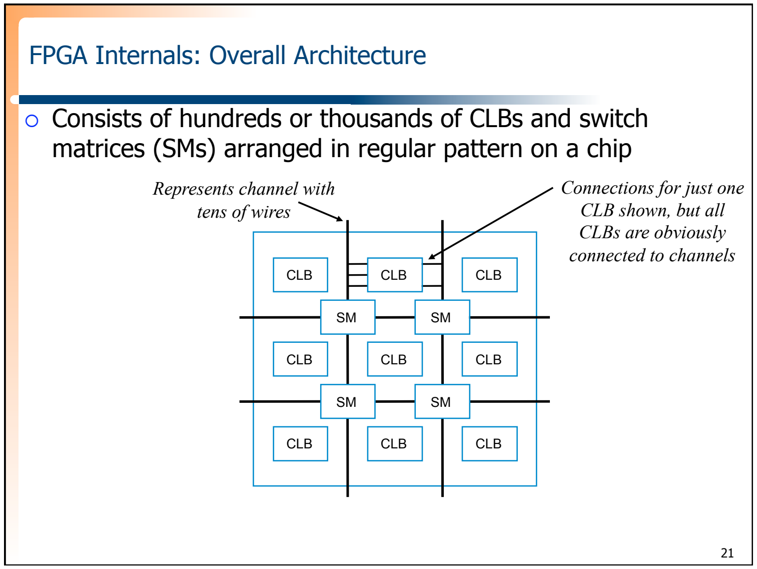
Left lookup table

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)

20



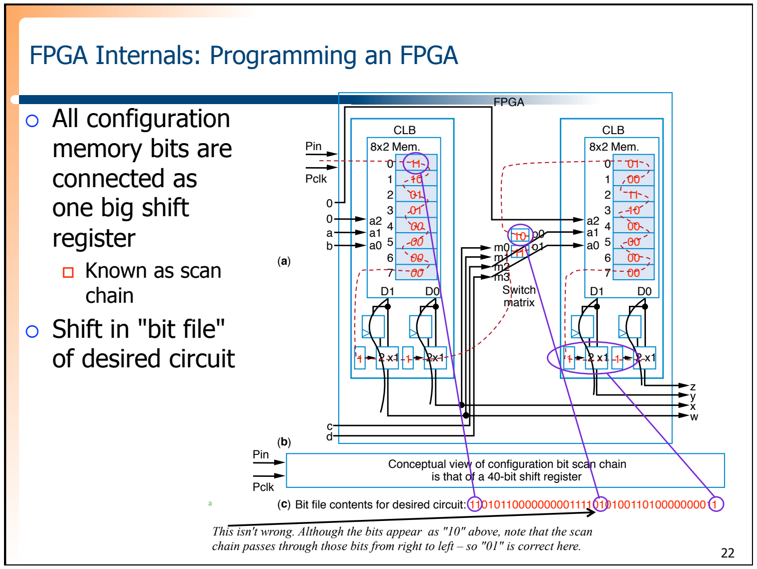
FPGA Internals: Overall Architecture

- 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

21



FPGA Internals: Programming an FPGA

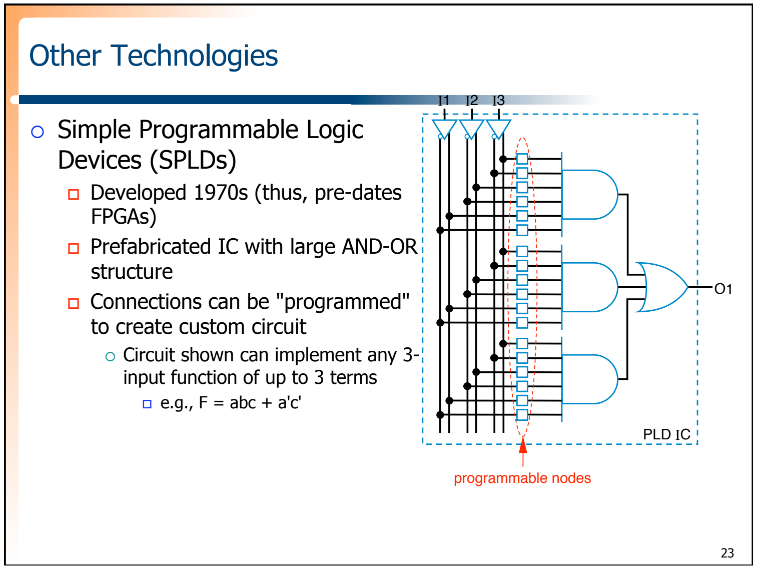
- All configuration memory bits are connected as one big shift register
  - Known as scan chain
- Shift in "bit file" of desired circuit

Conceptual view of configuration bit scan chain is that of a 40-bit shift register

(c) Bit file contents for desired circuit: 1101011000000000011110101001101000000011

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.

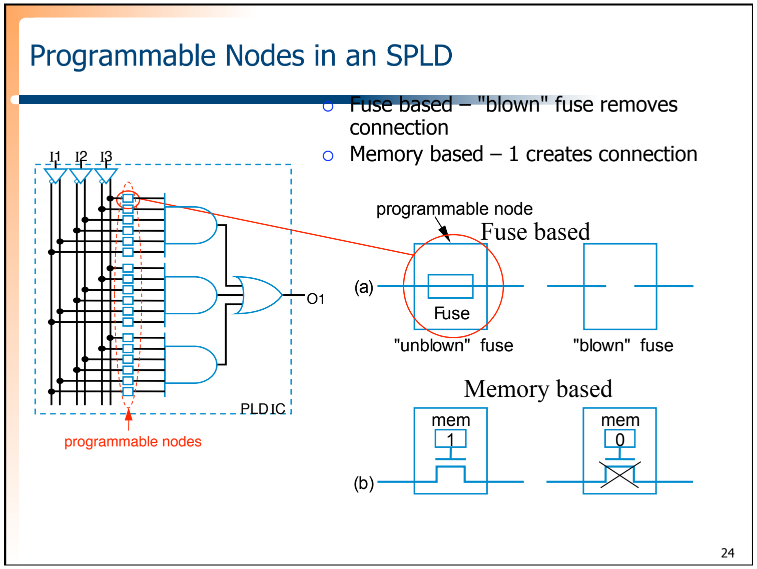
22



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'

23



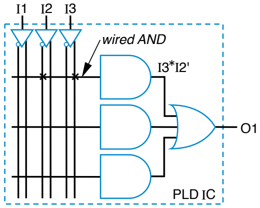
Programmable Nodes in an SPLD

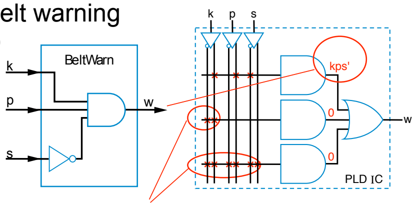
- Fuse based – "blown" fuse removes connection
- Memory based – 1 creates connection

24

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

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)

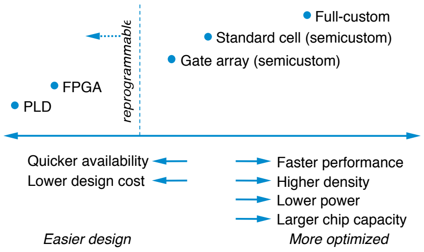
7.5

Technology Comparisons

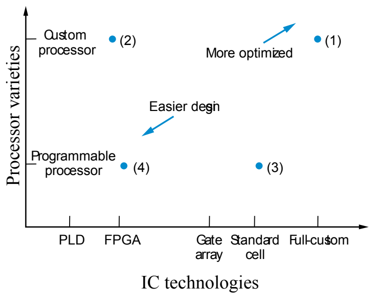
TABLE 7.2: Sample % of new implementations in various technologies. Total is more than 100% due to overlap among categories.

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