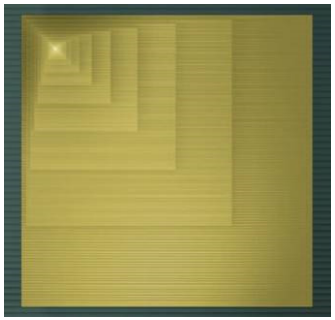


Digital Design

Chapter 7:

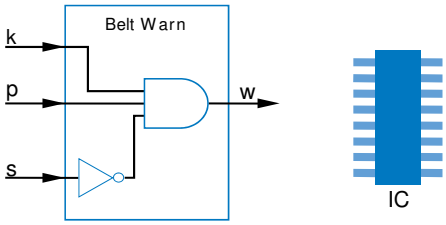
Physical Implementation

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

Some changes by Mark Brehob, any errors probably mine.

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)?

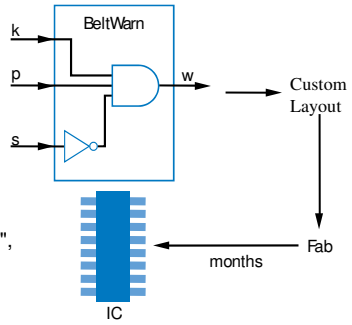


(a) Digital circuit design

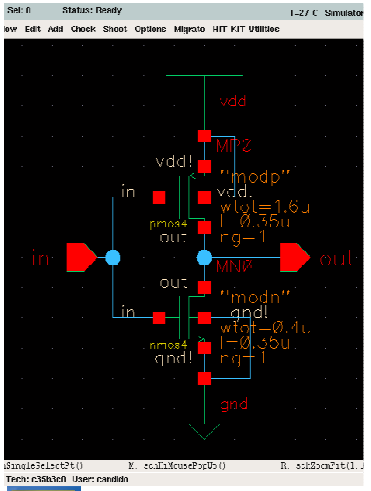
(b) Physical implementation

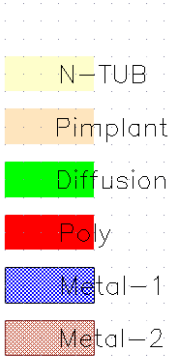
Manufactured IC Technologies

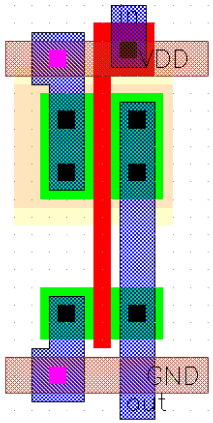
- We can manufacture our own IC
  - Months of time and millions of dollars
  - (1) Full-custom or (2) semicustom
- (1) Full-custom IC
  - We make a *full custom layout*
    - Using CAD tools
    - Layout describes the location and size of every transistor and wire
  - A *fab* (fabrication plant) builds 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



Full Custom inverter







Images from http://wiki.usgroup.eu/wiki/public/tutorials/fullcustomvirtuoso and http://www.iis.ee.ethz.ch/~kgf/aries/8.html

Take away – full custom

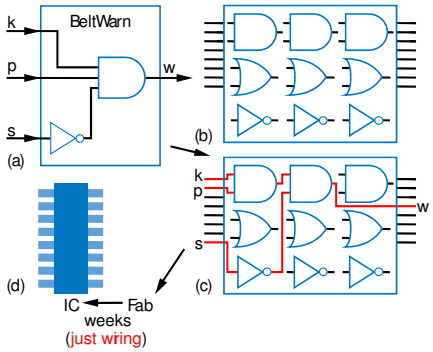
- Very expensive and slow
  - Requires folks who really know what they are doing and requires them to take time
    - EECS 312 final project is to build a fairly small device (e.g. 4-bit adder) with certain space and power bounds. This is a multi-week thing!
  - Fabrication will generally not have any short cuts.
    - So might take months to get this spun.
- Best performance
  - Can tune the heck out of things and make stuff go really fast or use very little power.
- Where do you think this sees use?

Gate array

- From Wikipedia:
  - A gate array circuit is a prefabricated silicon chip circuit with no particular function in which transistors, standard NAND or NOR logic gates, and other active devices are placed at regular predefined positions and manufactured on a wafer, usually called a master slice. Creation of a circuit with a specified function is accomplished by adding a final surface layer or layers of metal interconnects to the chips on the master slice late in the manufacturing process, joining these elements to allow the function of the chip to be customized as desired.
  - Gate array master slices are usually prefabricated and stockpiled in large quantities regardless of customer orders. The design and fabrication according to the individual customer specifications may be finished in a shorter time compared with standard cell or full custom design.

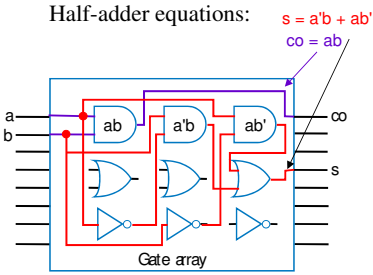
Manufactured IC Technologies – Gate Array ASIC

- (2) Semi-custom IC
  - "Application-specific IC" (ASIC)
  - (a) Gate array or (b) standard cell
- (2a) Gate array
  - Series of gates already laid out on chip
  - We just wire them together
    - Using CAD tools
  - Vs. full-custom
    - Cheaper and quicker to design
    - But worse performance, size, power
  - Was quite popular
    - Sees less use today AFAICT



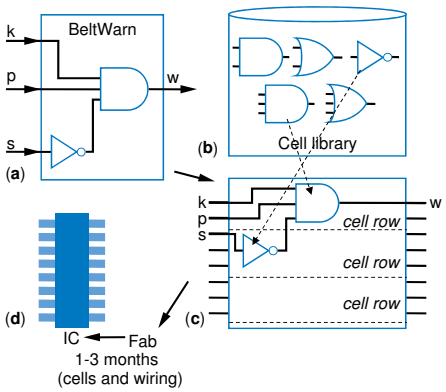
Manufactured IC Technologies – Gate Array ASIC

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



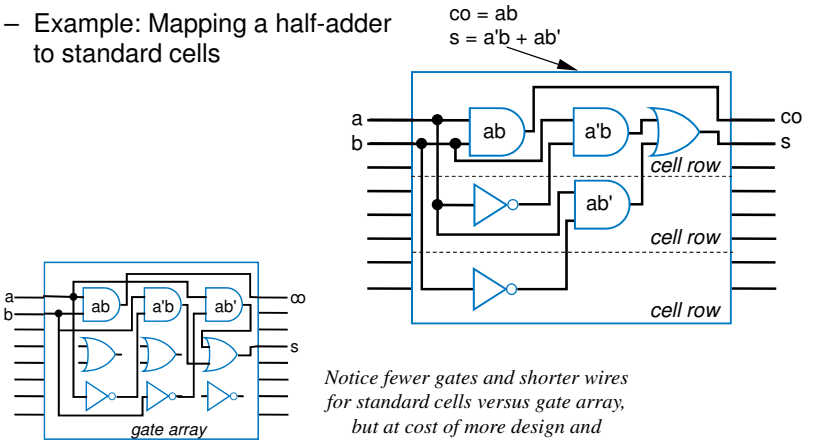
Manufactured IC Technologies – Standard Cell ASIC

- (2) Semicustom IC
  - "Application-specific IC" (ASIC)
  - (a) Gate array or (b) standard cell
- (2b) Standard cell
  - Pre-laid-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 and manufacturing effort

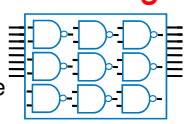
Question:

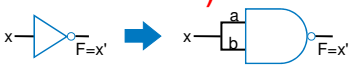
- What are the key differences between standard cells and gate array design?

Implementing Circuits Using NAND Gates Only

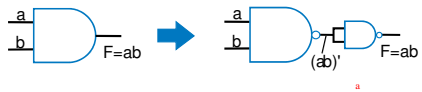
(We've been doing this since GA1!)

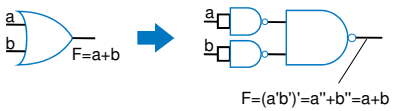
- Gate array may have NAND gates only
  - NAND is universal gate
    - Any circuit can be mapped to NANDs only
- Convert AND/OR/NOT circuit to NAND-only circuit using mapping rules
  - After converting, remove double inversions



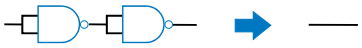


Inputs			Output
x	a	b	F
0	0	0	1
1	1	1	0



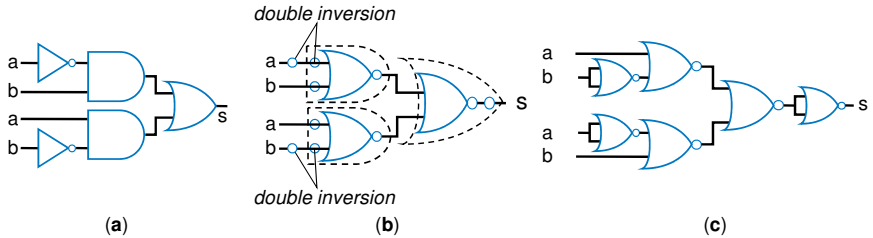


Double inversion



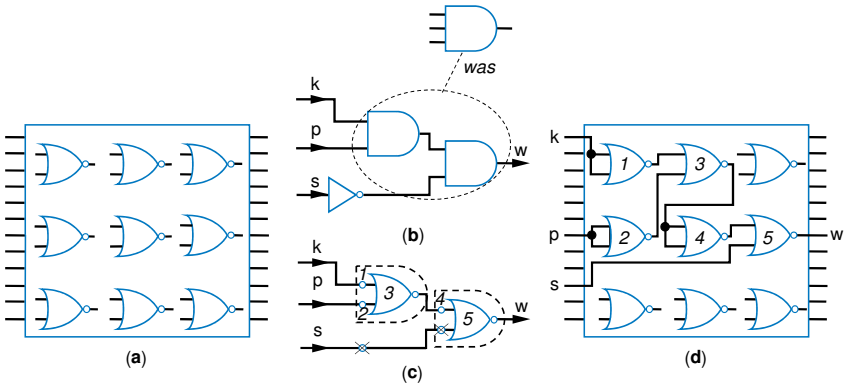
Implementing Circuits Using NOR Gates Only

- Example: Half adder



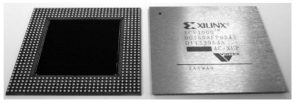
Implementing Circuits Using NOR Gates Only

- 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



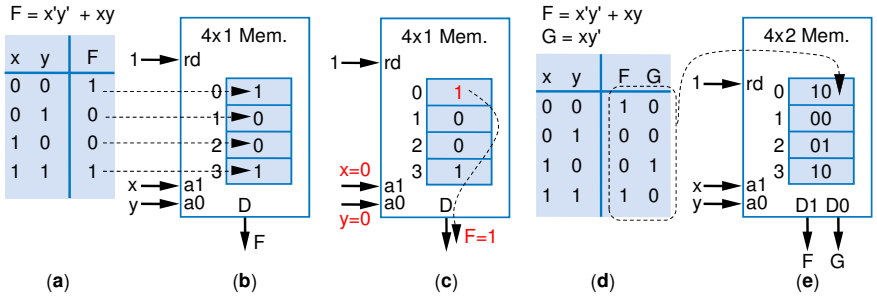
Programmable IC Technology – FPGA

- 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



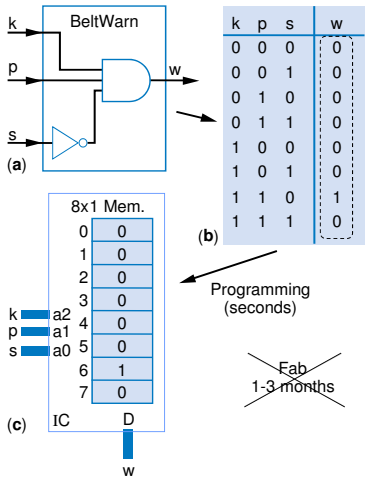
FPGA Internals: Lookup Tables (LUTs)

- 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)



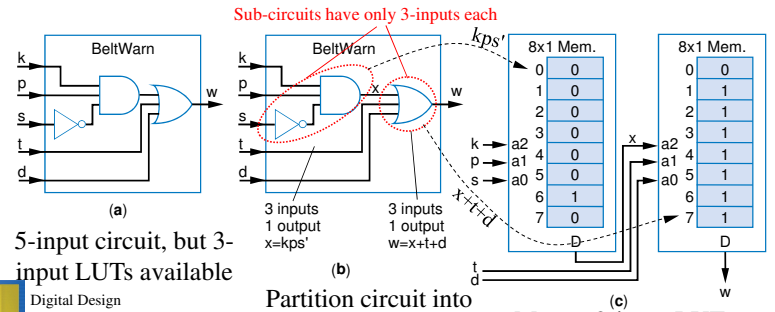
FPGA Internals: Lookup Tables (LUTs)

- Example: Seat-belt warning light (again)



FPGA Internals: Lookup Tables (LUTs)

- Lookup tables become inefficient for more inputs
  - 3 inputs → 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:

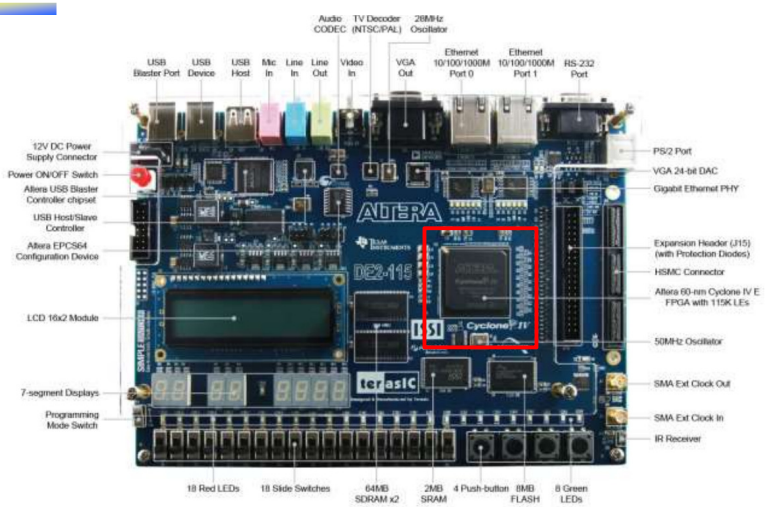


5-input circuit, but 3-input LUTs available

Partition circuit into 3-input sub-circuits

Map to 3-input LUTs

DE2-115 FPGA board



The DE2-115 FPGA

Table 1–1. Resources for the Cyclone IV E Device Family

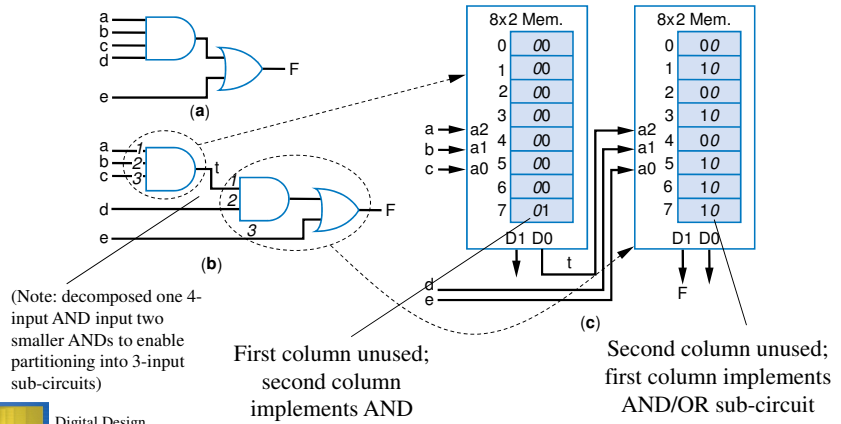
Resources	EP4CE6	EP4CE10	EP4CE15	EP4CE22	EP4CE30	EP4CE40	EP4CE55	EP4CE75	EP4CE115
Logic elements (LEs)	6,272	10,320	15,408	22,320	28,848	39,600	55,856	75,408	114,480
Embedded memory (Kbits)	270	414	504	594	594	1,134	2,340	2,745	3,888
Embedded 18 × 18 multipliers	15	23	56	66	66	116	154	200	266
General-purpose PLLs	2	2	4	4	4	4	4	4	4
Global Clock Networks	10	10	20	20	20	20	20	20	20
User I/O Banks	8	8	8	8	8	8	8	8	8
Maximum user I/O (1)	179	179	343	153	532	532	374	426	528

Each LE has a look up table (LUT) inside of it, as well as a number of other useful things.

Let's look at a generic LUT…

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



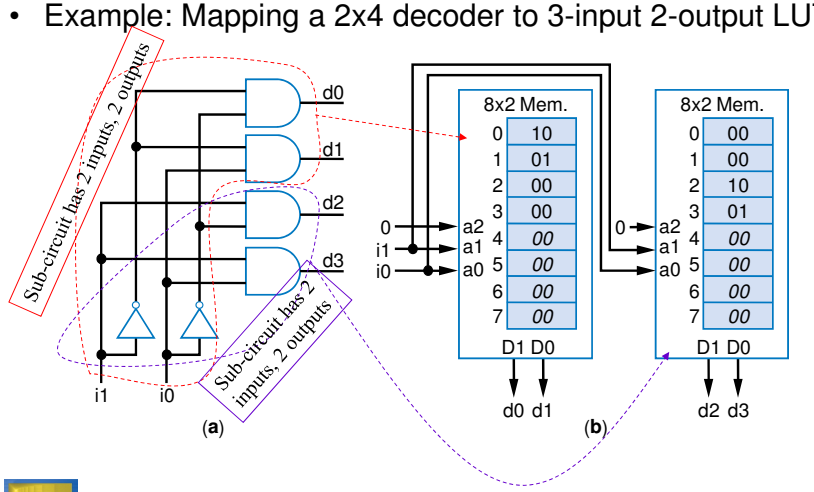
(Note: decomposed one 4-input AND input two smaller ANDs to enable partitioning into 3-input sub-circuits)

First column unused; second column implements AND

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

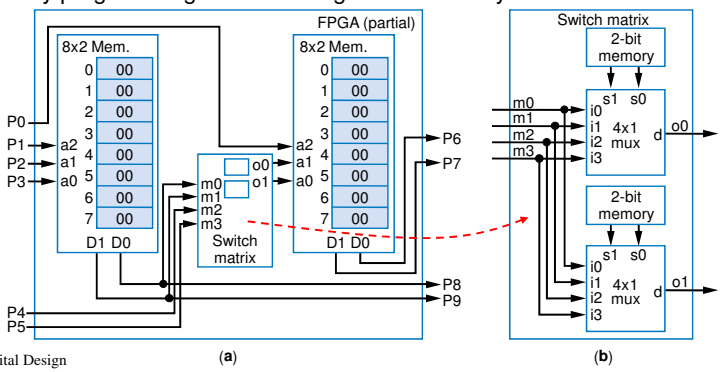
FPGA Internals: Lookup Tables (LUTs)

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



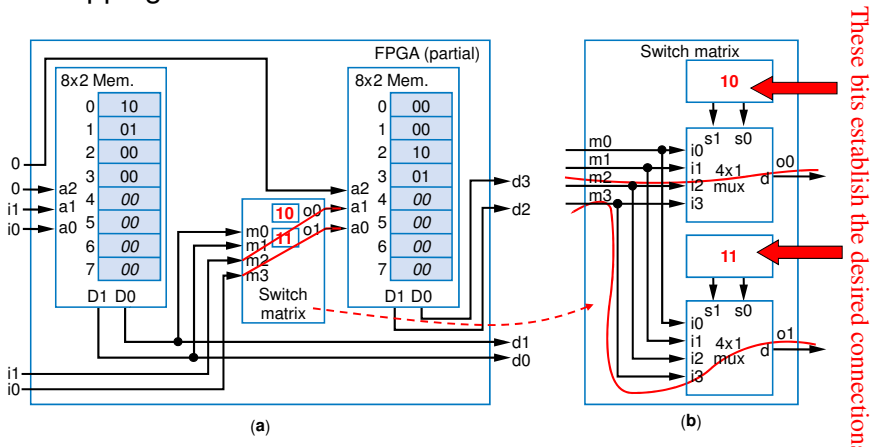
FPGA Internals: Switch Matrices

- 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

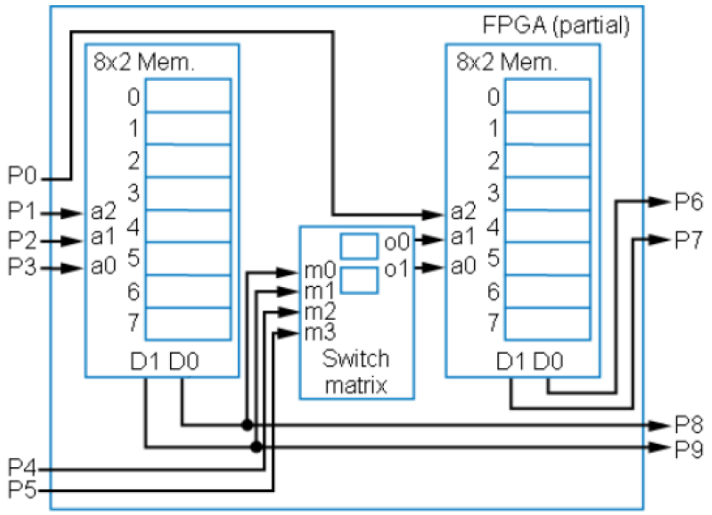
- Mapping a 2x4 decoder onto an FPGA with a switch matrix



These bits establish the desired connections

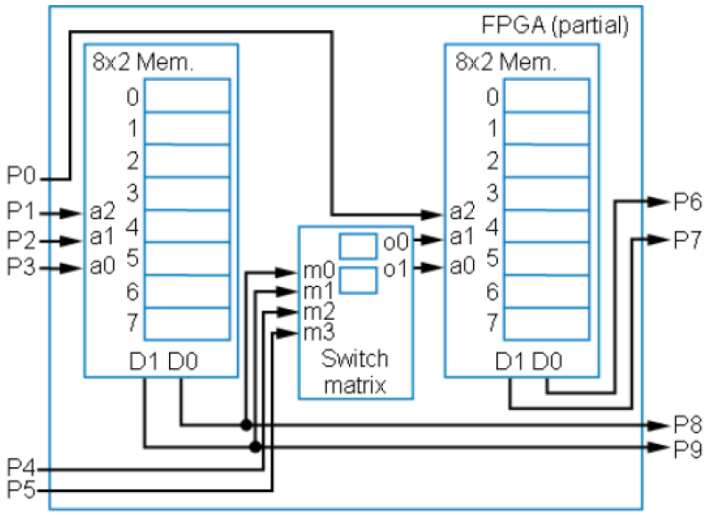
P0	P1	P2	P3	P4	P5	P6	P7	P8	P9

F=A*B+C
G=A*B*C*!D*E



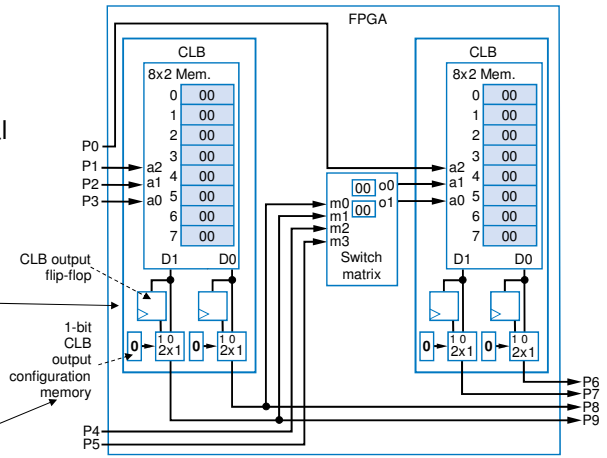
P0	P1	P2	P3	P4	P5	P6	P7	P8	P9

F=A*B+C
G=A*B*C*!D*E

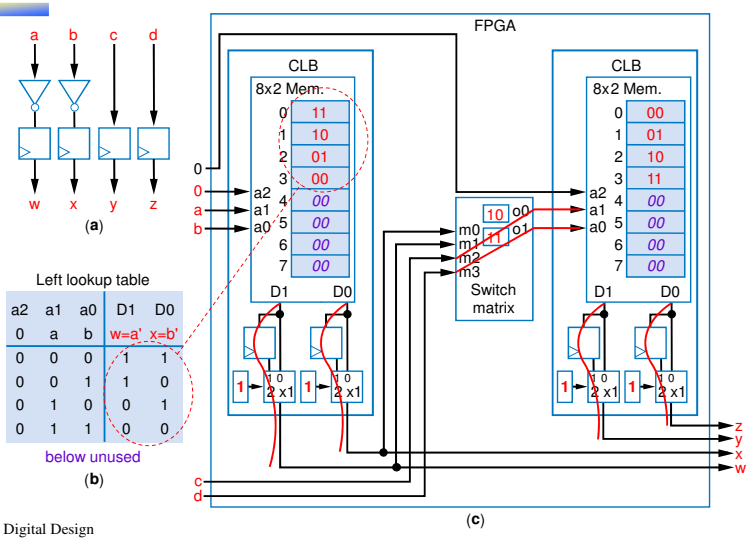


FPGA Internals: Configurable Logic Blocks (CLBs)

- 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

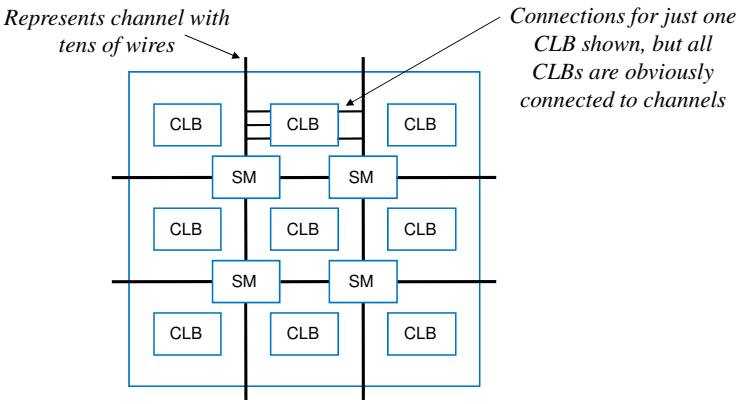


FPGA Internals: Sequential Circuit Example using CLBs



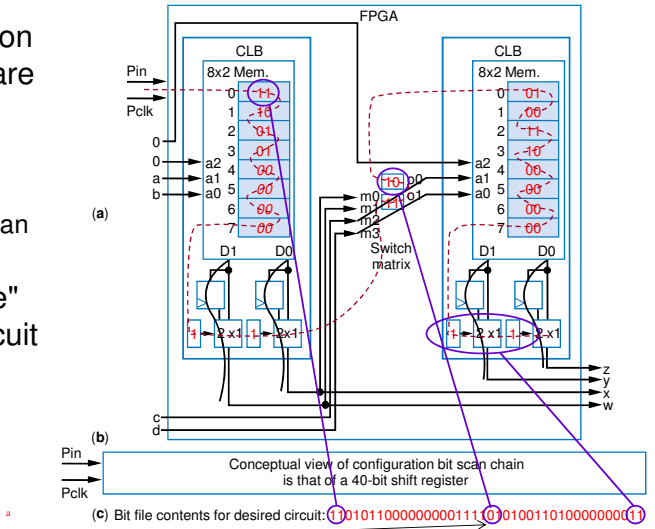
FPGA Internals: Overall Architecture

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



FPGA Internals: Programming an FPGA

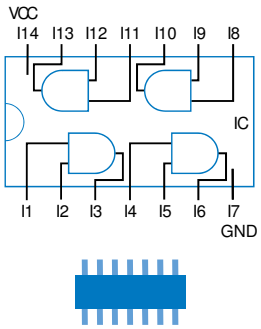
- All configuration memory bits are connected as one big shift register
  - Known as scan chain
- Shift in "bit file" of desired circuit



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.

Other Technologies

- 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

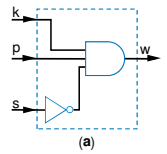
- Example: Seat belt warning light using off-the-shelf 7400 ICs
  - Option 1: Use one 74LS08 IC having 2-input AND gates, and one 74LS04 IC having inverters

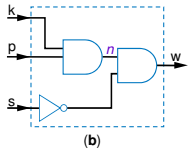
TABLE 7.1: Commonly used 7400-series ICs.

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

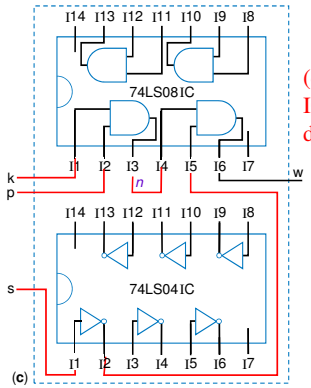
Source: www.digikey.com

(a) Desired circuit





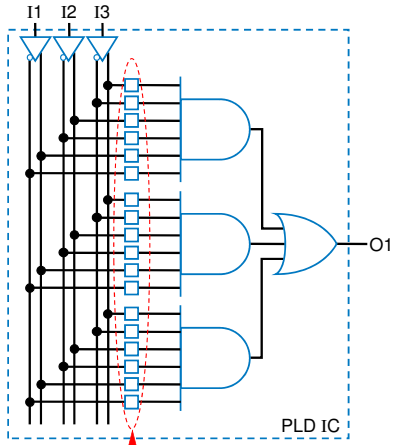
(b) Decompose into 2-input AND gates



(c) Connect ICs to create desired circuit

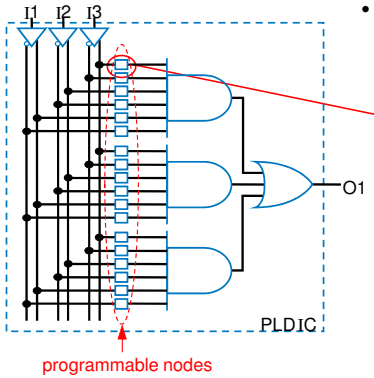
Other Technologies

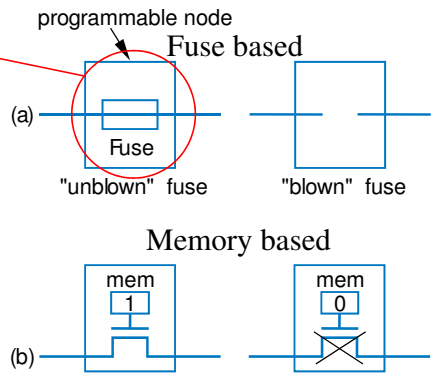
- Simple Programmable Logic Devices (SPLDs)
  - Developed 1970s (thus, pre-dates FPGAs)
  - Prefabricated IC with large AND-OR structure
  - Connections can be "programmed" to create custom circuit
    - Circuit shown can implement any 3-input function of up to 3 terms
      - e.g., F = abc + a'c'



Programmable Nodes in an SPLD

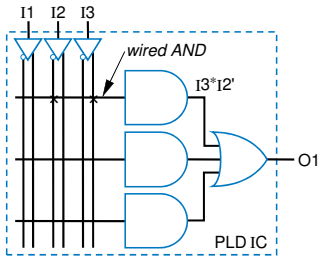
- Fuse based – "blown" fuse removes connection
- Memory based – 1 creates connection



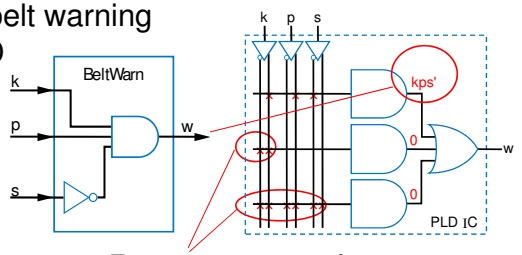


PLD Drawings and PLD Implementation Example

- 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

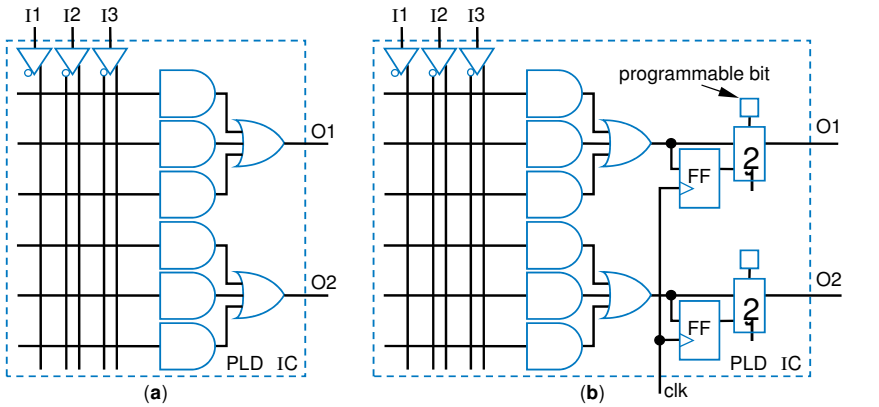


- Example: Seat belt warning light using SPLD



Two ways to generate a 0 term

PLD Extensions



Two-output PLD

PLD with programmable registered outputs

More on PLDs

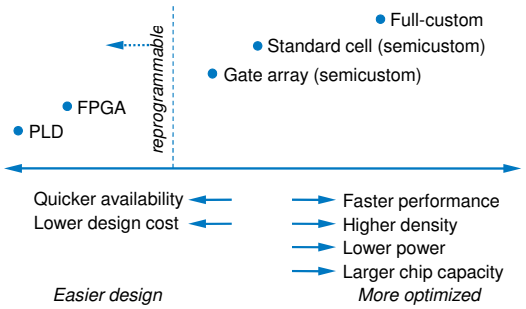
- Originally (1970s) known as Programmable Logic Array – *PLA*
  - Had programmable AND and OR arrays
- AMD created "Programmable Array Logic" – "*PAL*" (trademark)
  - Only AND array was programmable (fuse based)
- Lattice Semiconductor Corp. created "Generic Array Logic – "GAL" (trademark)
  - Memory 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)

Technology Comparisons

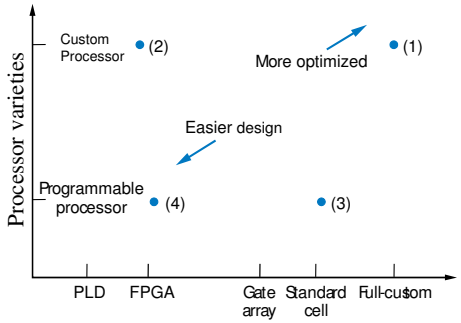
TABLE 7.2: Sample % of new implementations in various technologies. Total is more than 100% due to overlap among categories.

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



IC technologies

(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 were 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 – No-ish.