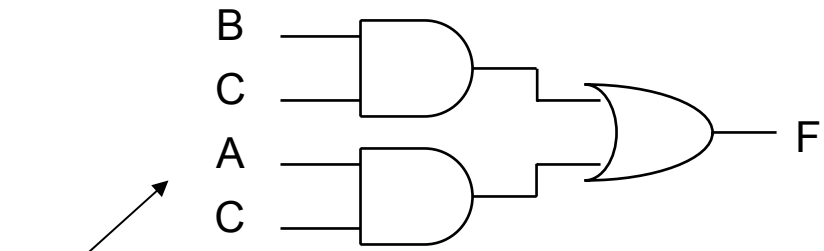
Field Programmable Gate Arrays

FPGA

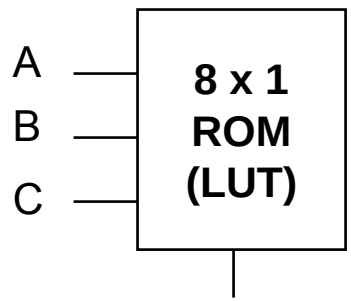
Using ROM as Combinational Logic

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

tt1



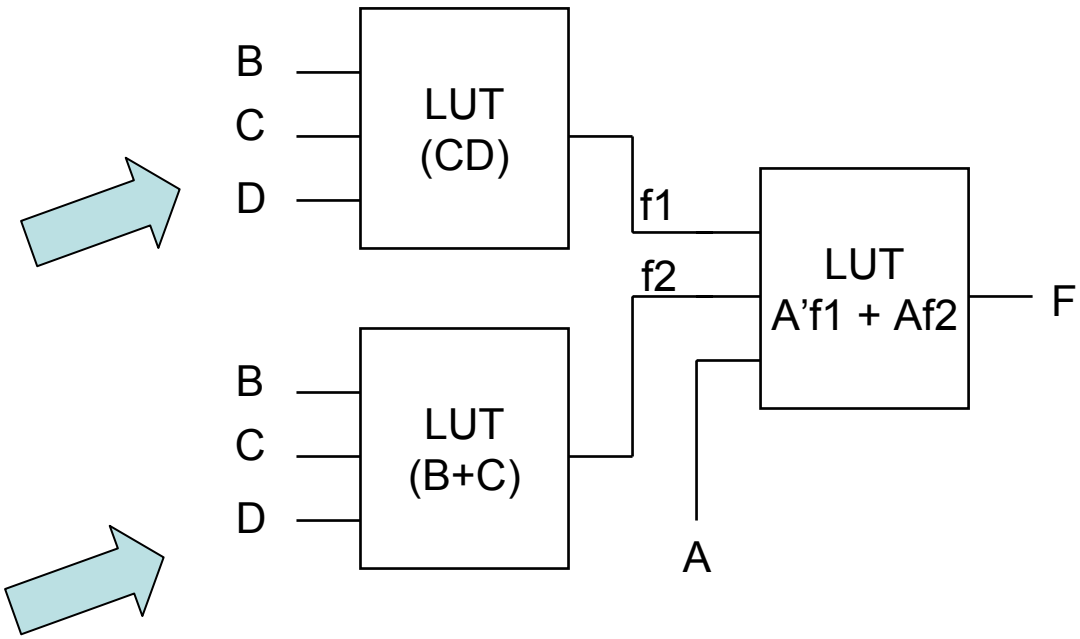
schem1



lut1

Mapping Larger Functions To ROMs

A	B	C	D	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1



Very similar to how we decomposed functions to implement with MUX blocks…

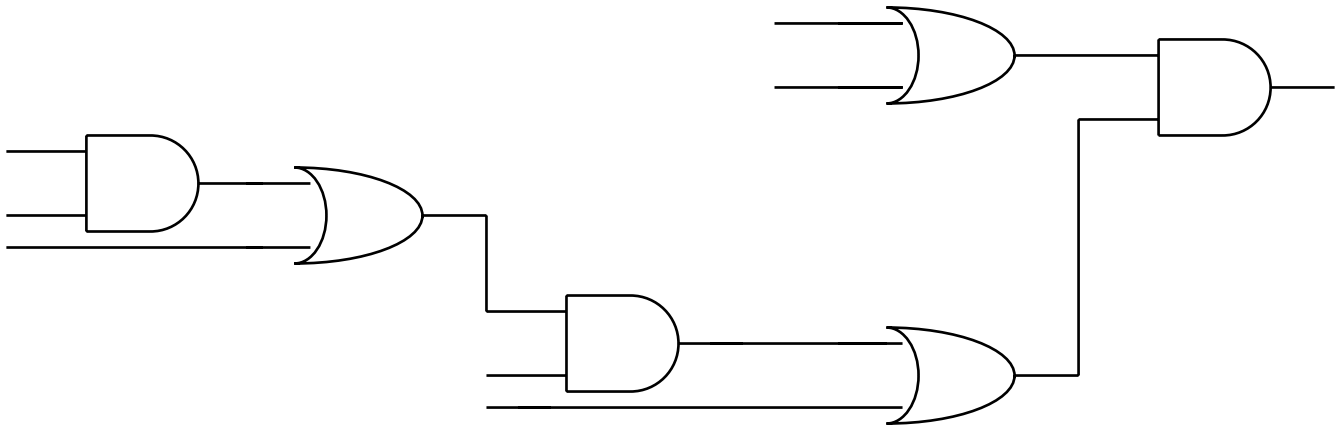
ROM vs. LUT

- A ROM can be used as a lookup table (LUT)

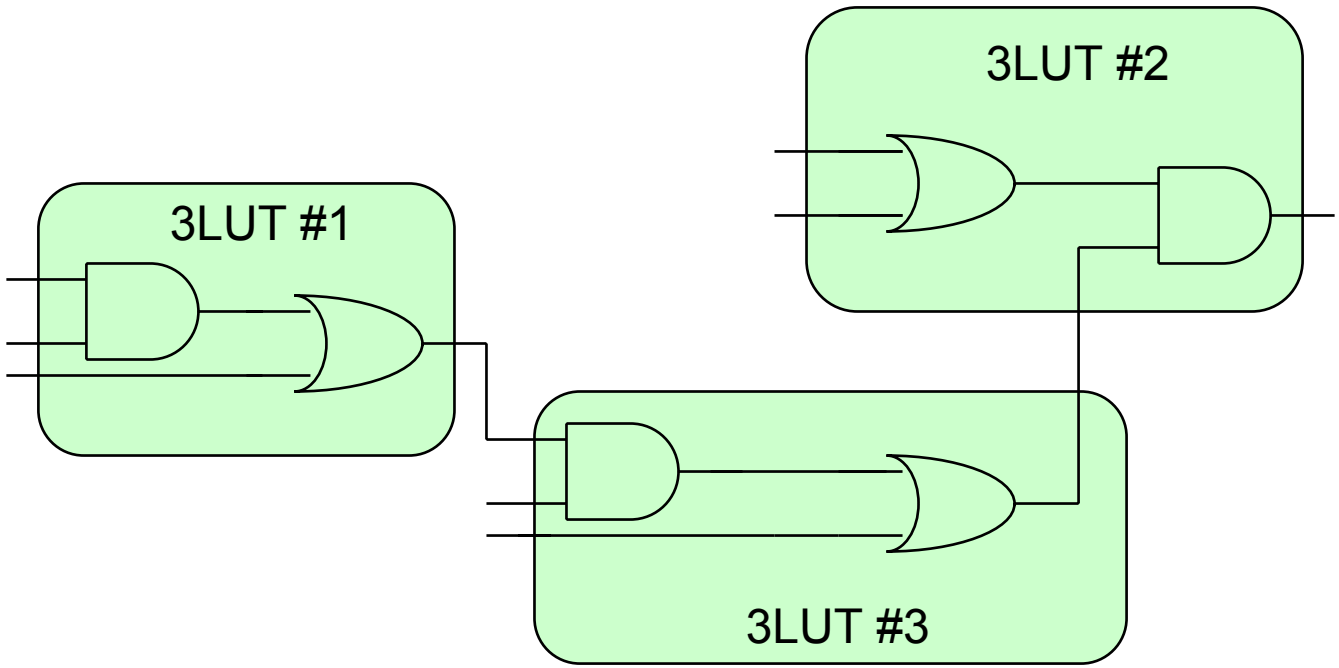
- An FPGA contains many, many such LUTs
  - 1,000's of them
- A 3LUT has 3 inputs + 1 output
- A 4LUT has 4 inputs + 1 output

- 3LUTs and 4LUTs are most common…

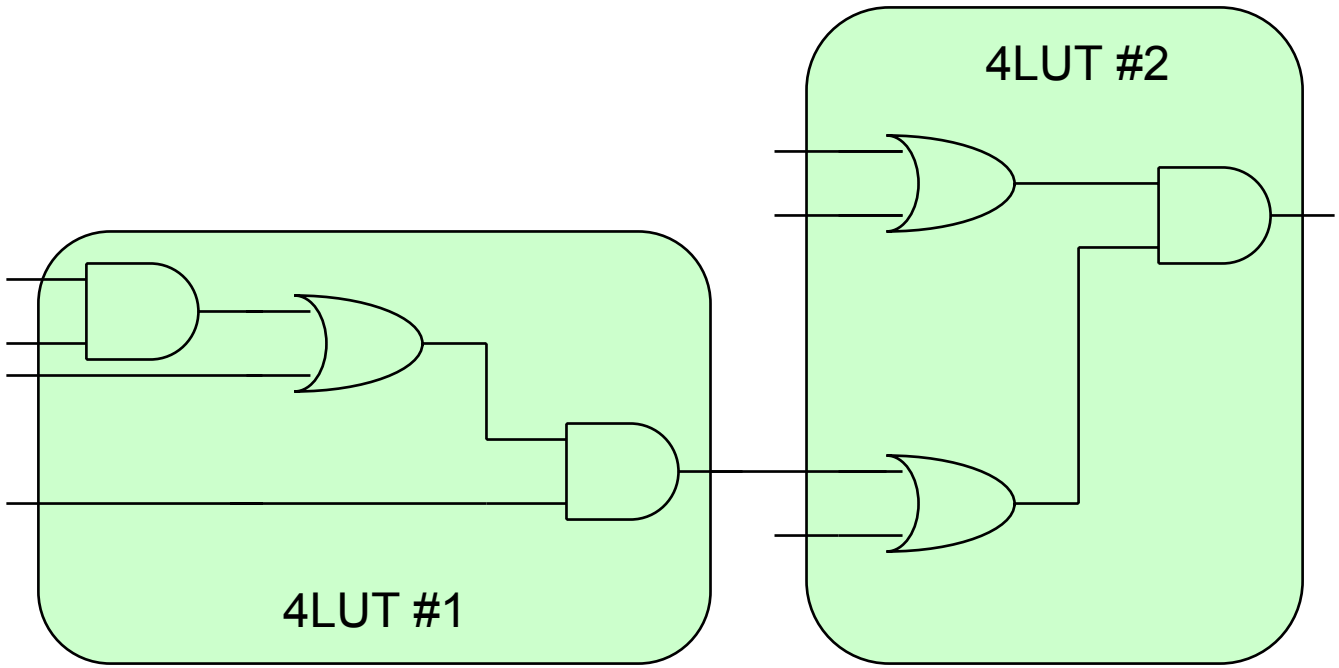
Mapping a Gate Network to LUTs



Mapping a Gate Network to 3LUTs



Mapping Same Network to 4LUTs



Mapping Equations to LUTs

- Each of these equations require 1 LUT:

  F = a

  F = abcd

  F = a′b′c′d + a′bcd′ + a′b′c′d′ + a′b′cd

  F = a + b + c + d

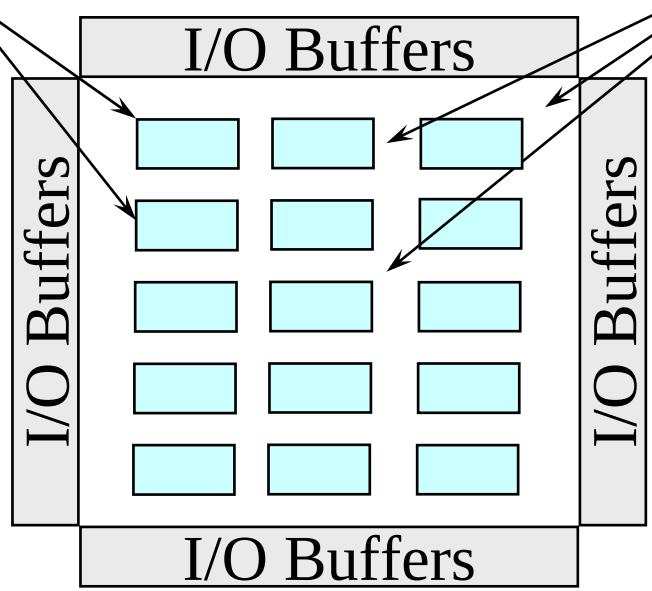
  F = (a +b + c + d)(a′ + b′ + c′ + d′)

Not a function of equation complexity,
is a function of # unique inputs

FPGAs – What Are They?

Programmable logic elements (LEs)

Programmable wiring areas

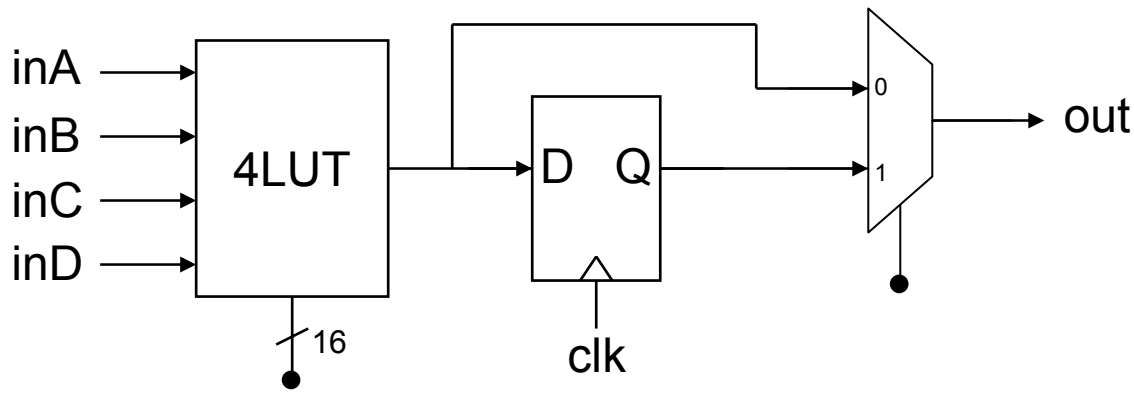


I/O Buffers communicate between FPGA and the outside world

An FPGA is a Programmable Logic Device (PLD)
It can be programmed to perform any function desired.

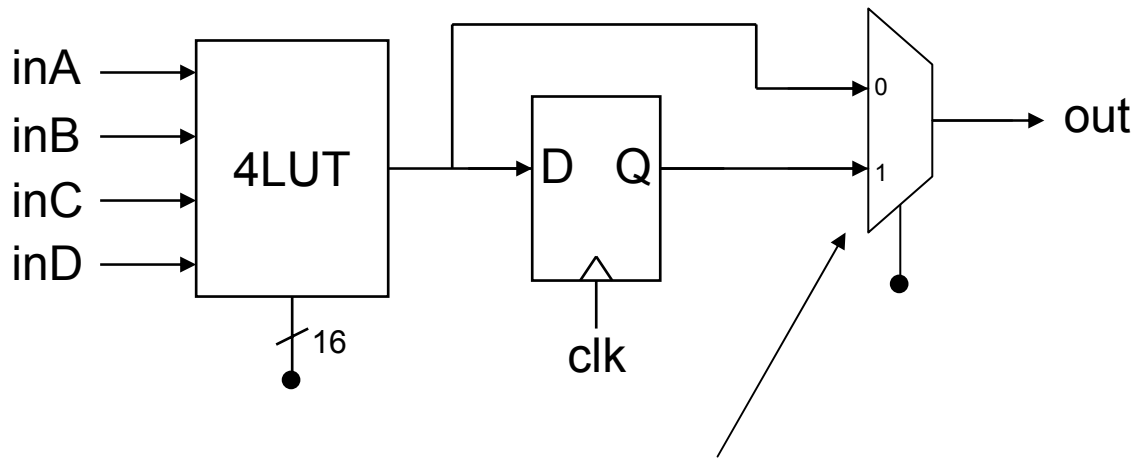
Programmable Logic Elements (LEs)

- Typically contain a LUT, wires, and storage



Programmable Logic Elements (LEs)

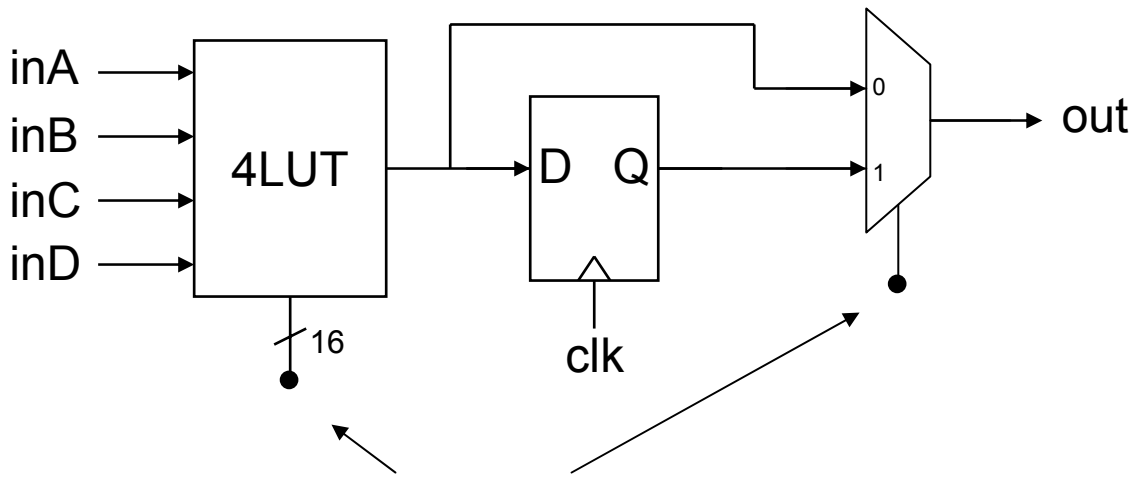
- Typically contain a LUT, wires, and storage



This one provides a registered or unregistered function of the 4 input variables

Programmable Logic Elements (LEs)

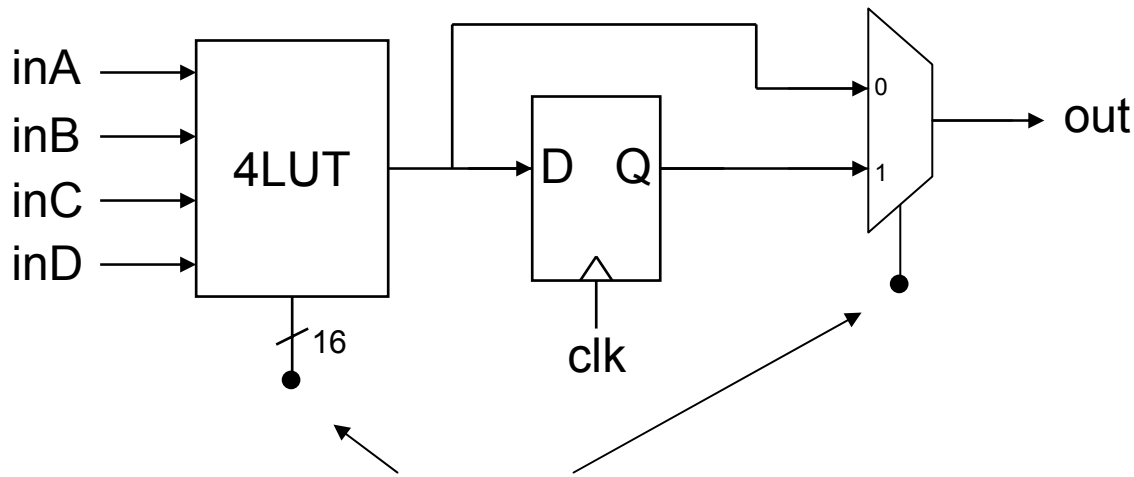
- Typically contain a LUT, wires, and storage



Black dots indicate programming bits
(configuration bits)

Programmable Logic Elements (LEs)

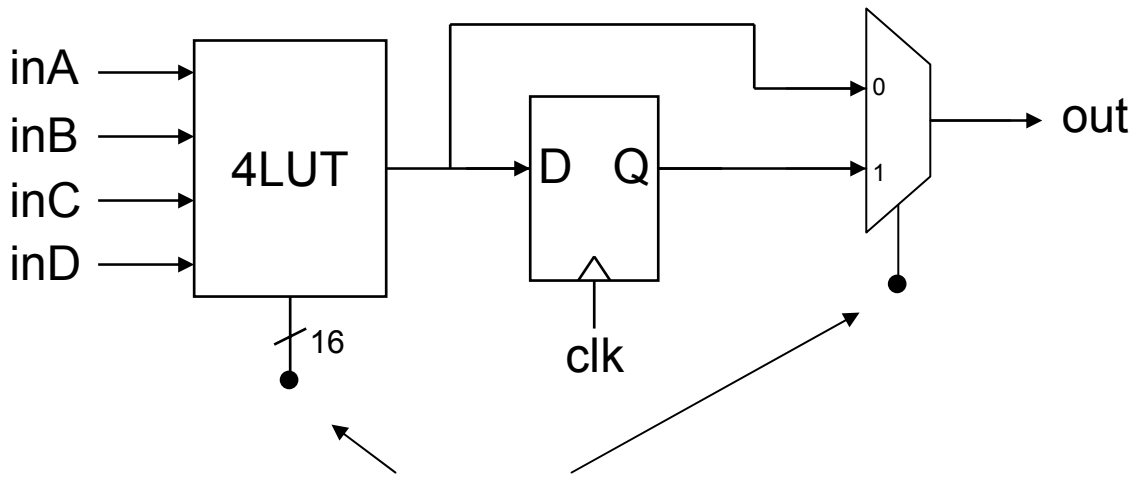
- Typically contain a LUT, wires, and storage



On power up, configuration bits are loaded into FPGA
This customizes its operation (LUT function, MUX selection)

Programmable Logic Elements (LEs)

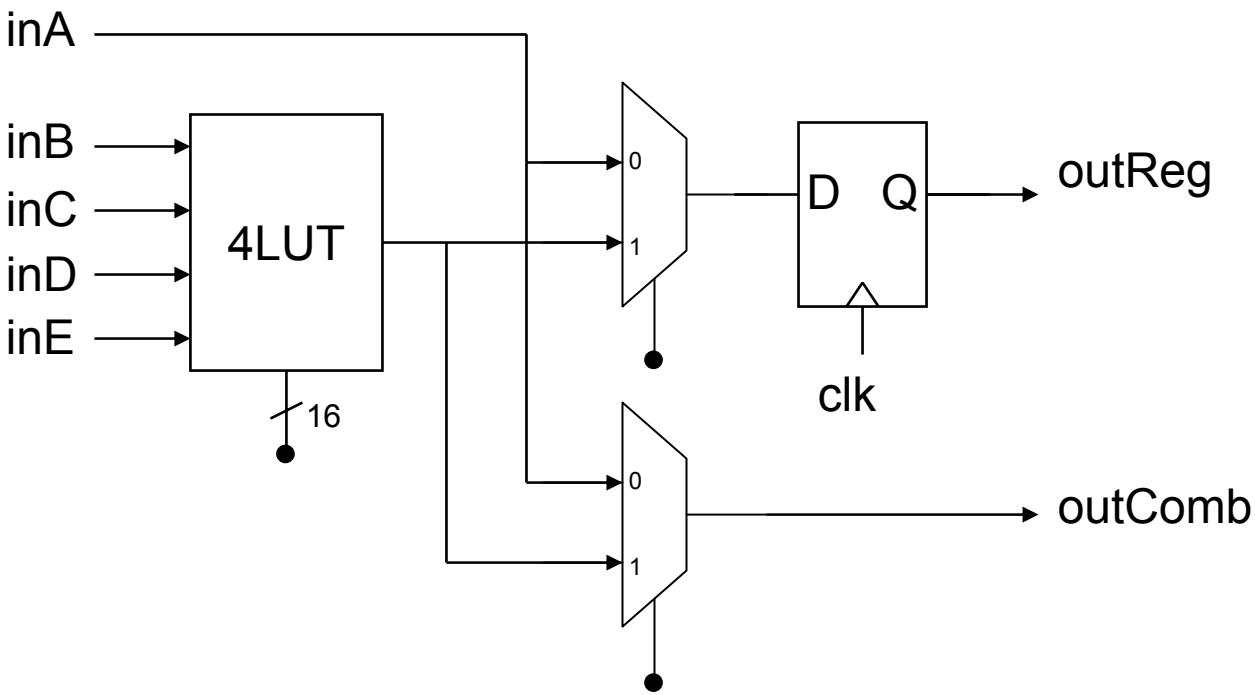
- Typically contain a LUT, wires, and storage



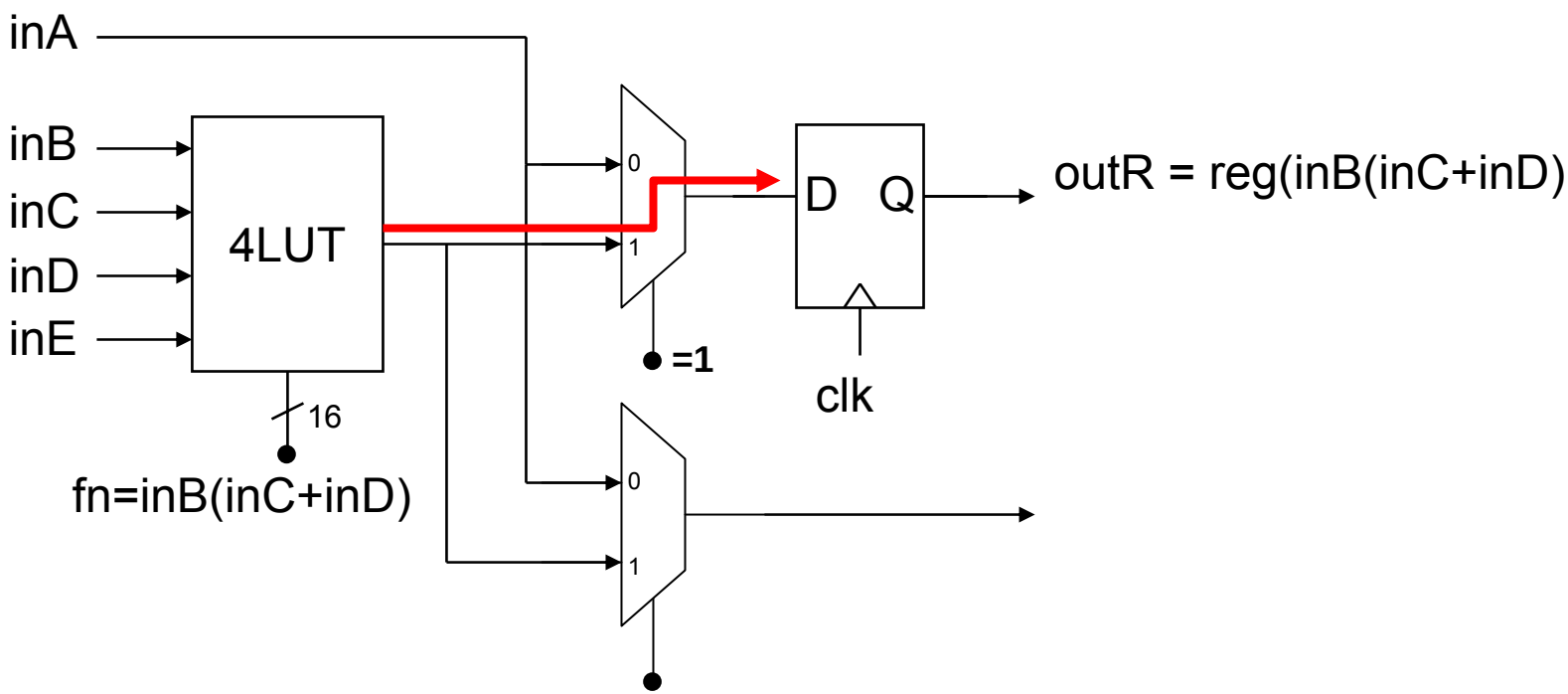
Typically, configuration bits are not changed during circuit operation
(but, some FPGA's are *dynamically reconfigurable*)

Another LE Structure

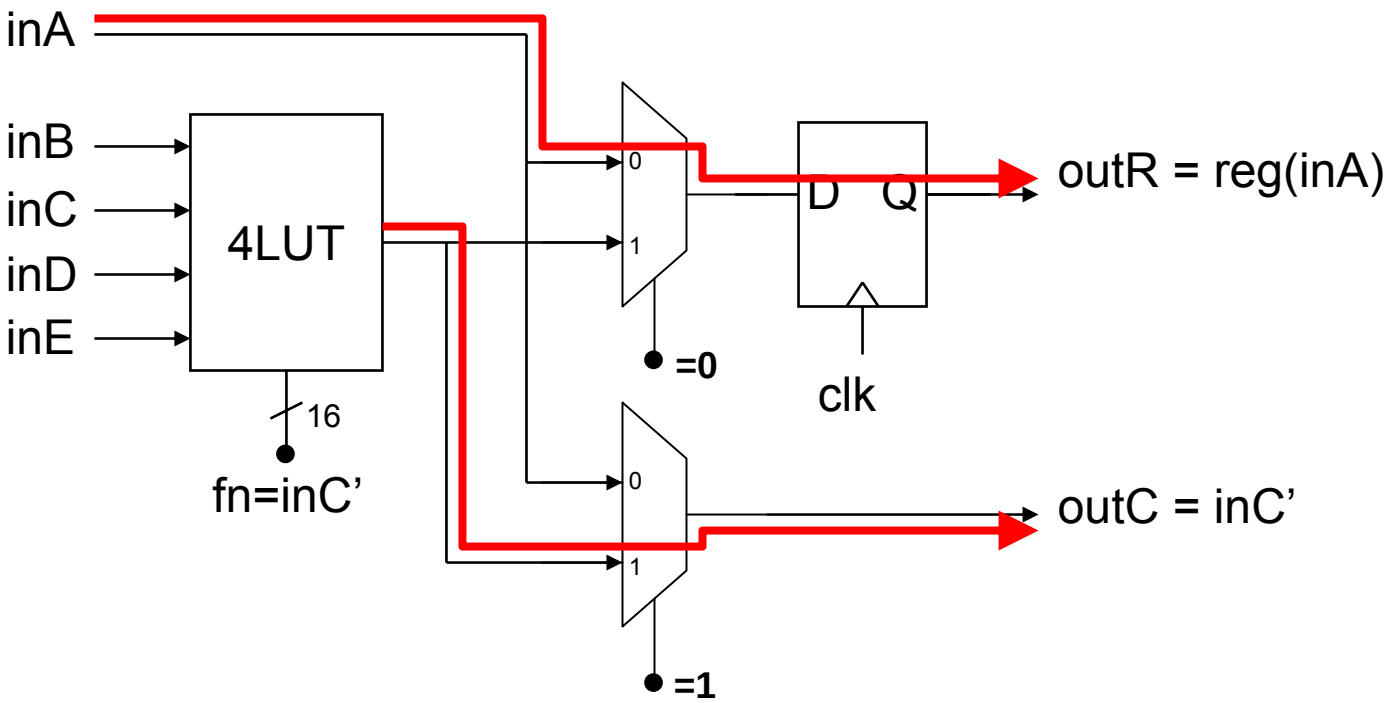
- Provides two outputs
  - One combinational, one registered



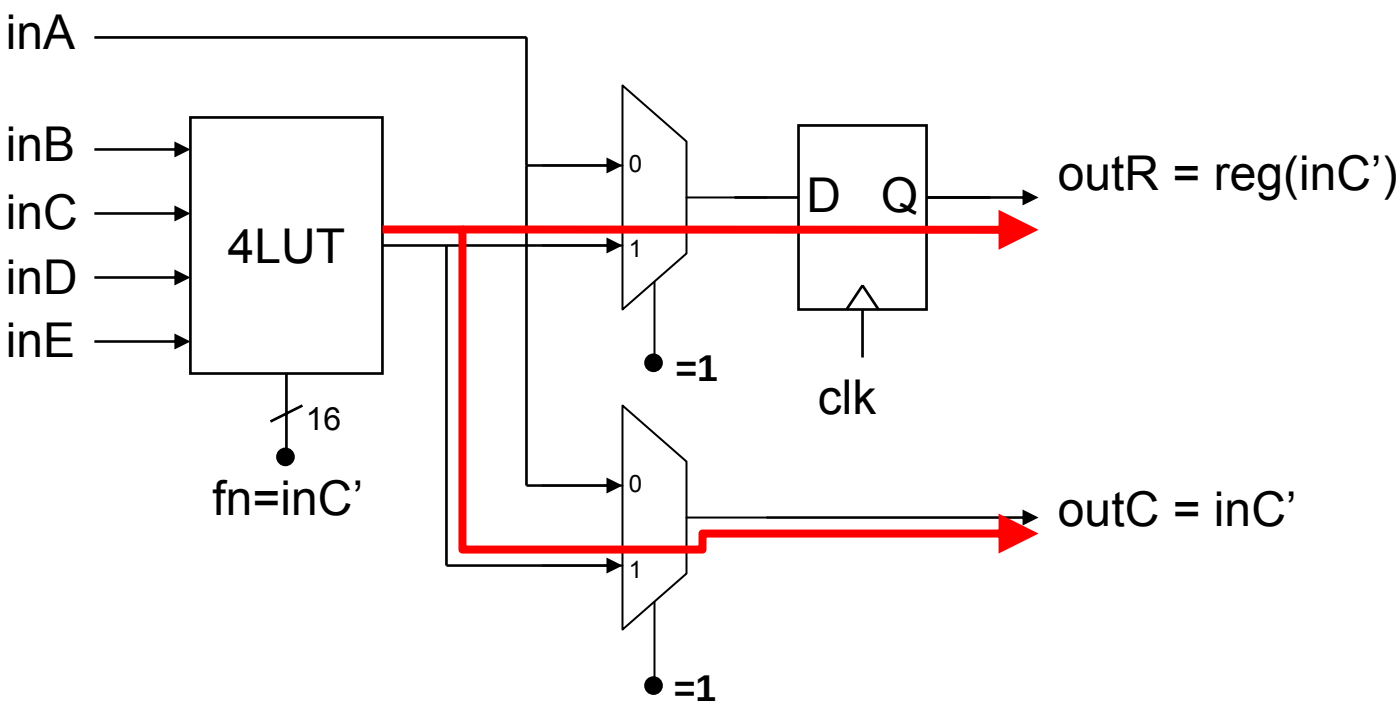
One Configuration



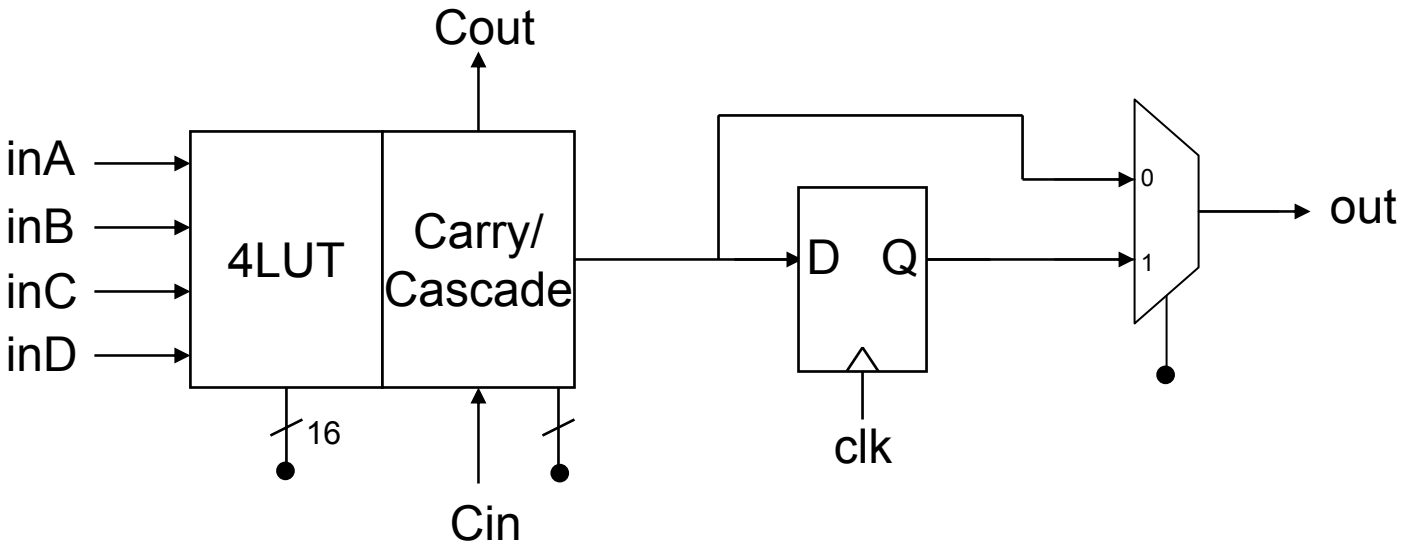
Another Configuration



Yet Another Configuration



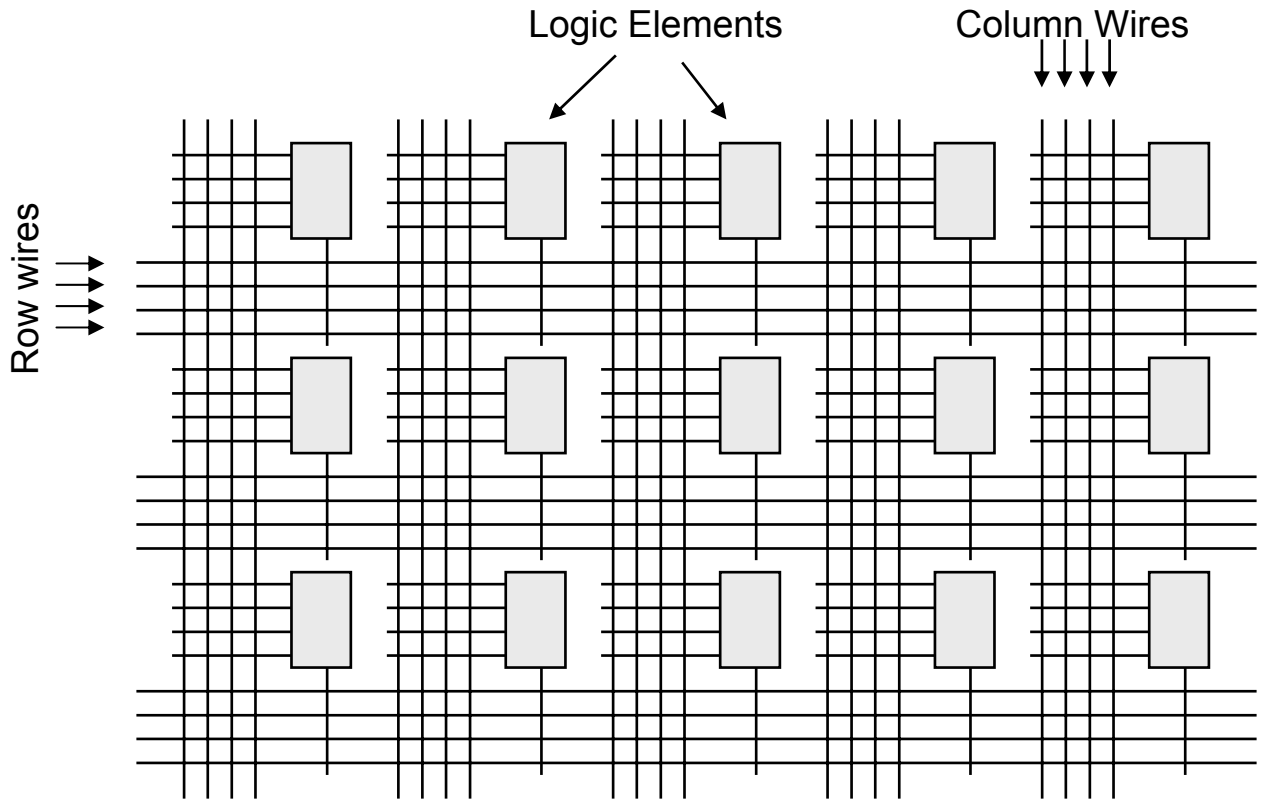
An LE With Carry/Cascade Logic



Can do two functions at once (cout and Cout)
Carry/cascade logic optimized for add/subtract + wide AND,OR, …

Cin and Cout have dedicated connections to neighboring LEs ⇔ fast carry chains ⇔ fast arithmetic

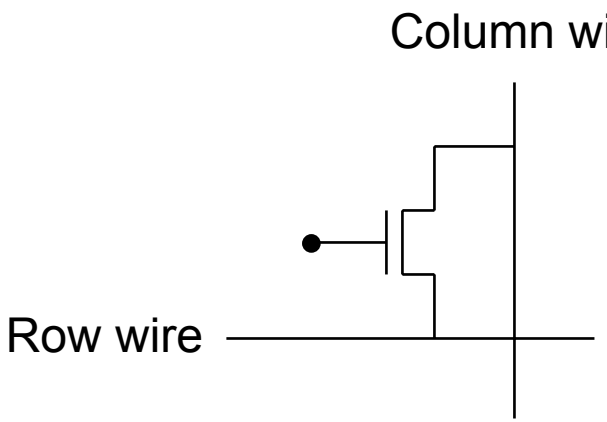
An FPGA Architecture (Island Style)

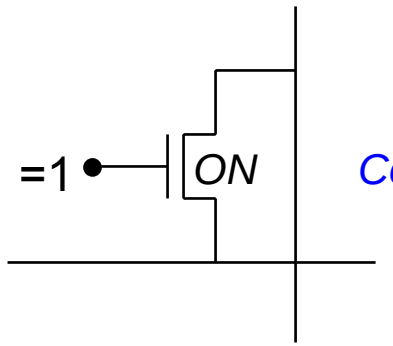


Each LE is configured to do a function

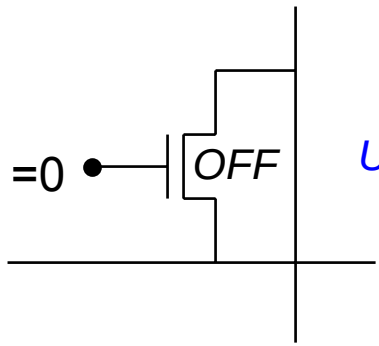
Wire intersections are programmed to either connect or not

Programmable Interconnect Junction





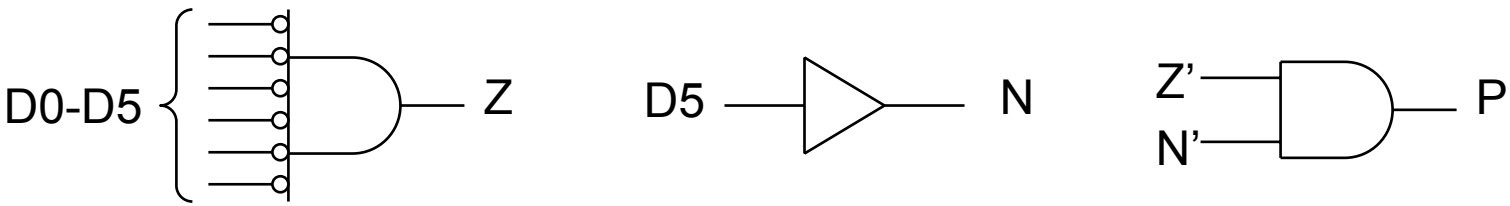
Connected



Unconnected

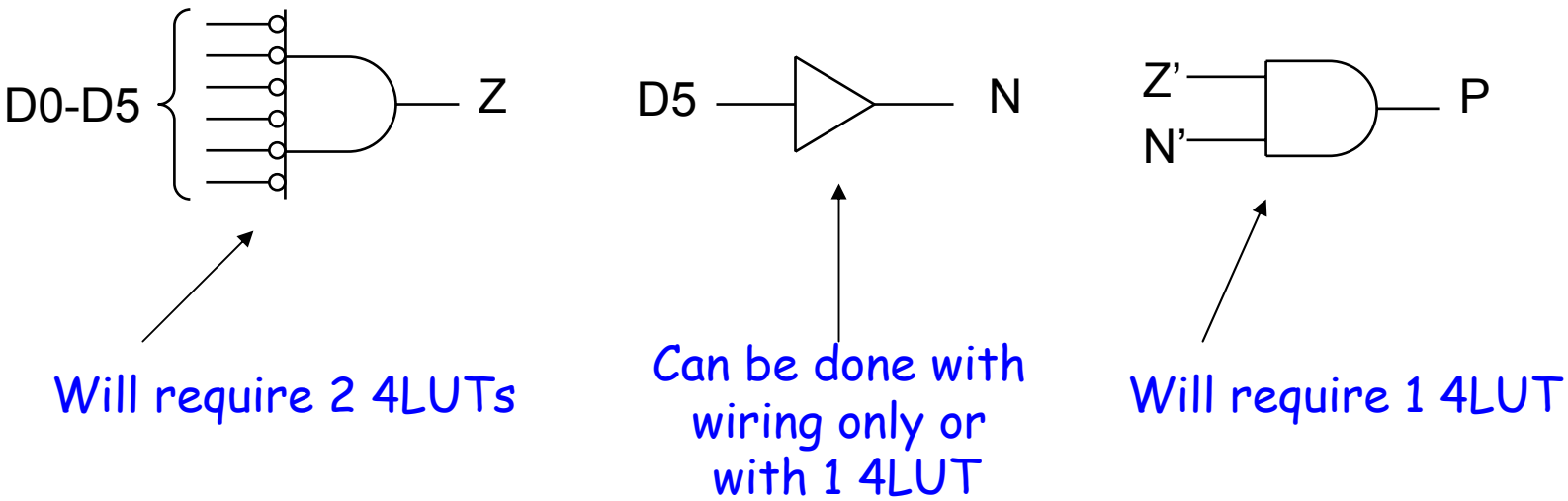
Example Problem

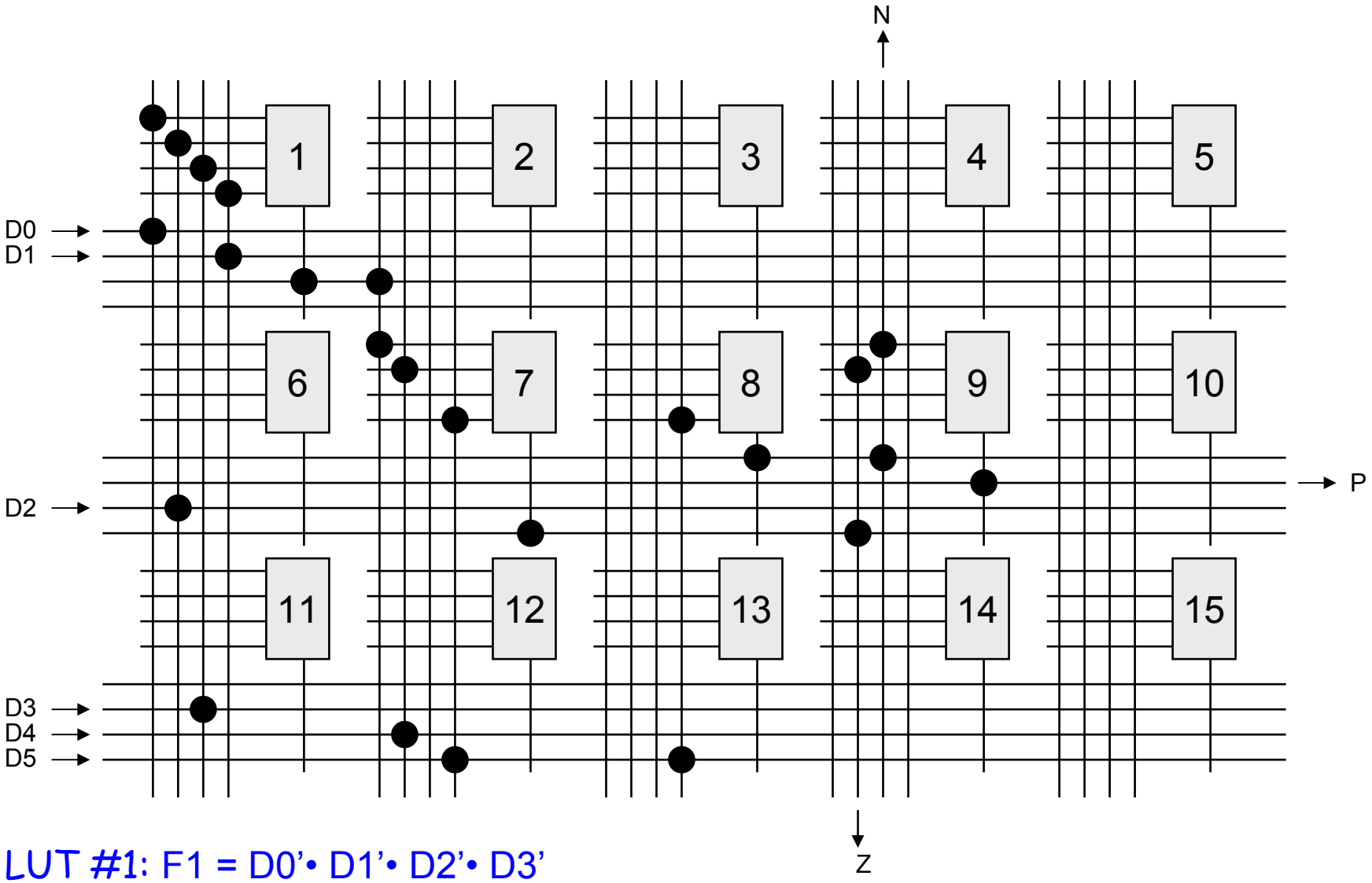
- Generate the N, Z, P status flags for a microprocessor



Example Problem

- Generate the N, Z, P status flags for a microprocessor



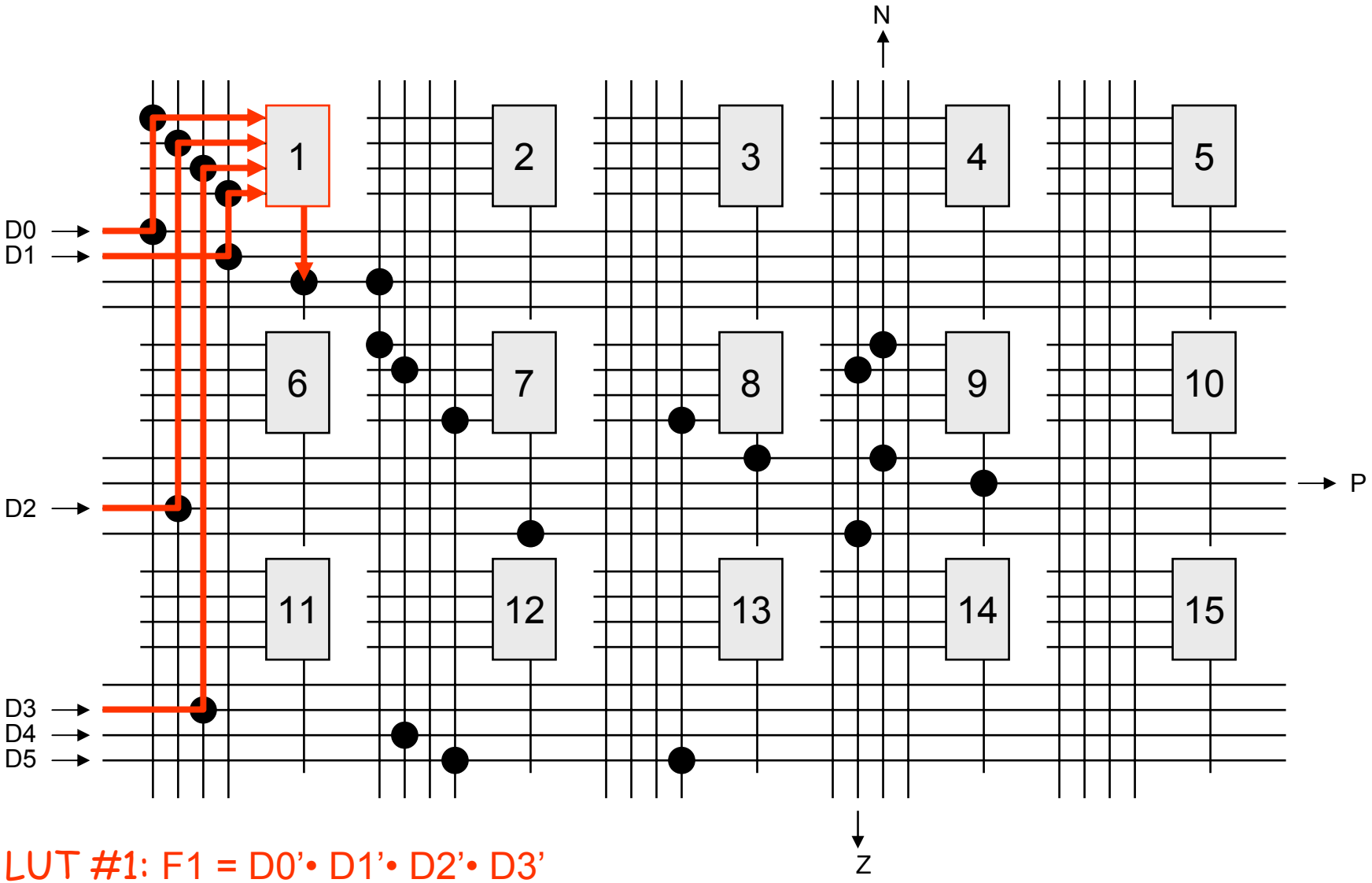


LUT #1: F1 = D0'• D1'• D2'• D3'

LUT #7: F2 = F1•D4'•D5' ⇔ Z output

LUT #8: F3 = D5 ⇔ N output

LUT #9: F4 = Z' • N' ⇔ P output

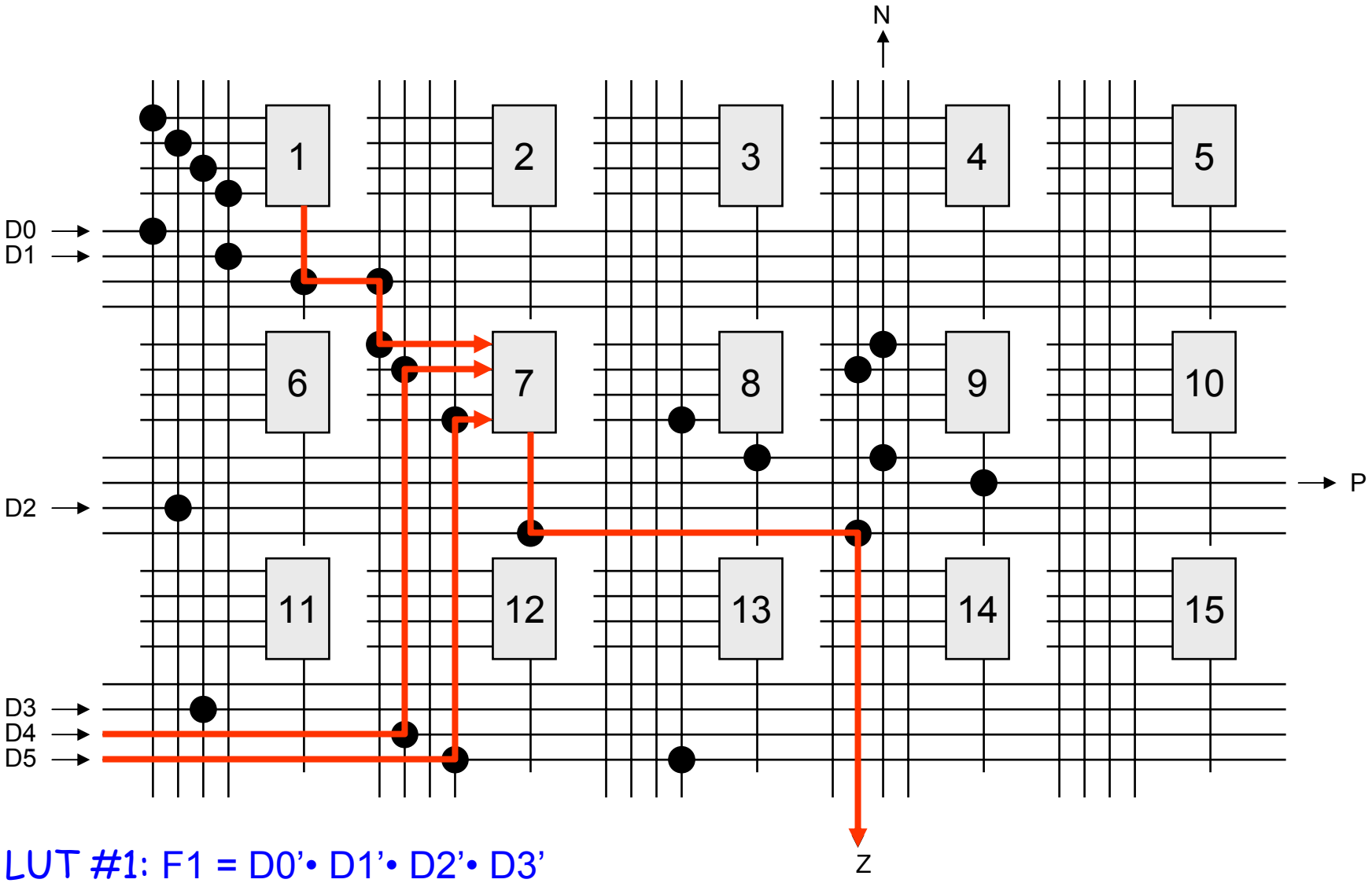


LUT #1: F1 = D0'• D1'• D2'• D3'

LUT #7: F2 = F1•D4'•D5' ⇔ Z output

LUT #8: F3 = D5 ⇔ N output

LUT #9: F4 = Z' • N' ⇔ P output

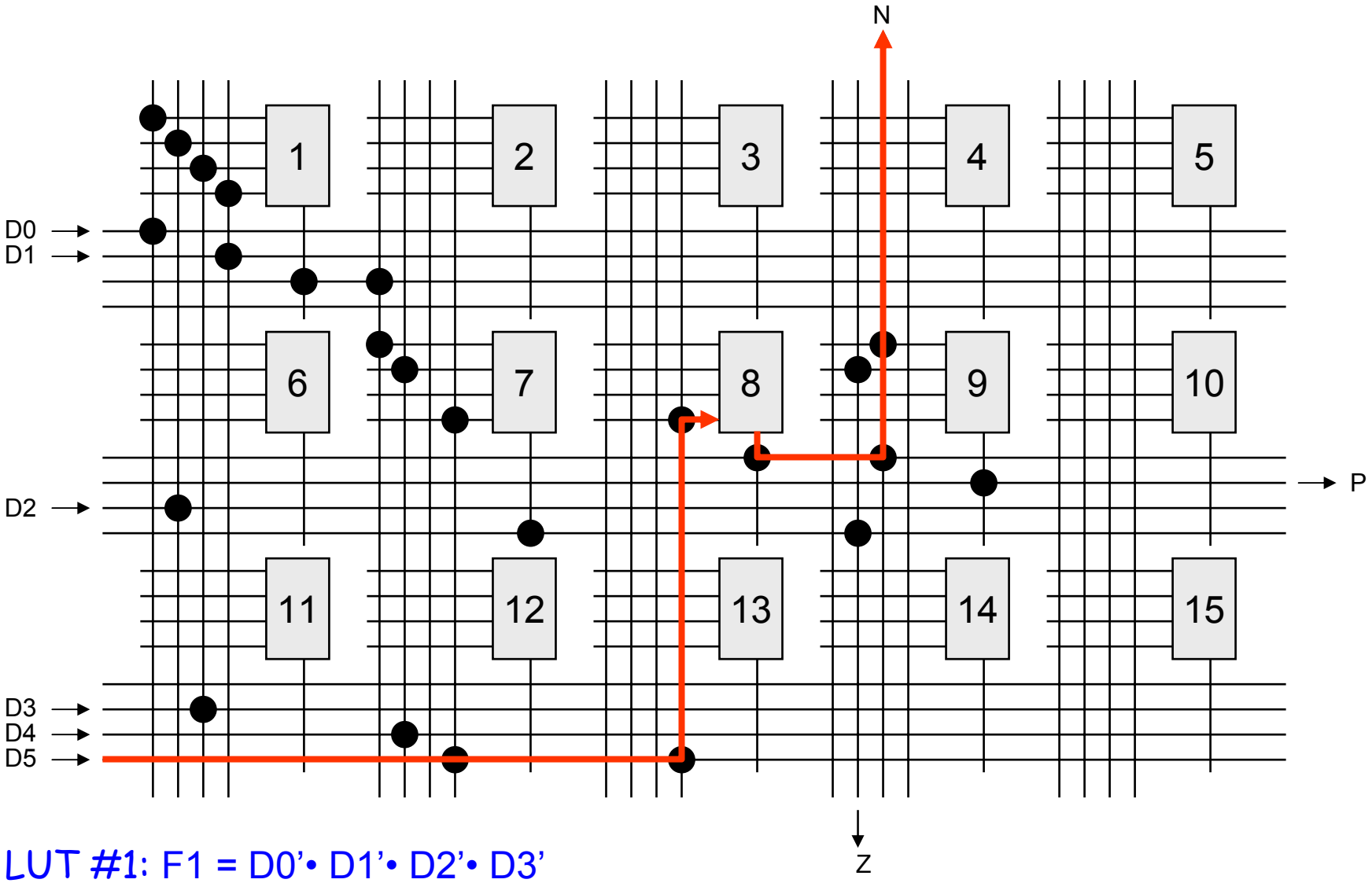


LUT #1: F1 = D0'• D1'• D2'• D3'

LUT #7: F2 = F1•D4'•D5' ⇔ Z output

LUT #8: F3 = D5 ⇔ N output

LUT #9: F4 = Z' • N' ⇔ P output

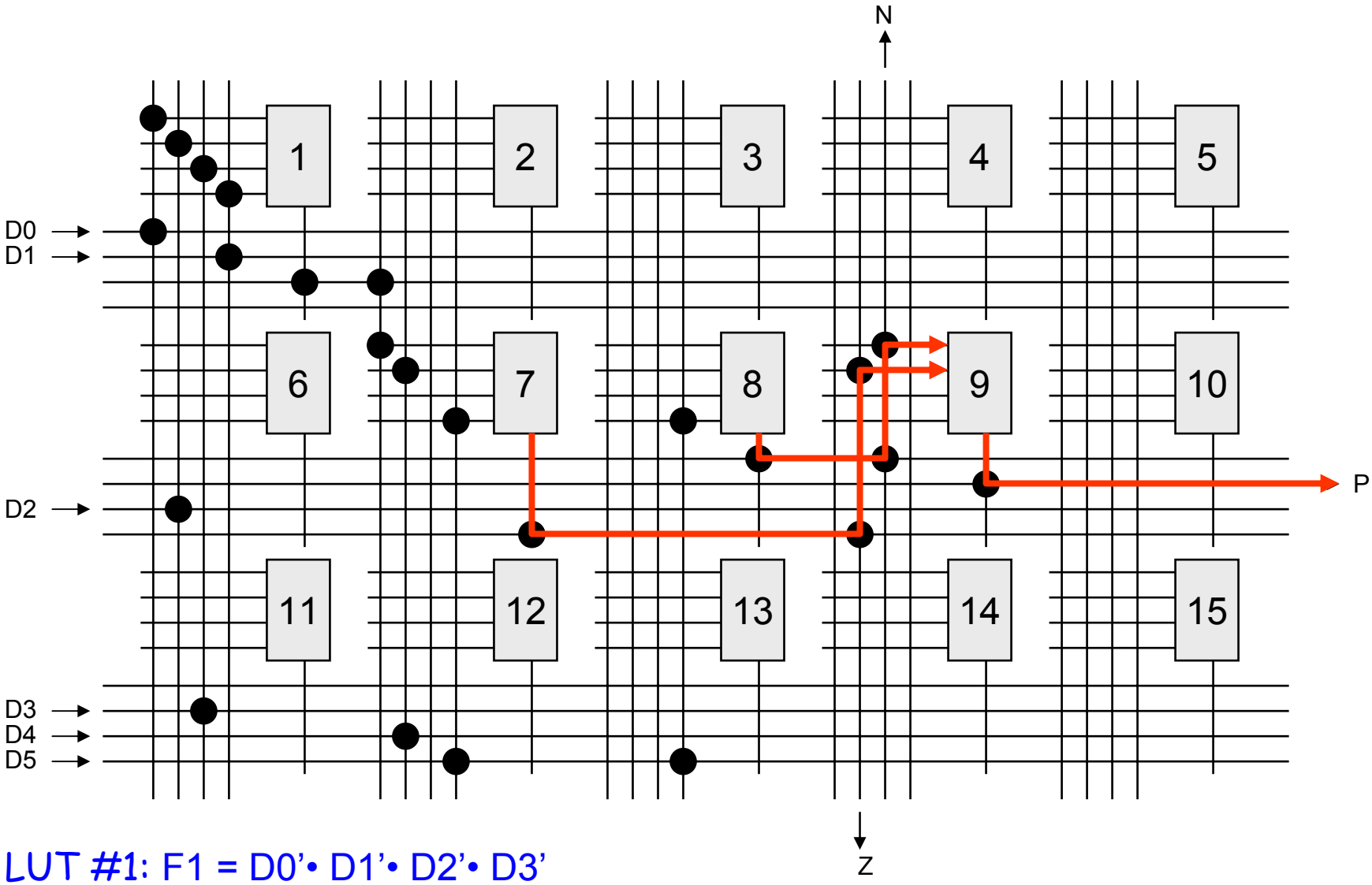


LUT #1: F1 = D0'• D1'• D2'• D3'

LUT #7: F2 = F1•D4'•D5' ⇔ Z output

LUT #8: F3 = D5 ⇔ N output

LUT #9: F4 = Z' • N' ⇔ P output



LUT #1: F1 = D0'• D1'• D2'• D3'

LUT #7: F2 = F1•D4'•D5' ⇔ Z output

LUT #8: F3 = D5 ⇔ N output

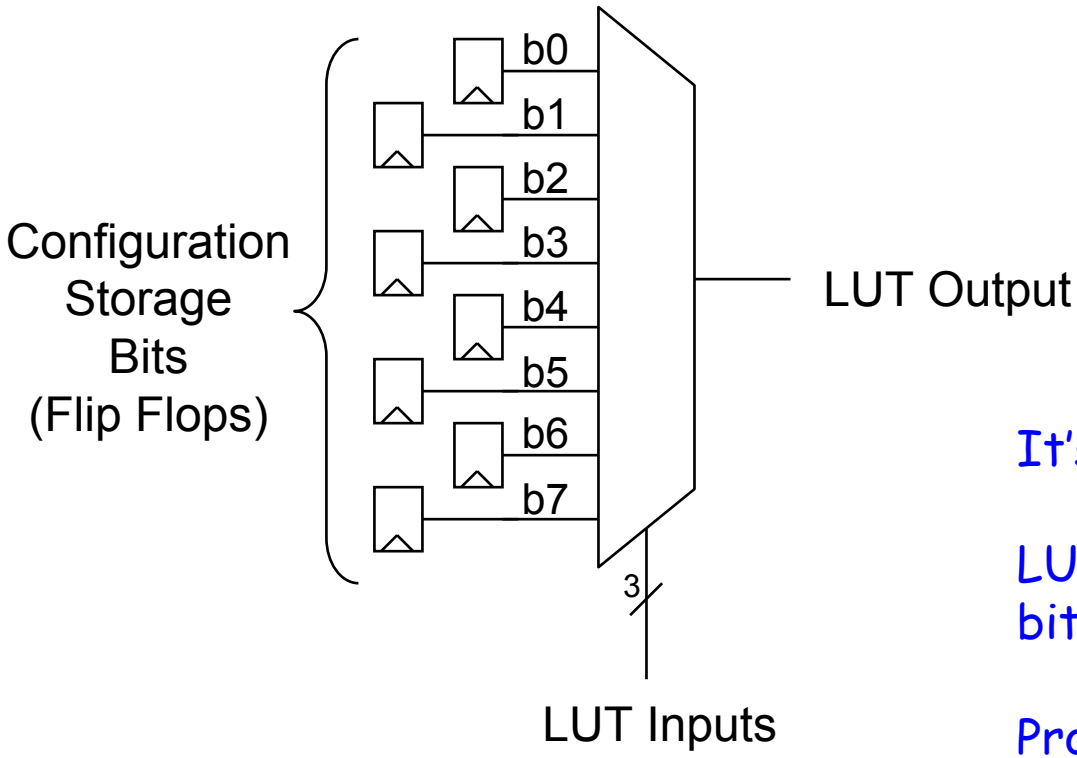
LUT #9: F4 = Z' • N' ⇔ P output

Configuring an FPGA

- An FPGA contains a configuration pin
  - Configuration bits are shifted into FPGA using this pin, one bit per cycle
  - Configuration bits in FPGA linked into a long shift register (SIPO)

- Examples on following slides ⇔ *conceptual*
  - Commercial devices slightly different

Structure of a 3LUT

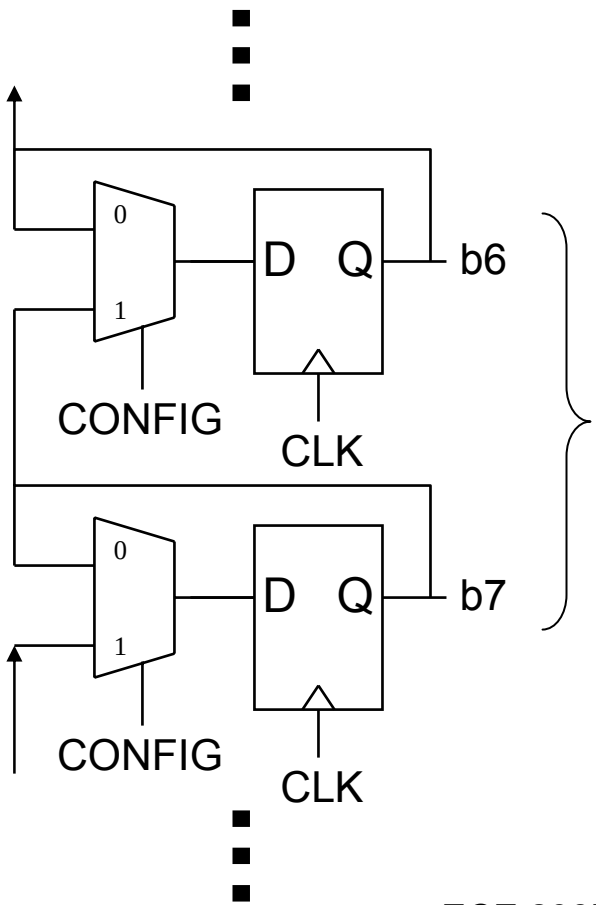


It's just an 8:1 MUX

LUT inputs select which config bit is sent to LUT output

Programming LUT function ⇔ setting configuration bits

How are the Configuration Bit Flip Flops Loaded?

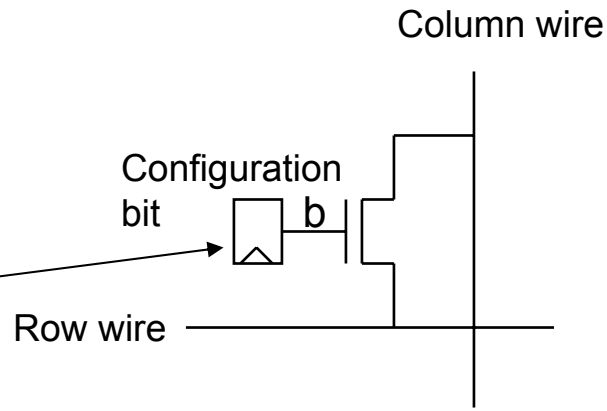


A serial-in/parallel-out (SIPO) shift register

These are the configuration bits which the LUT selects from

Configuring the Programmable Interconnect

Arranged in a SIPO shift register also



Additional FPGA Features

Found in commercial FPGAs

Configurable Input/Output

- I/O buffers are at periphery of chip
  - Some will be inputs, some will be outputs
    - Set when chip configured
- Configurable I/O cell
  - Input static discharge protection
  - Input signal conditioning (voltage levels, …)
  - Output drive
    - Fast, slow
    - Tri-state
  - Handles a variety of electrical standards
    - LVTTL
    - SGT
    - …
- Configured at same time rest of chip is configured

Configuration Storage

- Actual configuration storage is *not* flip flops
  - Use SRAM memory cells
- Many chips can be *partially configured*
  - While rest of chip is running
- Different technologies
  - Program once
    - Fuse, anti-fuse configuration
  - Program multiple times
    - SRAM-based configuration

Programmable Interconnections

- Expensive to put programmable connections at *every* wire junction
  - Commercial parts ⇔ partially populate junctions
  - No or little loss of routing flexibility

Hierarchical Routing

- Collection of short, medium, and long wires
  - Both rows and columns
- Signals use wires that go distance needed
- CAD tools make determinations
- Like alleys, streets, expressways in a city
  - Longer distance ⇔ more limited access points

Clustered LE's

- Cluster: >= 1 LE
    - Example: clump of 4
  - Dedicated wires within groups of 4 LE's
- Closely related to hierarchical routing
- Higher performance
- Different vendors ⇔ different clustering

Embedded Function Units

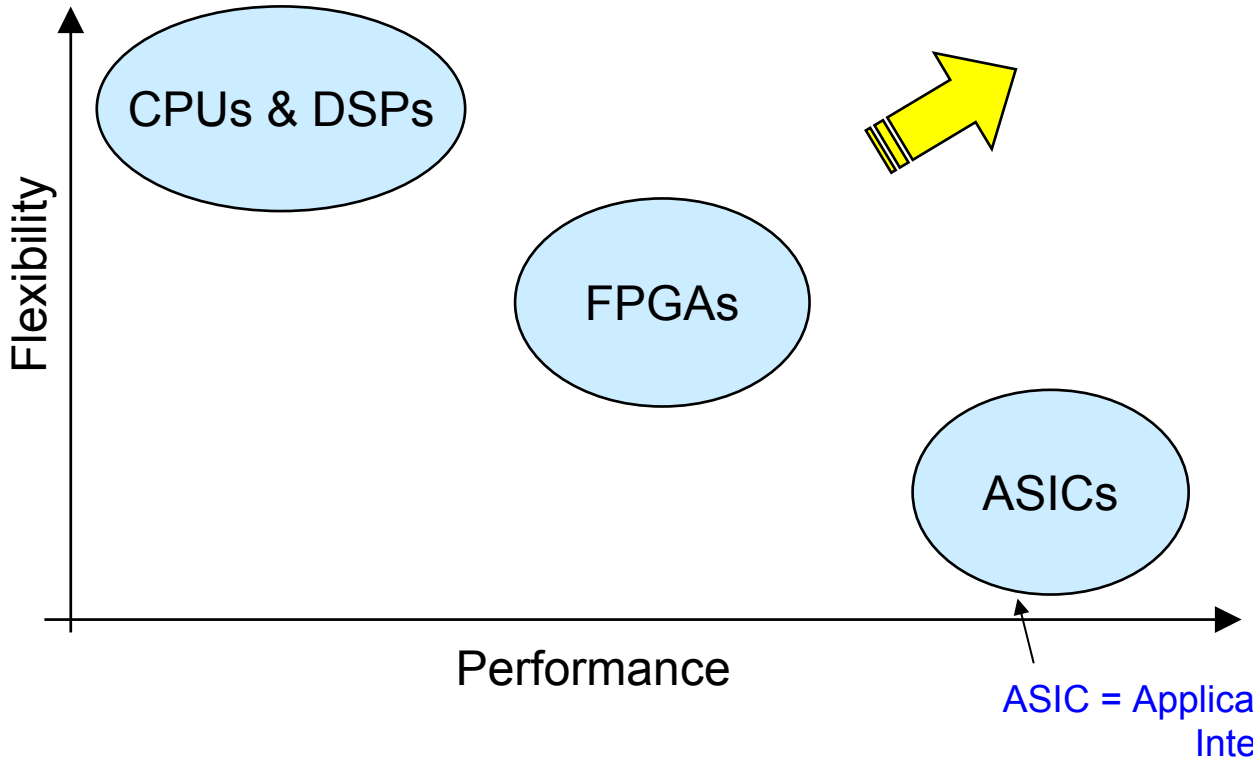
- Place the following into the FPGA fabric:

  multipliers
  memories
  I/O interfaces (*e.g.* gigabit serial links or ethernet PHY/MAC)
  CPU's (PowerPC, ARM, …)

- Result: higher speed, higher density designs

FPGAs Compared To Other Technologies

Performance vs. Flexibility



Goal: the performance of ASIC's with the flexibility of programmable processors.

FPGAs vs. CPUs – Flexibility

- Any function can be programmed to run on a CPU
  - Programming is relatively simple to do
- Any function can also be configured onto an FPGA
  - Design is significantly harder
    - It is digital logic design
- Both CPUs and FPGAs can be reprogrammed (reconfigured) in the field

- Advantage: CPUs

FPGAs vs. CPUs – Performance

- 10+ years of research ⇔ 10x-100x performance advantage for FPGAs
- Custom hardware ⇔
  - No fetch-decode-memoryAccess overhead
  - Significant parallelism and pipelining
- Case in point: 1998 SONAR beamformer on an FPGA
  - BYU Configurable Computing Lab research
  - 2 x 10^9 FLOP/sec *sustained*
  - 10x-80x faster than comparable CPU technology

- Advantage: FPGAs

FPGAs vs. ASICs - Flexibility

- ASIC is a *static* hardware design
  - Wires and transistors fixed at manufacture time
  - Cannot be upgraded (reprogrammed) in the field
- FPGA can be configured (and reconfigured)
  - FPGA platform ⇔ used for a variety of applications
    - CCM = Custom Computing Machine (based on FPGAs)

- Advantage: FPGAs

FPGAs vs. ASICs – Performance

- Both ASICs and FPGAs are *custom* hardware designs
- FPGA *configurability* comes at a cost
  - FPGA density = 1/10th to 1/100th of an ASIC
- Result:
  - Higher cost chip for FPGA
  - Higher performance/silicon area for ASIC
  - For some/many applications:
    - FPGA just not fast/big enough
- Advantage: ASICs

FPGAs vs. ASICs – Cost

- Once the design is complete:
  - FPGAs can be programmed in seconds
    - Negligible up-front costs
    - No penalty for small volume
    - Per part cost very high
  - ASICs require time-consuming, expensive manufacturing process
    - Fabrication facility >= $1G
    - Significant up-front costs
    - Penalty for small volume
    - Per part cost very low
- Advantage: ???? (depends on volumes and application)

FPGA vs. ASIC Production Cost Example

- FPGA
  - Non-recurring expenses (NRE) = $0
  - Per-part cost = $20 (prices vary from a few $'s to $100's)
  - Cost to produce 100 = $2K
  - Cost to produce 10^6 = $20M
- ASIC
  - NRE expenses = $5M
  - Per-part cost = $1
  - Cost to produce 100 = $5M + $100 ≈ $5M
  - Cost to produce 10^6 = $6M

Volume produced makes a big difference…

FPGAs vs. ASICs – Additional Points

- Manufacturing Time
  - FPGA = milliseconds for end customer
  - ASIC = weeks to months
- Fixing Bugs
  - FPGA = modify design + reconfigure FPGA
  - ASIC = modify design + remanufacture

- Advantage: FPGA

FPGAs vs. ASICs - Summary

- Two very different technologies
  - Flexibility: FPGA wins
  - Performance: ASIC wins
  - Risk: ?
  - Cost: ?

- Complex Decision
  - Only partially based on technical issues
  - Significant business issues
    - Risk
    - Time-to-market
    - Market characteristics (elasticity, price sensitivity, …)

Design for FPGAs

1. Draw schematics or write HDL
2. CAD tool ⇔ maps to LUTs + FFs
3. CAD tool generates bitstream
4. Load bitstream ⇔ *configure* the FPGA

Design for ASICs

1. Draw schematics or write HDL
2. CAD tool ⇔ map to physical silicon layout
3. Send layout data to silicon *fab*
4. Finished chip does intended function

- Step 1 fairly similar for FPGAs and ASICs
  - Major differences in later steps

FPGA/ASIC Vital Statistics

- A typical UG lab FPGA has:
  - 5,000-10,000 LUTs
  - 200-300 input/output buffers
  - Clock at ≈ 50 MHz

- A large modern CPU (Pentium circa 2004) has:
  - 125 million transistors
  - > 1000 input/output buffers
  - Clock at > 3GHz

For More Information

- FPGA companies' WWW sites tend to have lots of information
  - Catering to a wide range of customers
    - Novice to expert
    - Low volume to high volume
  - Data sheets
  - Application notes
  - Success stories
  - CAD tools