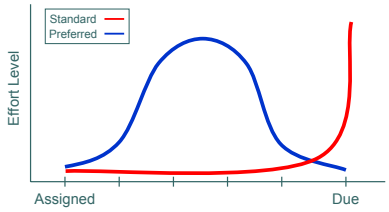
# CprE / ComS 583 Reconfigurable Computing

Prof. Joseph Zambreno
Department of Electrical and Computer Engineering
Iowa State University

Lecture #3 – FPGA Basics

## Quick Points

- HW #1 is out
  - Due Thursday, September 7 (12pm)
  - Submission via WebCT
- Possible strategies






## Recap

- FPGAs – spatial computation
- CPU – temporal computation
- FPGAs are by their nature more computationally "dense" than CPU
  - In terms of number of computations / time / area
  - Can be quantitatively measured and compared
- Capacity, cost, ease of programming still important issues
- Numerous challenges to reconfiguration



## Outline

- Recap
- FPGA Taxonomy
- Lookup Tables and Digital Logic
- Interconnect / Routing Structures
- FPGA Architectural Issues



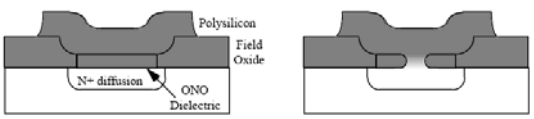
## FPGA Taxonomy

- *Programming technology* – how is the FPGA programmed? Where does it store configuration bits?
  - SRAM
  - Anti-fuse
  - EPROM
  - Flash memory (EEPROM)
- *Logic cell architecture* – what is the granularity of configurable component? Tradeoff between complexity and versitility
  - Transistors
  - Gates
  - PAL/PLAs
  - LUTs
  - CPUs
- *Interconnect architecture* – how do the logic cells communicate?
  - Tiled
  - Hierarchical
  - Local



## Anti-Fuse Technology

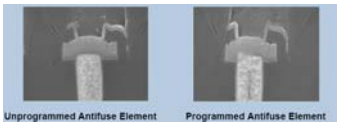
- Dielectric that prevents current flow
- Applying a voltage melts the dielectric




- One time programmable – not really *reconfigurable* computing

## Anti-Fuse Technology (cont.)

- Negligible programming overhead
- Fast routing
- Nonvolatile (hold data after power off)
- High level of security:
  - No bitstream can be intercepted in the field
  - Need a Scanning Electron Microscope (SEM) to figure out the anti-fuse states

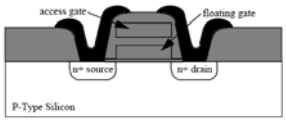



© Actel



## E(E)PROM Technology

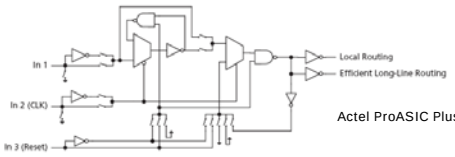
- To program a transistor, a voltage differential between the access/floating gates accelerates electrons from the source fast enough to travel across the insulator to the floating gate
- Electrons prevent the access gate from closing
- EPROM – Erasable Programmable Read-Only Memory
  - Nonvolatile
  - Can be erased using UV light
- EEPROM – Electrically Erasable Programmable Read-Only Memory
  - Removes the electrons by reversing the voltage differential
  - Limited number of erases possible
  - Precursor to Flash technology






## Flash / EEPROM Devices

- Migrated from early PLD technology
- Traditionally based on AND/OR architecture
- High ratio of logic-to-registers
- Future of Flash devices?
  - Logic elements (LUTs and flip flops)
  - Segmented routing
  - Low logic to register ratio

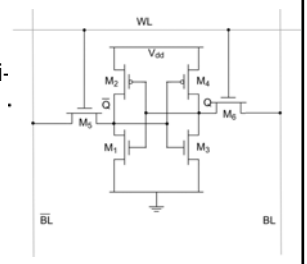



Actel ProASIC Plus logic cell



## SRAM Technology

- SRAM – Static Random Access Memory
- SRAM cells are larger (6 transistors) than anti-fuse or EEPROM
  - Slower
  - Less computational power per $\lambda^2$
- SRAM bits can be programmed many times





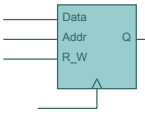

## Which Devices to Study?


- SRAM – ~95%, Flash/Anti-fuse – ~5%



## Lookup Tables (LUTs)

- What is a Lookup Table (LUT)?
  - In most generic terms, a pre-loaded memory



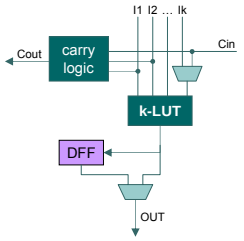

```
int table[256] = {1,7,8,9,…14};
int Q, addr;
...
Q = table[addr];
```

  - Great way of implementing some function without calculation – a **cheat sheet**

Q7: What is the answer to life, the universe, and everything?

Answer Key
.
.
.
A(Q7) = **42**

## LUTs and Digital Logic




- Each *k*-LUT operates on *k* one-bit inputs
- Output is one data bit
- Can perform <u>any</u> Boolean function of *k* inputs



## LUTs and Digital Logic (cont.)

- *k* inputs → $2^k$ possible input values
- *k*-LUT corresponds to $2^k$ x 1 bit memory
- How many different functions?
  - Two-input ($A_1$,$A_2$) case
  - F($A_1$,$A_2$) = {0, $A_1$ *nor* $A_2$, $\bar{A}_1$ *and* $A_2$, $\bar{A}_1$, $A_1$ *and* $\bar{A}_2$, $\bar{A}_2$, $A_1$ *xor* $A_2$, $A_1$ *nand* $A_2$, $A_1$ *and* $A_2$, $A_1$ *xnor* $A_2$, $A_2$, $\bar{A}_1$ *or* $A_2$, $A_1$, $A_1$ *or* $\bar{A}_2$, $A_1$ *or* $A_2$, 1} = 16 different possibilities
- What to store in the LUT?

| $A_1$ $A_2$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |



## LUTs and Digital Logic

- F(input) can be 0 or 1 independently for each of the $2^k$ bits
  - $2^{2^k}$ possible functions
  - Not all patterns are unique (*k*! permutations of the inputs)
  - Approximately $2^{2^k}$ / *k*! unique functions

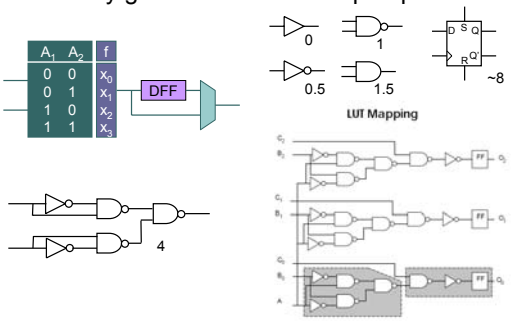
| $A_1$ $A_2$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

- Is this efficient?
- How to select *k* value?



## LUT Mapping

- How many gates in a 2-LUT + flip-flop?

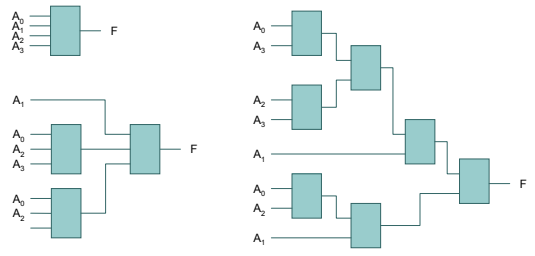





## LUT Mapping (cont.)

- Implement the following logic function with *k*-LUTs, for k={2, 3, 4}:

$$F = A_0A_1A_3 + A_1A_2\bar{A}_3 + \bar{A}_0\bar{A}_1\bar{A}_2$$



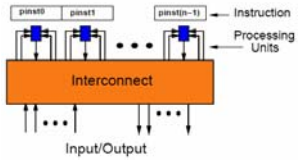



## Logic Block Granularity

- Each *k*-LUT requires $2^k$ configuration bits:
  - The 2-LUT implementation requires $2^2$ x 7 = 28 bits
  - The 3-LUT needs $2^3$ x 3 = 24 bits
  - The 4-LUT needs just $2^4$ x 1 = 16 bits
  - Using configuration bits as area measure (area cost), the 4-LUT implementation achieves minimum logic area

## Interconnect

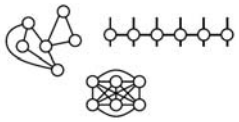
- Problem:
  - Thousands of operators producing results
  - Each taking as outputs the results of other bit operators
  - Initial assumptions – have to connect them all *simultaneously*






## Interconnect Design Issues

- *Flexibility* – route anything (within reason?)
  - Bisection bandwidth
  - Simultaneous routes
- *Area*
  - Bisection bandwidth
  - Switches
- *Delay* (and *Power*)
  - Switches in path
  - Wire length
- *Routability* – difficulty of finding a route





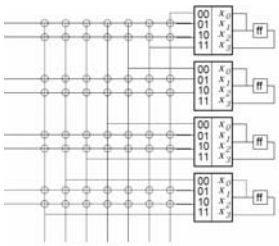
## General Routing Architecture

- A *wire segment* is a wire unbroken by programmable switches
  - Typically one switch is attached to each end of a wire segment
- A *track* is a sequence of one or more wire segments in a line
- A routing channel is a group of parallel tracks
- A *connection block* provides connectivity from the inputs and outputs of a logic block to the wire segments in the channels



## Attempt 1 – Crossbar

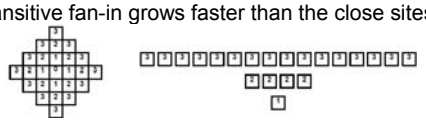
- Any operator may want to take as input the output of any other operator
- Let $n$ be the number of LUTs, and $l$ the length of wire
  - Delay:
    - Parasitic loads = $kn$
      Switches/path = $l$
      Wire length = $O(kn)$
      Delay = $O(kn)$
  - Area:
    - Bisection BW = $n$
      Switches = $kn^2$
      Area = $O(n^2)$



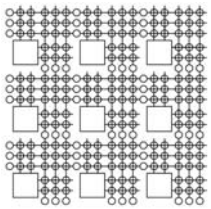


## Attempt 2 – Mesh

- Put connected components together
- Transitive fan-in grows faster than the close sites:

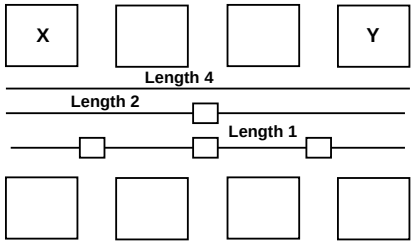


- Let $w$ be the channel width
  - Delay:
    - Switches/path = $l$
      Wire length = $O(l)$
      Best Delay = $O(w)$
      Worst Delay = $O(n^{1/2}w)$
  - Area:
    - Bisection BW = $wn^{1/2}$
      Switches = $O(nw^2)$
      Area = $O(nw^2)$





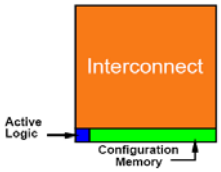
## Segmentation




- Segmentation distribution: how many of each length?
- Longer length
  - Better performance?
  - Reduced routability?

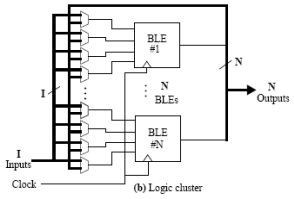
## Interconnect Area/Delay

- Interconnect area = ~10x configuration memory = ~100x logic
- The interconnect constitutes ~90% of the total area and 70-80% of the total delay
- See [Deh96A] for details






## Architectural Issues [AhmRos04A]




- What values of N, I, and K minimize the following parameters?
  - Area
  - Delay
  - Area-delay product
- Assumptions
  - All routing wires length 4
  - Fully populated IMUX
  - Wiring is half pass transistor, half tri-state
  - 180 nm
  - Routing performed with $W_{min}$ + 30% tracks



## Summary

- Three basic types of FPGA devices
  - Antifuse
  - EEPROM
  - SRAM
- Key issues for SRAM FPGA are logic cluster, connection box, and switch box.
- Most tasks have structure, which can be exploited by LUT arrays with programmable interconnect (FPGAs)
- Cannot afford full interconnect, or make all interconnects local