# Introduction to VLSI Design
# Logic Arrays

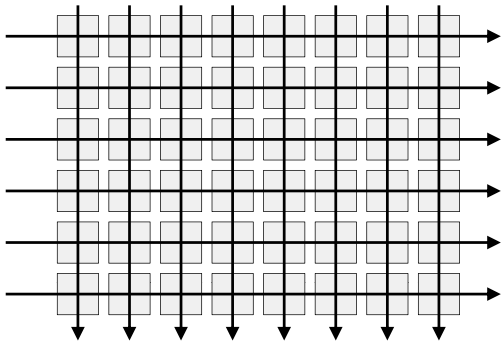
Tsung-Chu Huang
Electronics Eng., NCUE
2016/5/10

## Arrays

- Addressability
  - Addressable & Non-address
- Cell Unit
  - Components (Test Key)
  - Transistor switch (Weinberger, SOG, PLA)
  - Logic gates (Random Access Scan)
  - PLA, LUT, CLE, CLB (FPGA)
  - Storage Cell (Memory)
- Architecture
  - Centralized, Distributed
  - Discrete, Assembled, Embedded

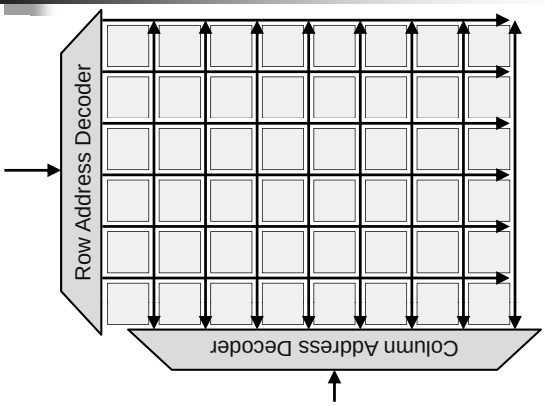


## Non-Addressable Arrays





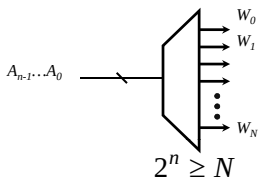
## Addressable Arrays






## Function of n-to-N Address Decoders

*Binary Code, Gray-Codes, or other codes*

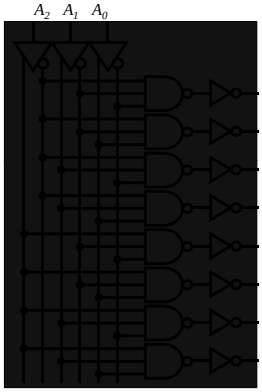



*1-Hot Codes in Binary Order, Gray-Coded Order, or other orders*

$2^n \geq N$

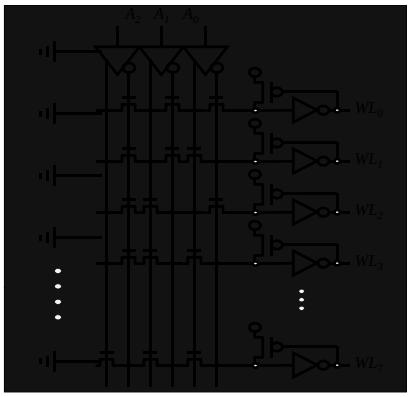


## Gate-Level Design of Address Decoders




- *The NAND can be substituted by NOR gates*
- *The gates can be CMOS cells in cell-based design, or diffusion stripes in transistor-level design.*

## Pseudo-NMOS Transistor-Level Single-Diffusion Design of Address Decoders



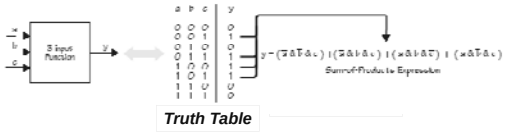


## Evolution of implementation technologies

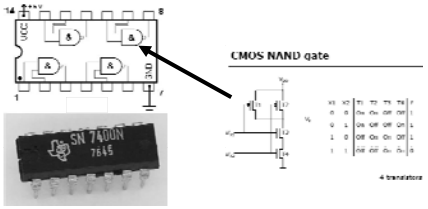
- 50's: Logic gates
- 60's: Regular structures for two-level logic
  - muxes and decoders, PLAs
- 70's: Programmable sum-of-products arrays
  - PLDs, complex PLDs ➔ PAL, GAL, SPLD
- 80's: Programmable gate arrays
  - densities high enough to permit entirely new class of application, e.g., prototyping, emulation, acceleration: FPGA ➔ PLD
- 90's: Field Programmable Logic Arrays
  - FPGA ~ CPLD



## Digital Logic



*Truth Table*



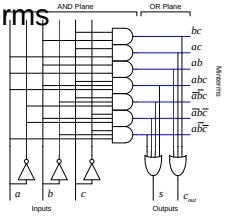



## PLAs

- A *Programmable Logic Array* performs any function in sum-of-products form.
- *Literals*: inputs & complements
- *Products / Minterms*: AND of literals
- *Outputs*: OR of Minterms
- Example: Full Adder

$$s = a\bar{b}\bar{c} + \bar{a}b\bar{c} + \bar{a}\bar{b}c + abc$$

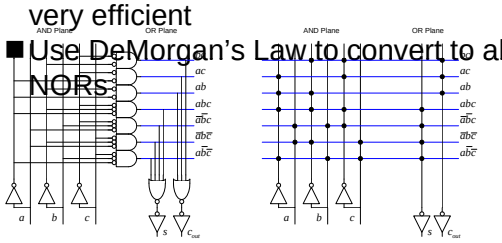
$$c_{\text{out}} = ab + bc + ac$$





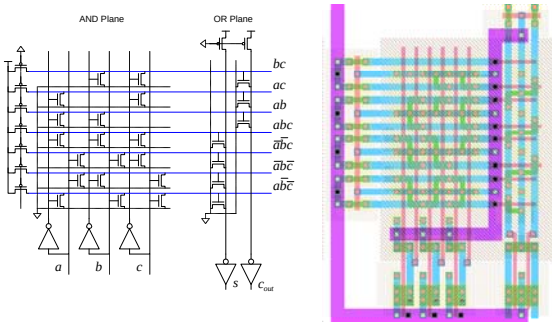

## NOR-NOR PLAs

- ANDs and ORs are not very efficient in CMOS
- Dynamic or Pseudo-nMOS NORs are very efficient
- Use DeMorgan's Law to convert to all NORs






## PLA Schematic & Layout



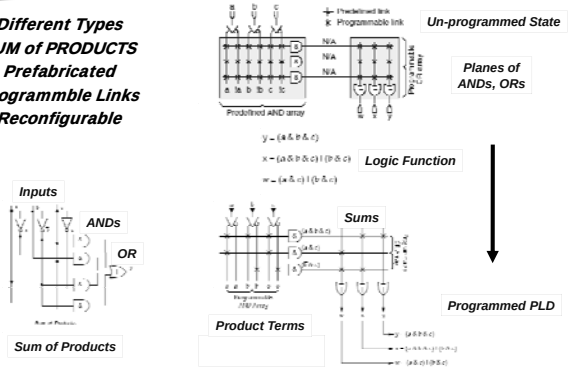
## PLAs vs. ROMs

- The OR plane of the PLA is like the ROM array
- The AND plane of the PLA is like the ROM decoder
- PLAs are more flexible than ROMs
  - No need to have $2^n$ rows for n inputs
  - Only generate the minterms that are needed
  - Take advantage of logic simplification



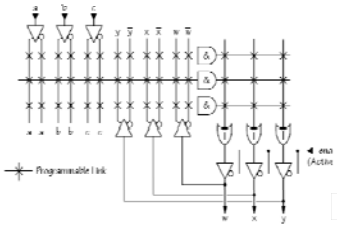
## Programmable Logic Devices PLDs

- ***Different Types***
- ***SUM of PRODUCTS***
- ***Prefabricated***
- ***Programmble Links***
- ***Reconfigurable***

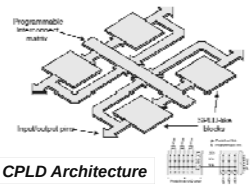





## PAL ➔ GAL ➔ CPLD



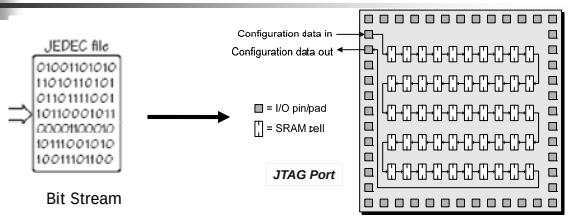
***PAL® Structure***



***CPLD Architecture***



## Configuring an FPGA




- Millions of SRAM cells holding LUTs and Interconnect Routing
- Volatile Memory. Loses configuration when board power is turned off.
- Keep Bit Pattern describing the SRAM cells in non-Volatile Memory e.g. ROM or Digital Camera card
- Configuration takes ~ secs



## Layout EDA in General

**Technology**

1. PCB
   1. TTL 74XXX
   2. Discrete Components
2. VLSI
   1. Full-Custom
   2. Cell-Based
3. FPGA
   1. Embedded Processor
   2. Embedded Memory
   3. Embedded ASIC
   4. CLB
      1. Slices
         1. LUT (Lookup Table)

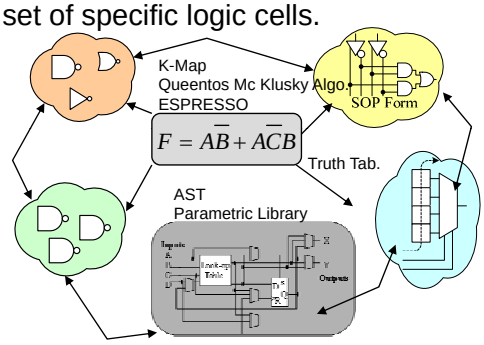
**Flow**

1. Floor-plan
2. Partition
   1. Modularization
3. Placement
   1. Technology mapping
   2. Clustering
   3. Placement
4. Routing
   1. Channel Routing
   2. Global Routing



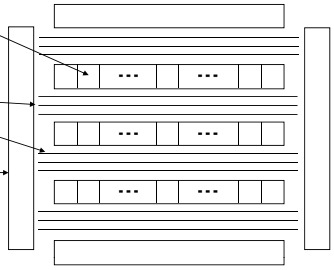
## Technology Mapping

- Mapping generic Boolean functions to a set of specific logic cells.



## Gate Array Technology (IBM - 1970s)

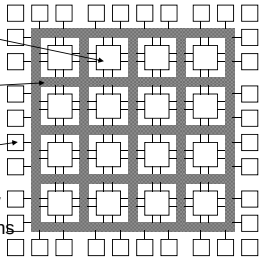
- Simple logic gates
  - combine transistors to implement combinational and sequential logic
- Interconnect
  - wires to connect inputs and outputs to logic blocks
- I/O blocks
  - special blocks at periphery for external connections
- Add wires to make connections
  - done when chip is fabbed
    - ✓"mask-programmable"
  - construct any circuit





## Field-Programmable Gate Arrays

- Logic blocks
  - to implement combinational and sequential logic
- Interconnect
  - wires to connect inputs and outputs to logic blocks
- I/O blocks
  - special logic blocks at periphery of device for external connections
- Key questions:
  - how to make logic blocks programmable?
  - how to connect the wires?
  - *after the chip has been fabbed*





## Enabling Technology

- Cheap/fast fuse connections
  - small area (can fit lots of them)
  - low resistance wires (fast even if in multiple segments)
  - very high resistance when not connected
  - small capacitance (wires can be longer)
- Pass transistors (switches)
  - used to connect wires
  - bi-directional
- Multiplexors
  - used to connect one of a set of possible sources to input
  - can be used to implement logic functions



## Programming Technologies

- Fuse and anti-fuse
  - fuse makes or breaks link between two wires
  - typical connections are 50-300 ohm
  - one-time programmable
- Flash
  - High density
  - Process issues
- RAM-based
  - memory bit controls a switch that connects/disconnects two wires
  - typical connections are .5K-1K ohm
  - can be programmed and re-programmed



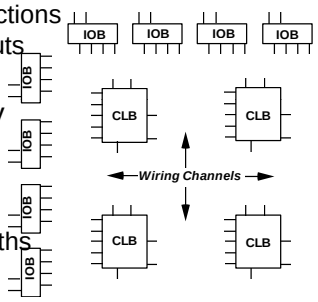
## Tradeoffs in FPGAs

- Logic block - how are functions implemented: fixed functions (manipulate inputs) or programmable?
  - support complex functions, need fewer blocks, but they are bigger so less of them on chip
  - support simple functions, need more blocks, but they are smaller so more of them on chip
- Interconnect
  - how are logic blocks arranged?
  - how many wires will be needed between them?
  - are wires evenly distributed across chip?



## Xilinx Programmable Gate Arrays

- CLB - Configurable Logic Block
  - 5-input, 1 output function
  - or 2 4-input, 1 output functions
  - optional register on outputs
- Built-in fast carry logic
- Can be used as memory
- Three types of routing
  - direct
  - general-purpose
  - long lines of various lengths
- RAM-programmable
  - can be reconfigured



## Overview of Virtex-E Architecture

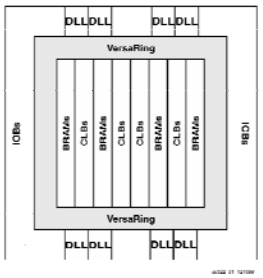



*Figure 1:* **Virtex-E Architecture Overview**



## The Virtex CLB

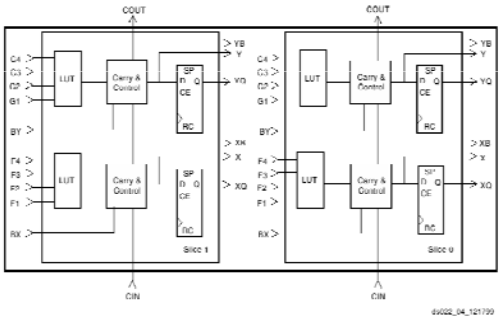



*Figure 4:* **2-Slice Virtex-E CLB**



## Details of One Virtex Slice

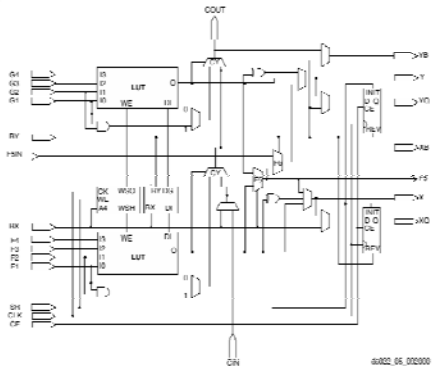
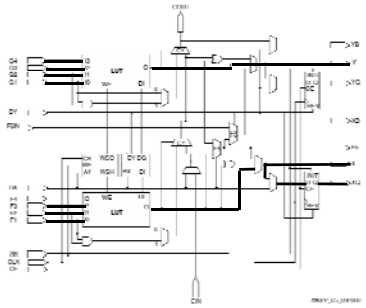



*Figure 5:* **Detailed View of Virtex-E Slice**



## Implements any Two 4-Input Functions



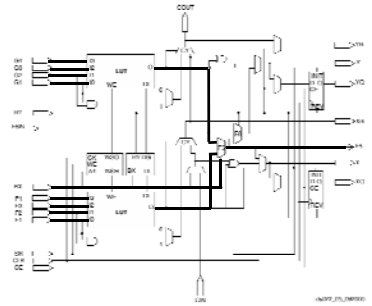
*4-input function*

*3-input function; registered*

*Figure 5:* **Detailed View of Virtex-E Slice**



## Implements any 5-input Function

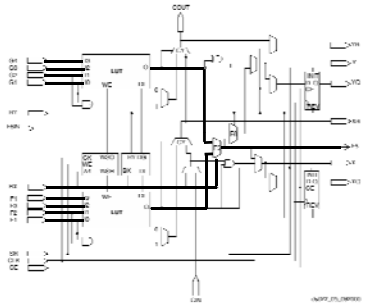


*5-input function*

*Figure 5:* **Detailed View of Virtex-E Slice**



## Implement Some Larger Functions



*e.g. 9-input parity*

*Figure 5:* **Detailed View of Virtex-E Slice**

## Two Slices: Any 6-input Function

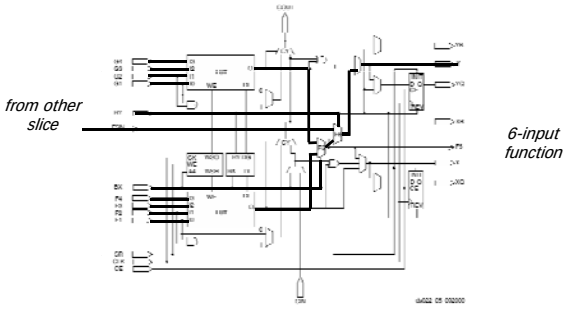



*Figure 5:* **Detailed View of Virtex-E Slice**



## 2 Slices: Implement larger functions

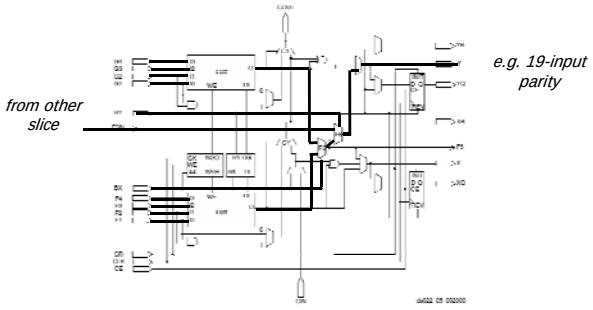



*Figure 5:* **Detailed View of Virtex-E Slice**



## Fast Carry Chain: Add 2 bits per slice

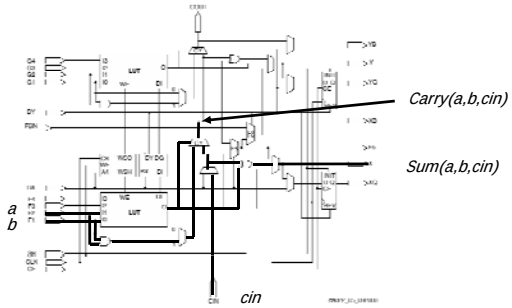



*Figure 5:* **Detailed View of Virtex-E Slice**



## LUTs as Distributed RAM/ROM

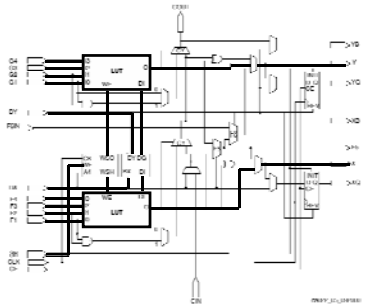


*Figure 5:* **Detailed View of Virtex-E Slice**



## LUTs used as memory

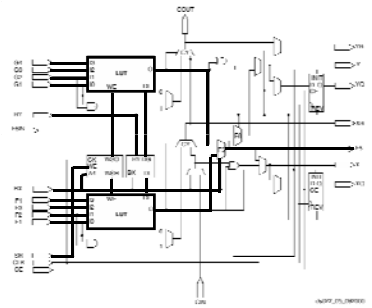


*Figure 5:* **Detailed View of Virtex-E Slice**



## Block RAM

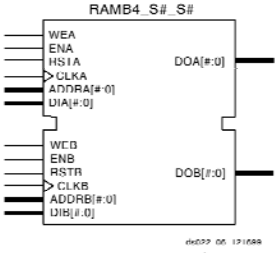



*Figure 6:* **Dual-Port Block SelectRAM**

## IO Blocks

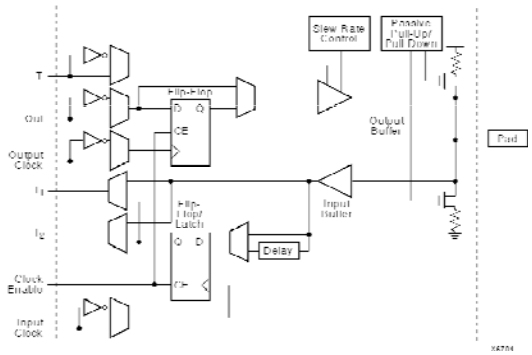



Figure 15: Simplified Block Diagram of XC4000L IOB



## Virtex Routing

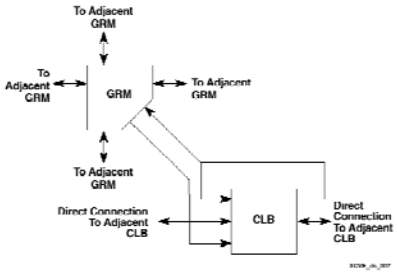



*Figure 7:* **Virtex-E Local Routing**



## Virtex Routing



## Non-Local Routing

- Hex wires
  - Extend 6 CLBs in one direction
  - Connections at 3 and 6 CLBs
    - ✓"Express busses"
    - ✓Take advantage of many metal layers
- Long wires
  - Extend the length/height of the chip
- Global signals
  - e.g. clk, reset
- Tri-state busses
  - Extend across the chip
  - Use for datapath bit-slice



## Using the DLL to De-Skew the Clock

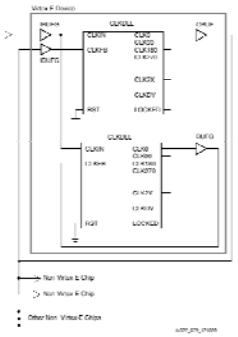


*Figure 28:* **DLL De-skew of Board Level Clock**



## Virtex IOB

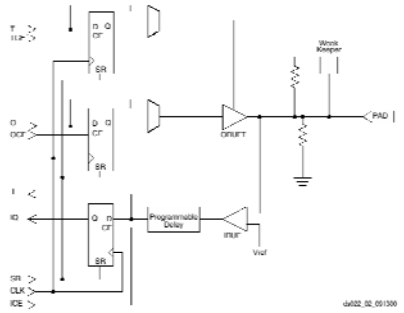



*Figure 2:* **Virtex-E Input/Output Block (IOB)**

## Computer-aided Design

- Can't design FPGAs by hand
  - way too much logic to manage, hard to make changes
- Hardware description languages
  - specify functionality of logic at a high level
- Validation - high-level simulation to catch specification errors
  - verify pin-outs and connections to other system components
  - low-level to verify mapping and check performance
- Logic synthesis
  - process of compiling HDL program into logic gates and flip-flops



## CAD Tool Path (cont'd)

- Placement and routing
  - assign logic blocks to functions
  - make wiring connections
- Timing analysis - verify paths
  - determine delays as routed
  - look at critical paths and ways to improve
- Partitioning and constraining
  - if design does not fit or is unroutable as placed split into multiple chips
  - if design it too slow prioritize critical paths, fix placement of cells, etc.
  - few tools to help with these tasks exist today
- Generate programming files - bits to be loaded

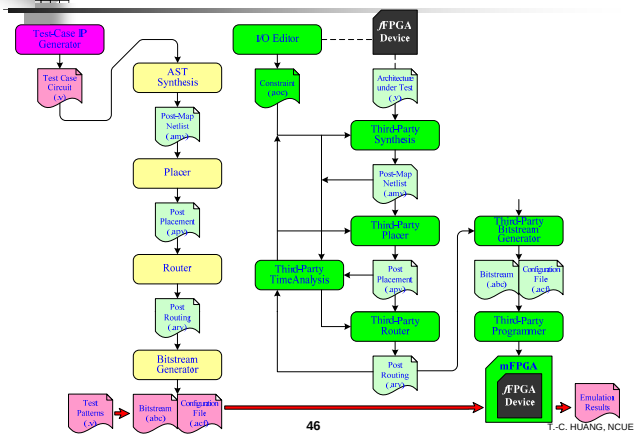


## Xilinx CAD Tools

- Verilog/VHDL used to specify logic at Hi-level
  - combine with schematics, library components
- Synplicity
  - compiles Verilog to logic
  - maps logic to the FPGA cells
  - optimizes logic
- Xilinx APR - automatic place and route (simulated annealing)
  - provides controllability through constraints
  - handles global signals
- Xilinx Xdelay
  - measure delay properties of mapping and aid
- Xilinx XACT
  - design editor to view final mapping results



## FPGA Design Flow






## Applications of FPGAs

- Implementation of random logic
  - easier changes at system-level (one device is modified)
  - can eliminate need for full-custom chips
- Prototyping
  - ensemble of gate arrays used to emulate a circuit to be manufactured
  - get more/better/faster debugging done than possible with simulation
- Reconfigurable hardware
  - one hardware block used to implement more than one function
  - functions must be mutually-exclusive in time
  - can greatly reduce cost while enhancing flexibility
  - RAM-based only option
- Special-purpose computation engines
  - hardware dedicated to solving one problem (or class of problems)
  - accelerators attached to general-purpose computers