

Outline

□ Field Programmable Gate Arrays

- ❖ Overview

- ❖ Historical perspective

□ Programming Technologies

□ Architectures

- ❖ Programmable logic

- ❖ Interconnect network

- ❖ I/O buffers

- ❖ Specialized cores

□ Programming Interfaces

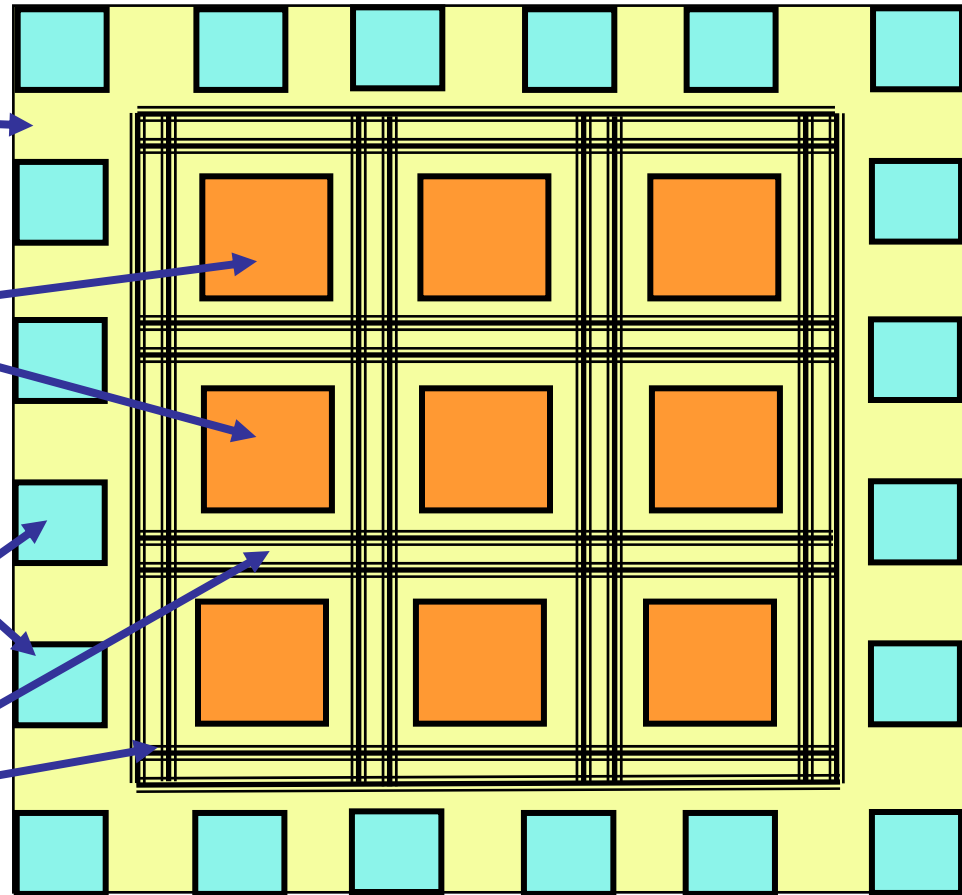
Field Programmable Gate Arrays

□ Configuration Memory

□ Programmable Logic Blocks (PLBs)

□ Programmable Input/Output Cells

□ Programmable Interconnect



Typical Complexity = 5M – 1B transistors

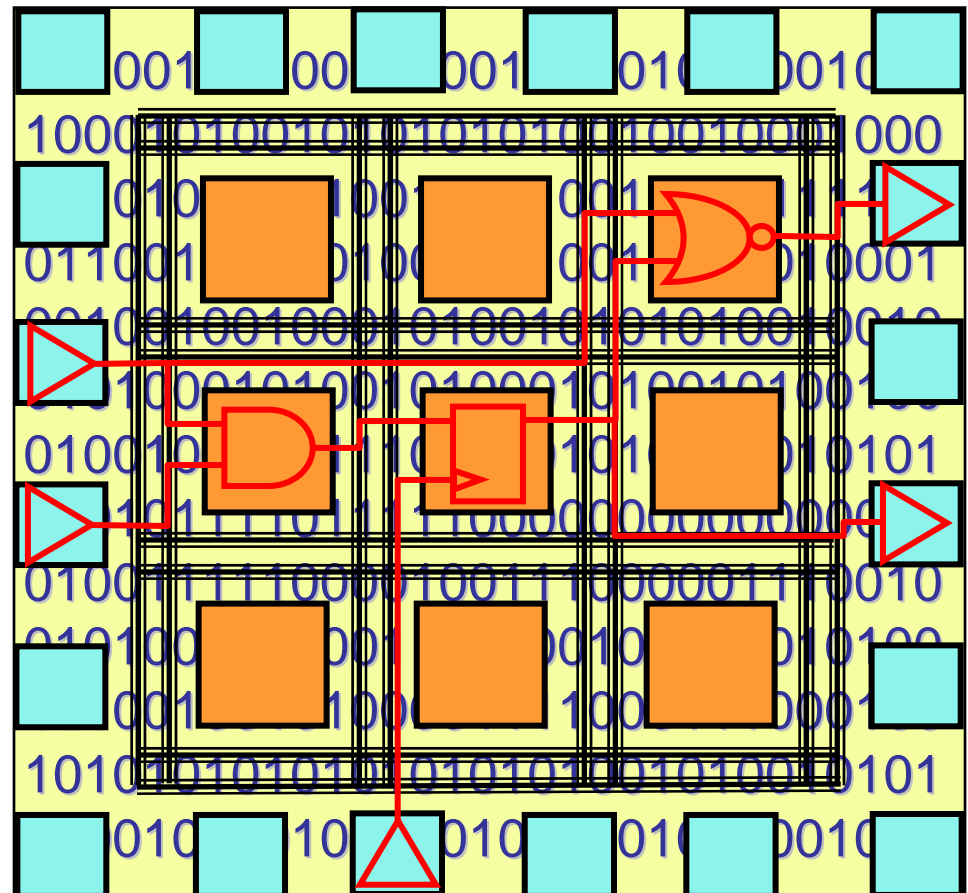
Basic FPGA Operation

Write Configuration Memory

- ❑ Defines system function
 - ❖ Input/Output Cells
 - ❖ Logic in PLBs
 - ❖ Connections between PLBs & I/O cells

Changing configuration memory data => changes system function

- ❑ Can change at anytime
 - ❖ Even while system function is in operation
 - ❖ Run-time reconfiguration (RTR)



History

- Programmable Logic Arrays ~ 1970
 - ❖ Can implement any set of sum-of-products logic equations
 - ❖ Incorporated in VLSI devices
- Programmable Logic Devices ~ 1980
 - ❖ MMI Programmable Array Logic (PAL)
 - ✓ 16L8 – combinational logic only
 - ✓ 16R8 – sequential logic only
 - ❖ AMD 22V10 and Lattice 16V8
 - ❖ Complex PLDs – arrays of PLDs with routing network
- Field Programmable Gate Arrays ~ 1985
 - ❖ Xilinx Logic Cell Array (LCA)
- CPLD & FPGA architectures became similar ~ 2000

Programming Technologies

- ❑ PLAs were mask programmable
- ❑ PALs used fuses for programming
- ❑ Early PLDs & CPLDs used floating gate technology
 - ❖ Erasable Programmable Read Only Memory (EPROM)
 - ✓ Ultra-violet erasable (UVEPROM)
 - ✓ Electrically erasable (EEPROM)
 - ✓ Flash memory came later and was used for CPLDs
- ❑ FPGAs used RAM for programming
- ❑ Later trends
 - ❖ Fuses were replaced with anti-fuses
 - ✓ Better reliability
 - ❖ Large CPLDs went to RAM-based programming

Programming Technologies

□ RAM

- ❖ Volatile – must configure after power-up
- ❖ In-System Re-programmable (ISR)
- ❖ Run-Time Reconfiguration (RTR)
 - ✓ dynamic reconfiguration while system is operating

□ Floating gate technologies

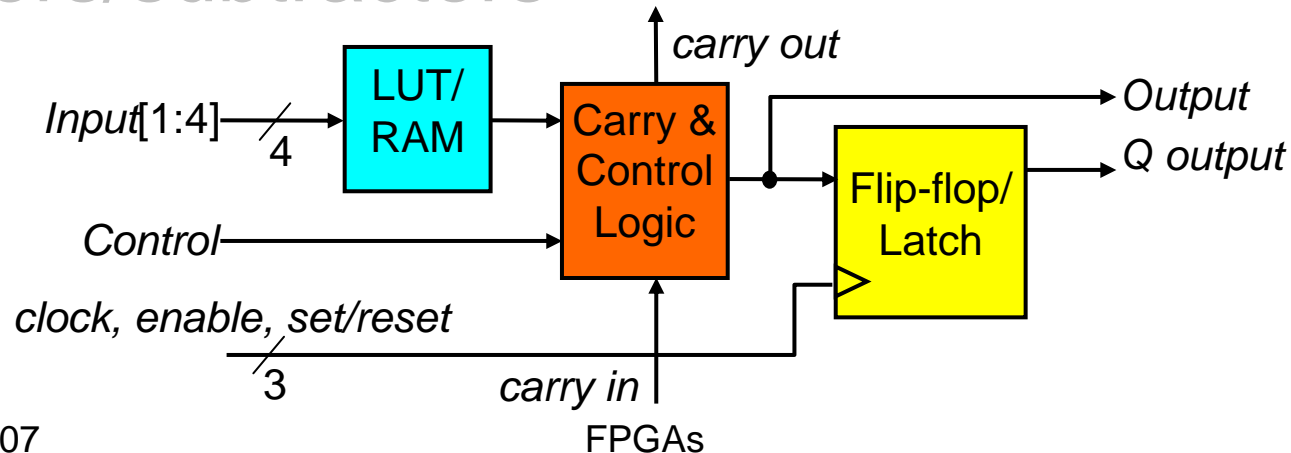
- ❖ Non-volatile but re-usable
 - ✓ UV EPROM, EEPROM, and flash memory
- ❖ In-System Programmable (ISP)
 - ✓ EEPROM and flash memory
- ❖ In-System Re-programmable (ISR)
 - ✓ Flash memory

□ Fuse/anti-fuse

- ❖ Non-volatile but not re-usable
- ❖ One Time Programmable (OTP)

Basic PLB Architecture

- ❑ Look-up Table (LUT) implements truth table
- ❑ Memory elements:
 - ❖ Flip-flop/latch
 - ❖ Some FPGAs - LUTs can also implement small RAMs
- ❑ Carry & control logic implements fast adders/subtractors



A Simple CLB

Two 3-input LUTs

- Can implement any 4-input combinational logic function

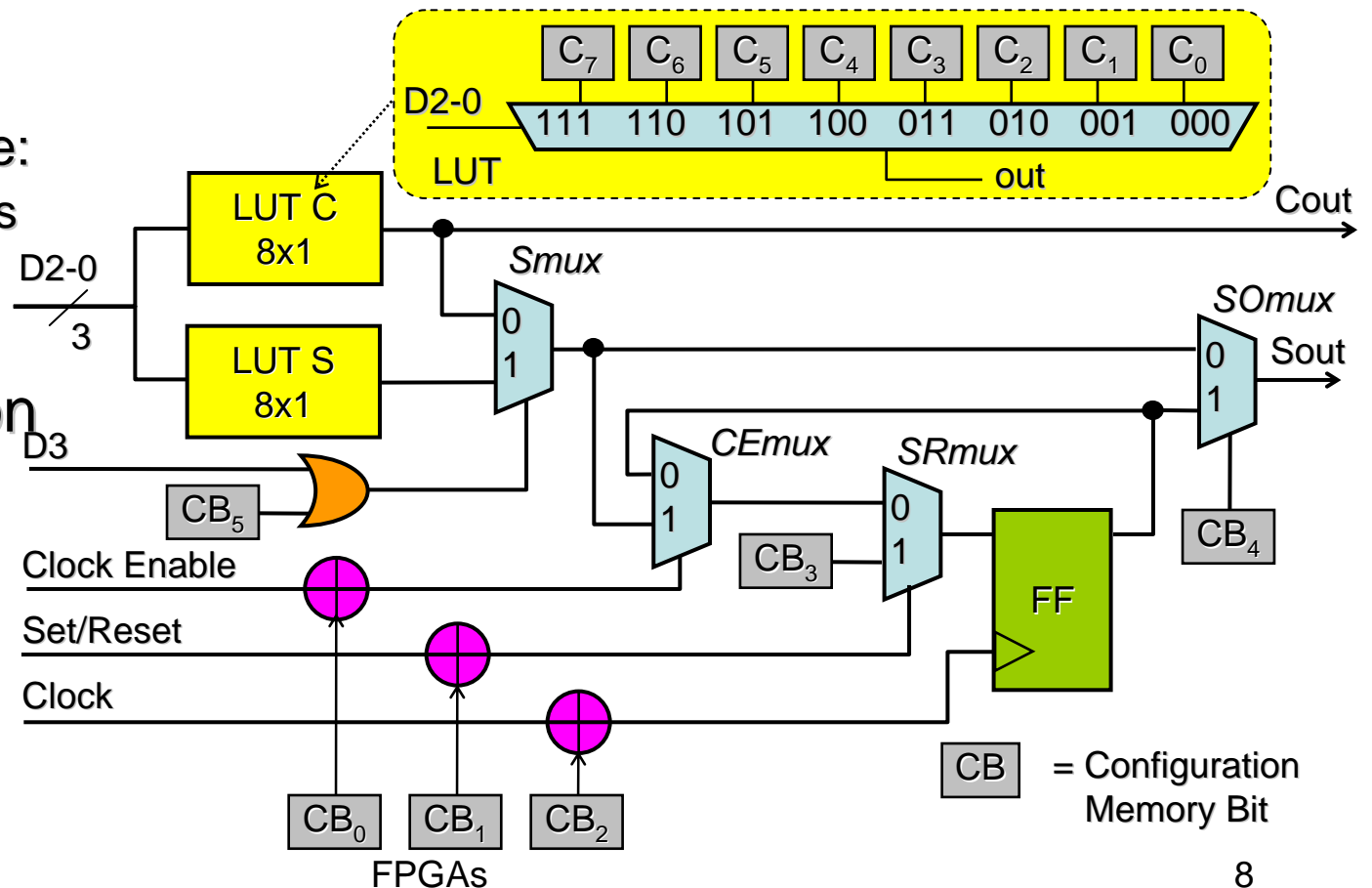
Config Bits	Configuration #1	Configuration #2	Configuration #3
LUT C (C_7-C_0)	XNOR (01101001)	XOR (10010110)	XOR (10010110)
LUT S (S_7-S_0)	XOR (10010110)	XNOR (01101001)	XNOR (01101001)
CB_0-CB_5	000010	111110	000001
Individual FC	149/174 = 85.6%	149/174 = 85.6%	108/174 = 62.1%
Cumulative FC	85.6%	97.7%	100%

1 flip-flop

- Programmable:
 - Active levels
 - Clock edge
 - Set/reset

22 configuration memory bits

- 8 per LUT
 - C_0-7
 - S_0-7
- 6 controls
 - CB_0-5



Combinational Logic Functions

□ Gates are combined to create complex circuits

□ Multiplexer example

❖ If $S = 0$, $Z = A$

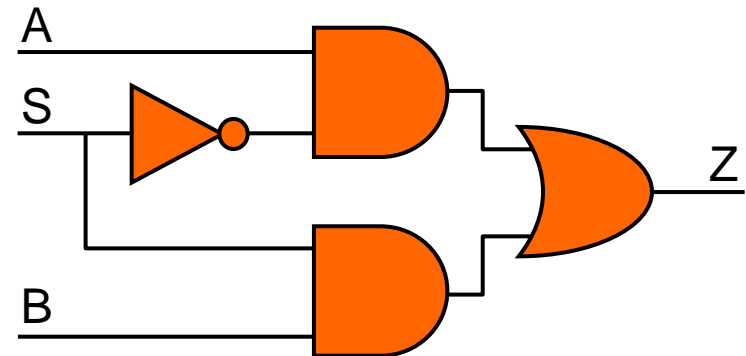
❖ If $S = 1$, $Z = B$

❖ Very common digital circuit

❖ Heavily used in FPGAs

✓ S input controlled by configuration memory bit

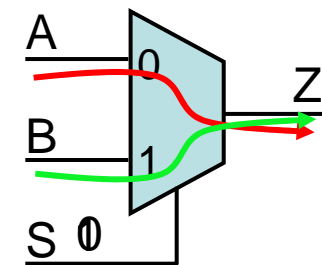
✓ We'll see it again



Truth table

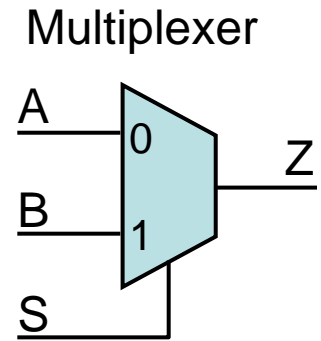
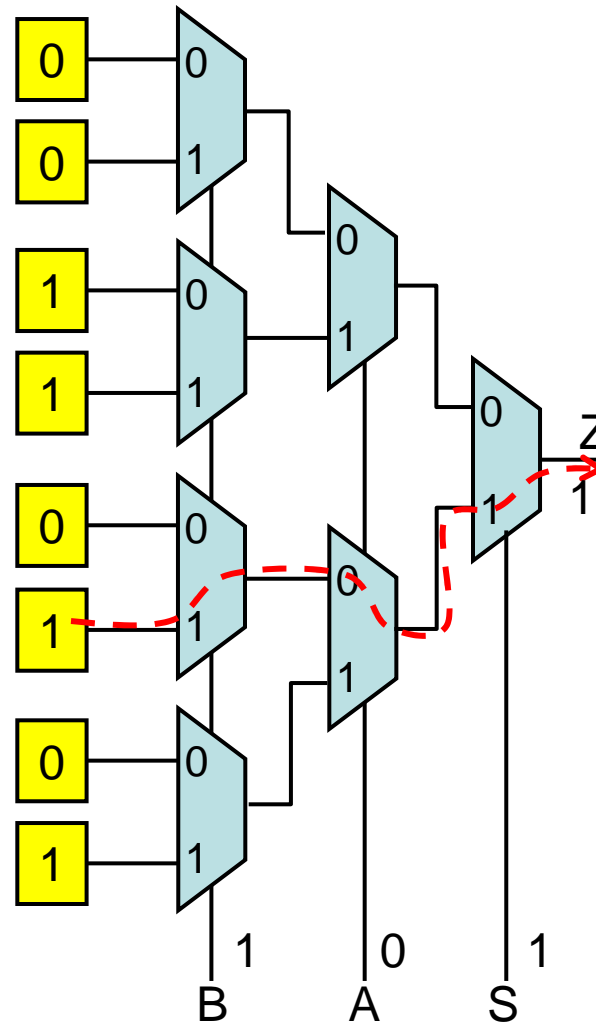
S	A	B	Z
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Logic symbol



Look-up Tables

- ❑ Recall multiplexer example
- ❑ Configuration memory holds outputs for truth table
- ❑ Internal signals connect to control signals of multiplexers to select value of truth table for any given input value



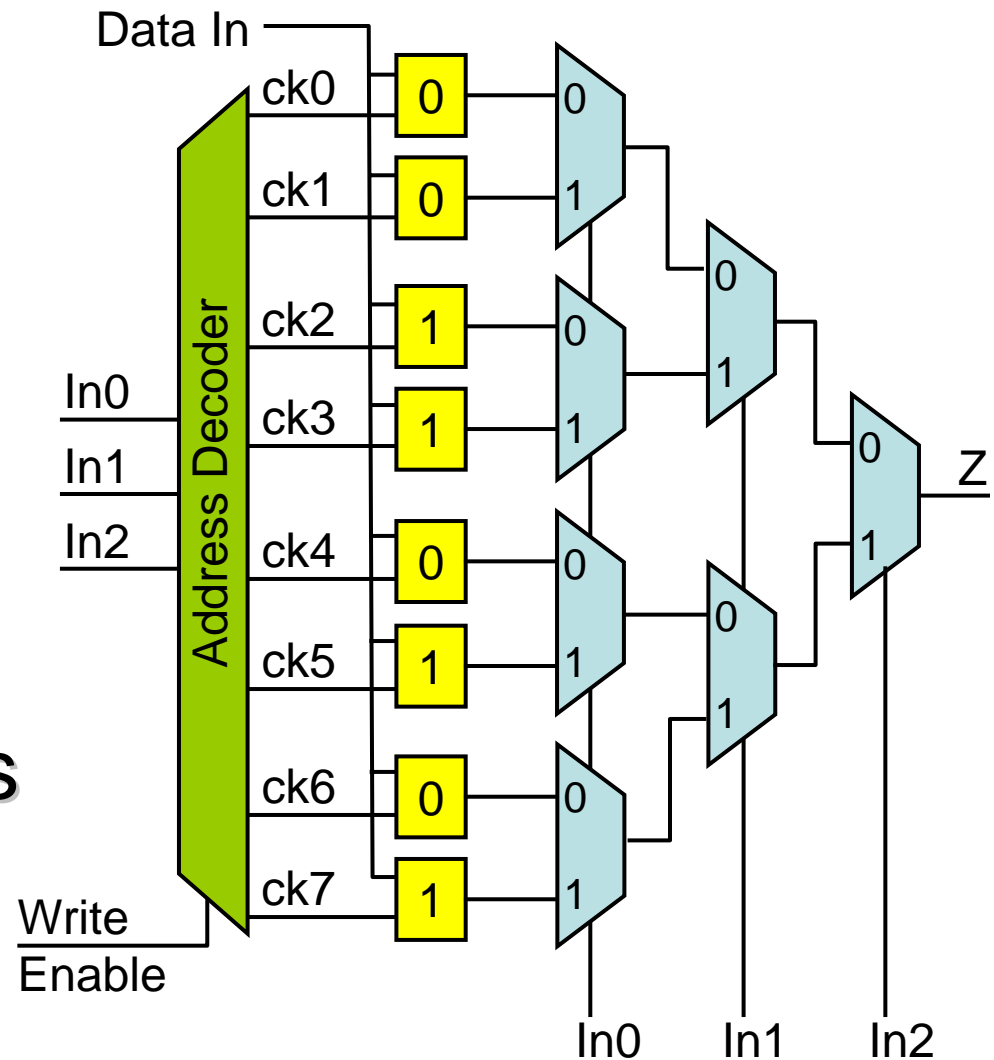
Multiplexer

Truth table

S	A	B	Z
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Look-up Table Based RAMs

- ❑ Normal LUT mode performs read operations
- ❑ Address decoder with write enable generates clock signals to latches for write operations
- ❑ Small RAMs but can be combined for larger RAMs



Input/Output Cells

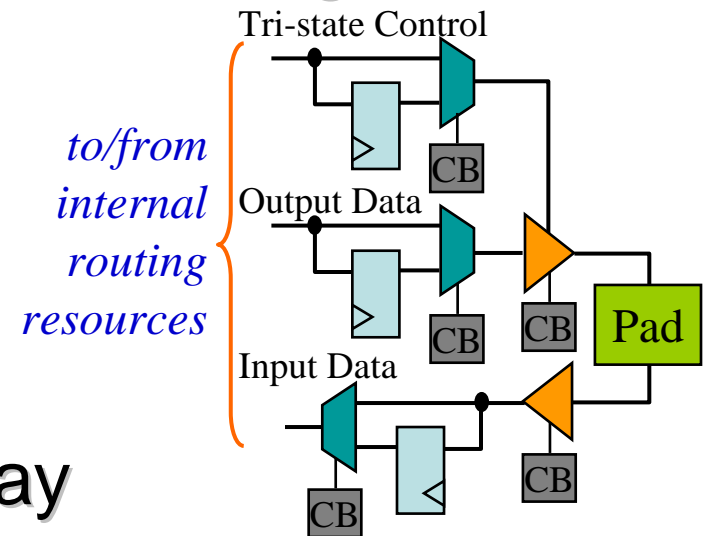
□ Bi-directional buffers

- ❖ Programmable for input or output
- ❖ Tri-state control for bi-directional operation
- ❖ Flip-flops/latches for improved timing
 - ✓ Set-up and hold times
 - ✓ Clock-to-output delay
- ❖ Pull-up/down resistors

□ Routing resources

- ❖ Connections to core of array

□ Programmable I/O voltage & current levels



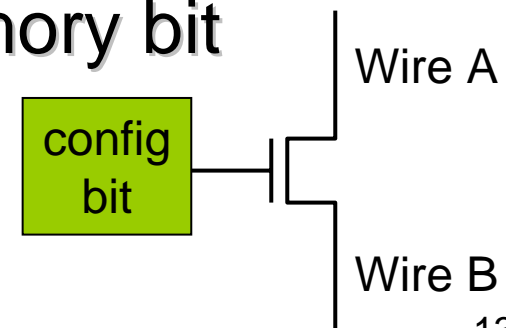
Interconnect Network

□ Wire segments of varying length

- ❖ $xN = N$ PLBs in length
 - ✓ 1, 2, 4, and 6 are most common
- ❖ $xH =$ half the array in length
- ❖ $xL =$ length of full array

□ Programmable Interconnect Points (PIPs)

- ✓ Also known as Configurable Interconnect Points (CIPs)
- ❖ Transmission gate connects to 2 wire segments
- ❖ Controlled by configuration memory bit
 - ✓ 0 = wires disconnected
 - ✓ 1 = wires connected



Programmable Interconnect Points

□ Break-point PIP

- ❖ Connect or isolate 2 wire segments

□ Cross-point PIP

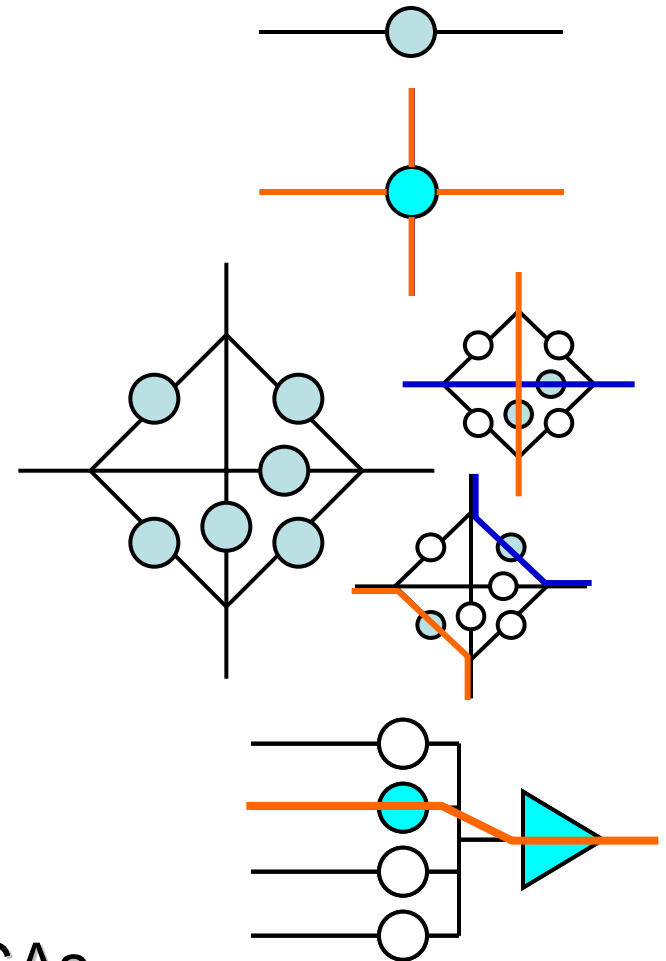
- ❖ 2 nets straight through
- ❖ 1 net turns corner and/or fans out

□ Compound cross-point PIP

- ❖ Collection of 6 break-point PIPs
 - ✓ Can route to two isolated signal nets
- ❖ Significant resource in 4000 series

□ Multiplexer PIP

- ❖ Directional and buffered
- ❖ Main routing resource in Virtex FPGAs
- ❖ Select 1-of- N inputs for output
 - ✓ Decoded MUX PIP – N configuration bits select from 2^N inputs
 - ✓ Non-decoded MUX PIP – 1 configuration bit per input



Spartan 3 Routing Resources

switch matrix
over 2,400 PIPs
mostly MUX PIPs

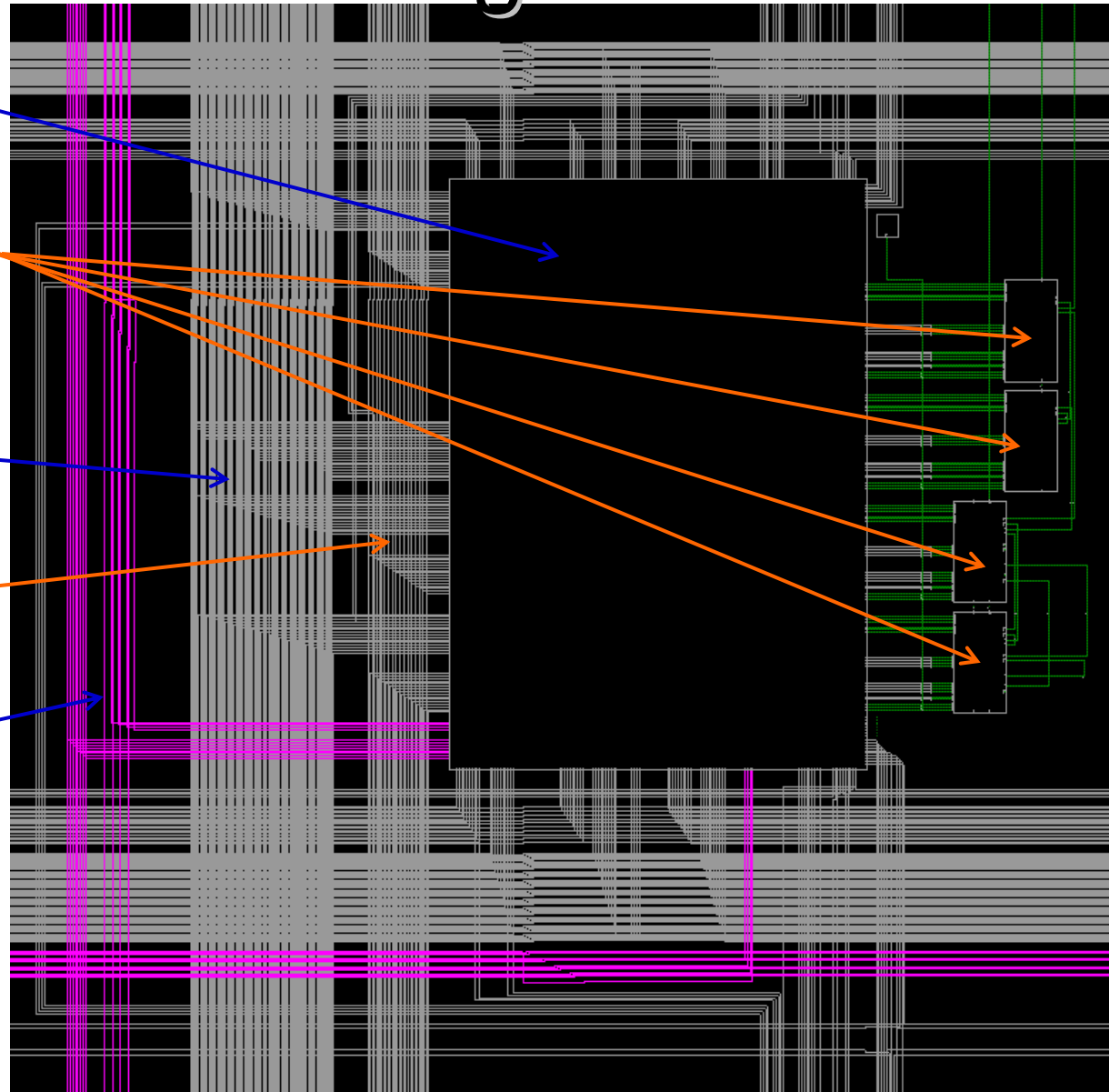
PLB consists
of 4 slices

x6 wire
segments

x2 wire
segments

xH & xL wire
segments

over 450
total wire
segments
in PLB



FPGA Architectures

❑ 4000/Spartan

- ❖ $N \times N$ array of unit cells
 - ✓ Unit cell = CLB + routing
 - Special routing along center axes
- ❖ I/O cells around perimeter

❑ Virtex/Spartan-2

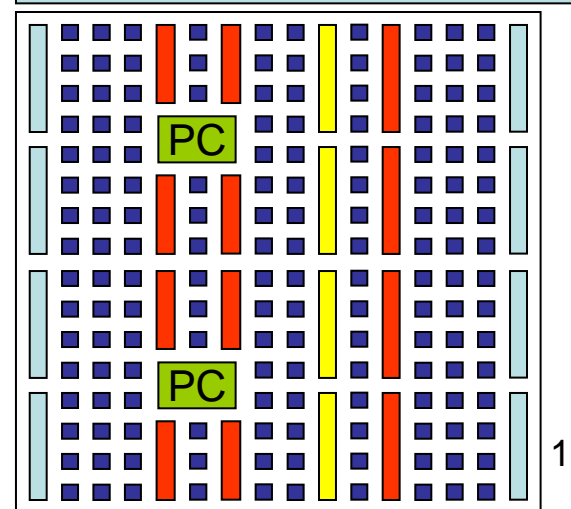
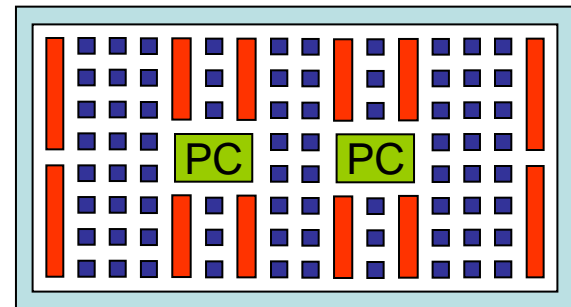
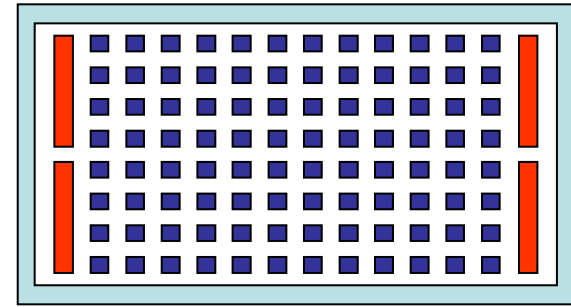
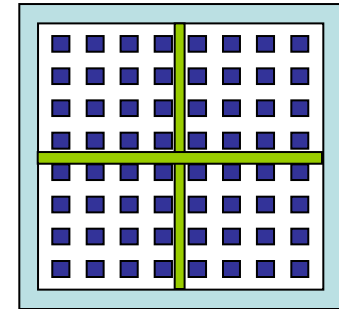
- ❖ $M \times N$ array of unit cells
- ❖ Added block 4K RAMs at edges

❑ Virtex-2/Spartan-3

- ❖ Block 18K RAMs in array
- ❖ Added 18x18 multipliers with each RAM
- ❖ Added PowerPCs in Virtex-2 Pro

❑ Virtex-4/Virtex-5

- ❖ Added 48-bit DSP cores w/multipliers
- ❖ I/O cells along columns for BGA



Ranges of Resources

FPGA Resource		Small FPGA	Large FPGA
Logic	PLBs per FPGA	256	25,920
	LUTs and flip-flops per PLB	1	8
Routing	Wire segments per PLB	45	406
	PIPs per PLB	139	3,462
Specialized Cores	Bits per memory core	128	36,864
	Memory cores per FPGA	16	576
	DSP cores	0	512
Other	Input/output cells	62	1,200
	Configuration memory bits	42,104	79,704,832

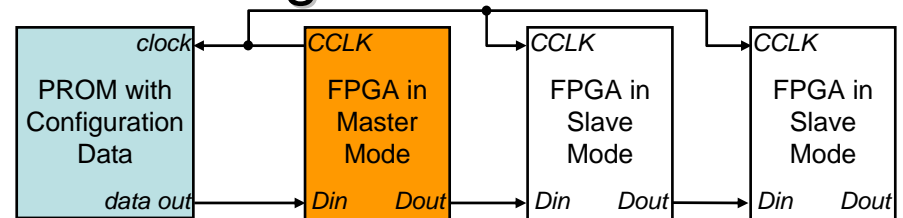
Programmable RAMs

- 18 Kbit dual-port RAM
- Each port independently configurable as
 - ❖ 512 words x 36 bits
 - ✓ 32 data bits + 4 parity bits
 - ❖ 1K words x 18 bits
 - ✓ 16 data bits + 2 parity bits
 - ❖ 2K words x 9 bits
 - ✓ 8 data bits + 1 parity bit
 - ❖ 4K words x 4 bits (no parity)
 - ❖ 8K words x 2 bits (no parity)
 - ❖ 16K words x 1 bit (no parity)
- Each port has independently programmable
 - ❖ clock edge
 - ❖ active levels for write enable, RAM enable, reset

Configuration Interfaces

- ❑ Master – FPGA retrieves its own configuration from ROM after power-up

- ❖ Serial or Parallel options



- ❑ Slave – FPGA configured by external source (i.e., a μ P)

- ❖ Serial or Parallel options
 - ❖ Used for dynamic reconfiguration
 - ❖ Can also read configuration memory contents

- ❑ Boundary Scan Interface

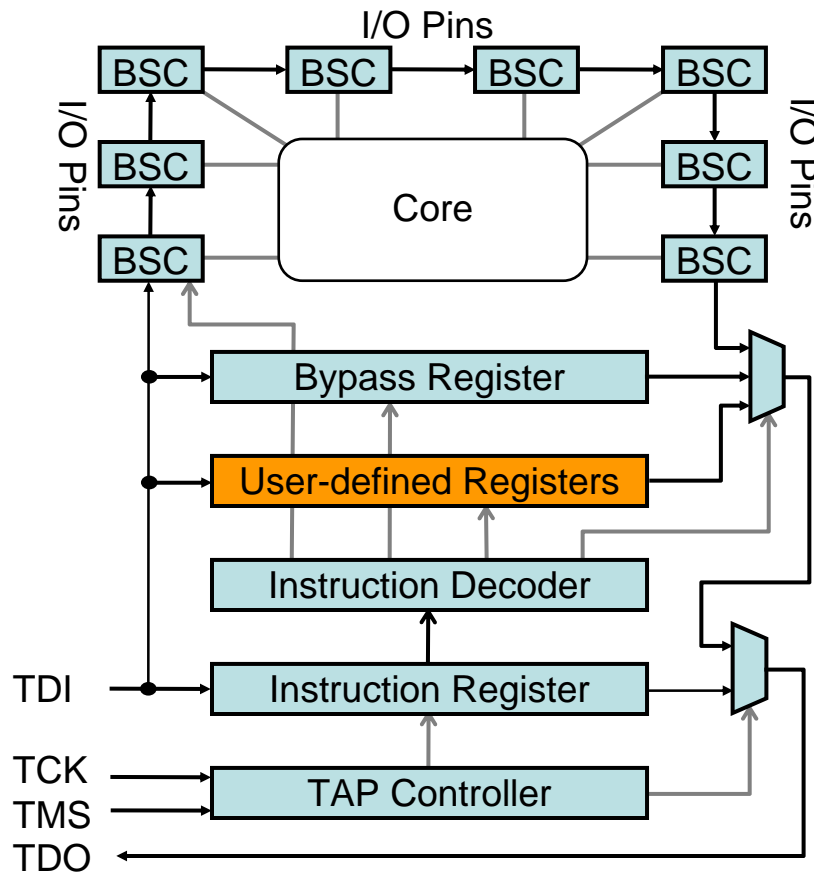
- ❖ 4-wire IEEE standard serial interface for testing
 - ❖ Write and read access to configuration memory
 - ✓ Not available in all FPGAs
 - ✓ Used for dynamic partial reconfiguration
 - ❖ Interfaces to FPGA core
 - ✓ Not available in all FPGAs
 - ✓ Connections between Boundary Scan Interface and internal routing network and PLBs (Xilinx provides 2-4 of these ports)

- ❑ Other configuration interfaces in some FPGAs

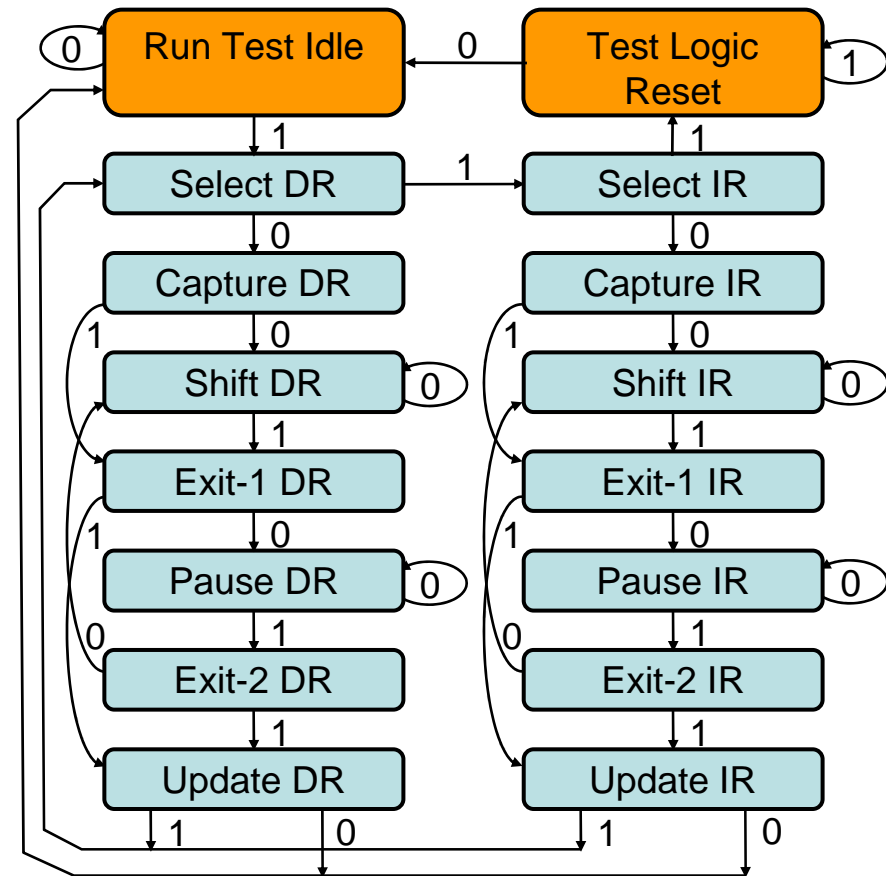
Boundary Scan Interface

User registers can include:

- a) configuration memory
- b) access to FPGA core



TAP controller state diagram



Configuration Techniques

□ Full configuration & readback

- ❖ Simple configuration interface
 - ✓ Internal automatic calculation of frame address
- ❖ Long download time for large FPGAs

□ Partial reconfiguration & readback

- ❖ Only change portions of configuration memory with respect to reference design
 - ✓ Reduces download time for reconfiguration
- ❖ Requires more complicated interface
 - ✓ Command Register (CMR)
 - ✓ Frame Length Register (FLR)
 - ✓ Frame Address Register (FAR)
 - ✓ Frame Data Register
 - Input (FDRI) – for download
 - Output (FDRO) – for readback (*note separate access*)

Configuration Techniques

□ Compressed configuration

- ❖ Requires multiple frame write capability
 - ✓ Write identical frames of config data to multiple frame addresses
- ❖ Extension of partial reconfiguration interface capabilities
 - ✓ Frame address is much smaller than frame of configuration data
- ❖ Reduces download time for initial configuration depending on
 - ✓ Regularity of system function design
 - ✓ % utilization of array
 - Unused portions written with default configuration data

Partial Reconfiguration Example

- ❑ Dummy Word 0xFFFFFFFF
- ❑ Synchronization Word 0xAA995566
- ❑ CMD Write 0x30008001
 - ❖ Reset CRC 0x00000007
- ❑ IDCODE Write 0x3001C001
 - ❖ Device ID = 0x0140D093 (3S50)
- ❑ COR Write 0x30012001
 - ❖ COR Write Packet Data 0x00003FE5
- ❑ CMD Write 0x30008001
 - ❖ Shutdown 0x0000000B
- ❑ CRC Write 0x30000001
 - ❖ CRC = 0x00002CE9
- ❑ CMD Write 0x30008001
 - ❖ AGhigh 0x00000008
- ❑ CMD Write 0x30008001
 - ❖ WCFG 0x00000001
- ❑ FAR Write 0x30002001
 - ❖ FAR = 0x00080000 (partial config)
- ❑ Part Reconfig Reg Write 0x3001E001
 - ❖ Null 0x00000000
- ❑ FDMI Write 0x300042E4
 - ❖ #words to write 0x000002E4

```
...
Bits:      26656
111111111111111111111111111111111111
10101010100110010101010101100110
00110000000000010000000000000001
00000000000000000000000000000111
00110000000000011100000000000001
00000001010000001101000010010011
00110000000000010010000000000001
0100000000000000011111111100101
00110000000000010000000000000001
000000000000000000000000000001011
00110000000000000000000000000001
000000000000000000010110011101001
... 4 NOOPs 0x20000000
00110000000000010000000000000001
000000000000000000000000000001000
00110000000000010000000000000001
00000000000000000000000000000001
00110000000000010000000000000001
00000000000000000000000000000000
... 16 NOOPs 0x20000000
001100000000000100001011100100
start of actual configuration data
```

FPGA Configuration Memories

□ PLB addressable

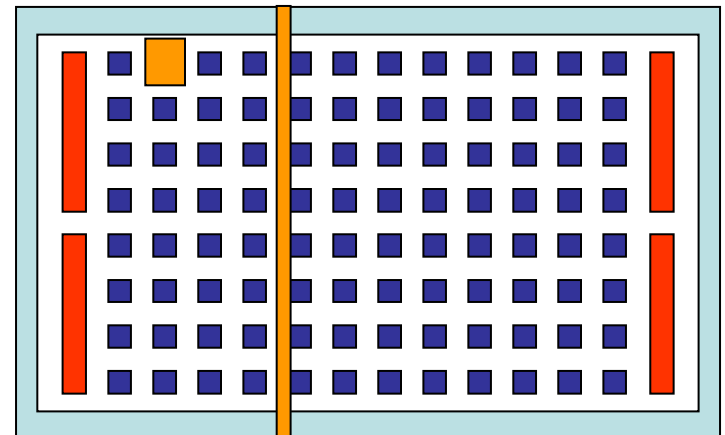
- ❖ Good for partial reconfiguration
- ❖ X-Y coordinates of PLB location to be written

- ✓ “Z” coordinate identifies which resources will be configured

Atmel uses PLB addressable memory

□ Frame addressable

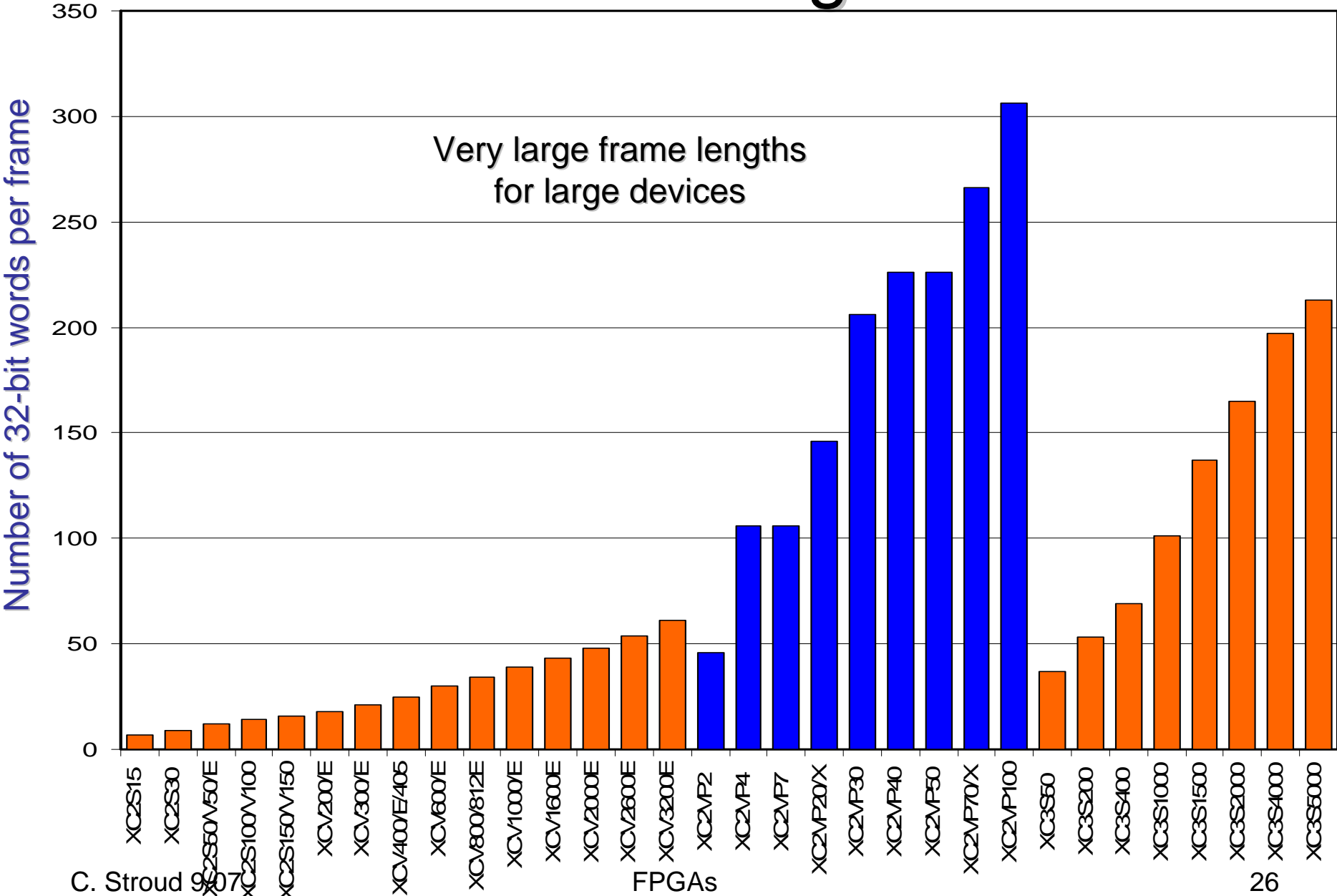
- ❖ Vertical or horizontal frame
 - ✓ Vertical frames most common
- ❖ Access to all PLBs in frame
 - ✓ Only portion of logic and routing resources accessible in a given frame
 - ✓ Many frames to configure PLBs

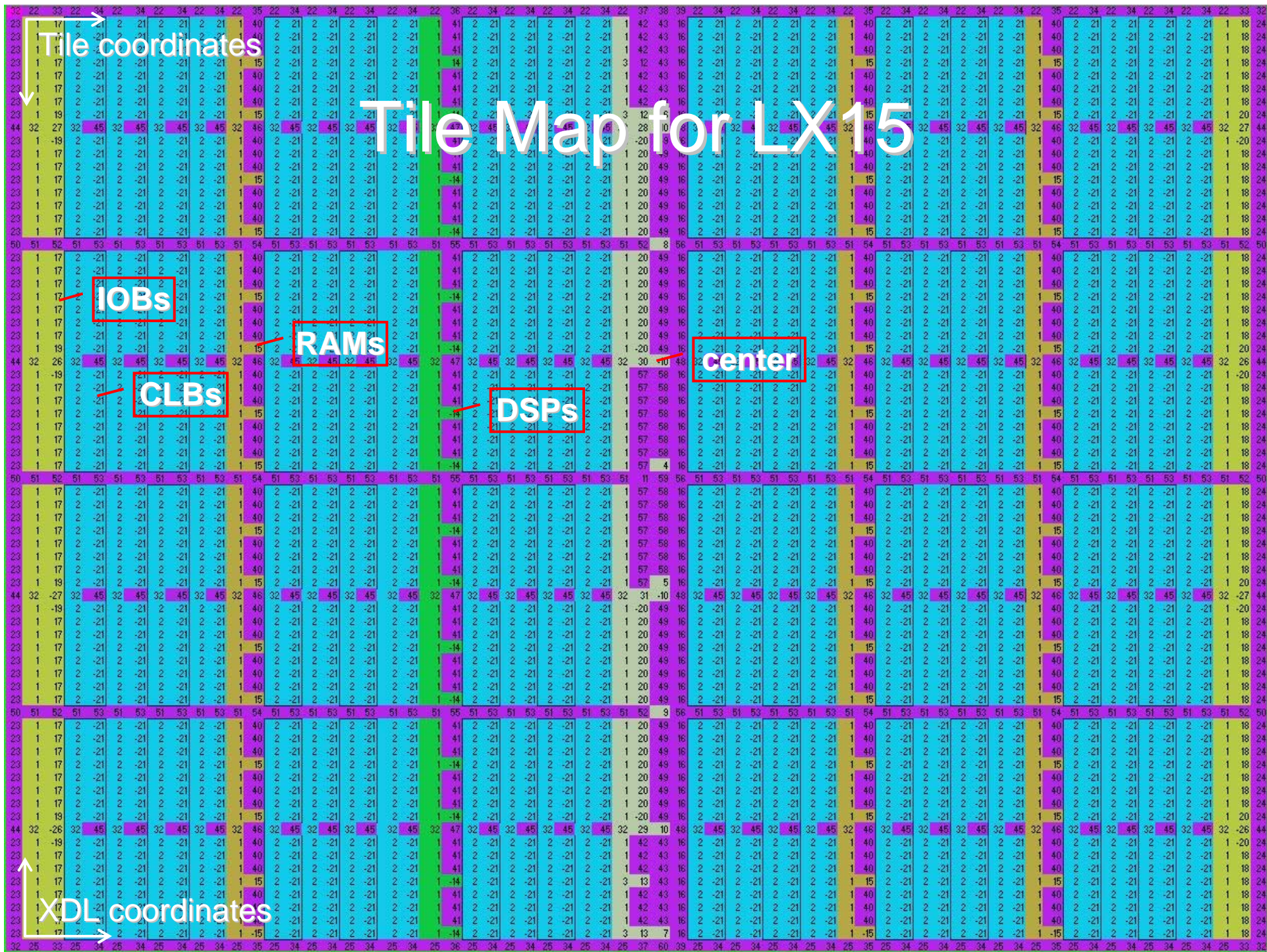


Xilinx 4000, Spartan, Virtex, Spartan-2, Virtex-2, & Spartan-3 FPGAs all use frame addressable memory

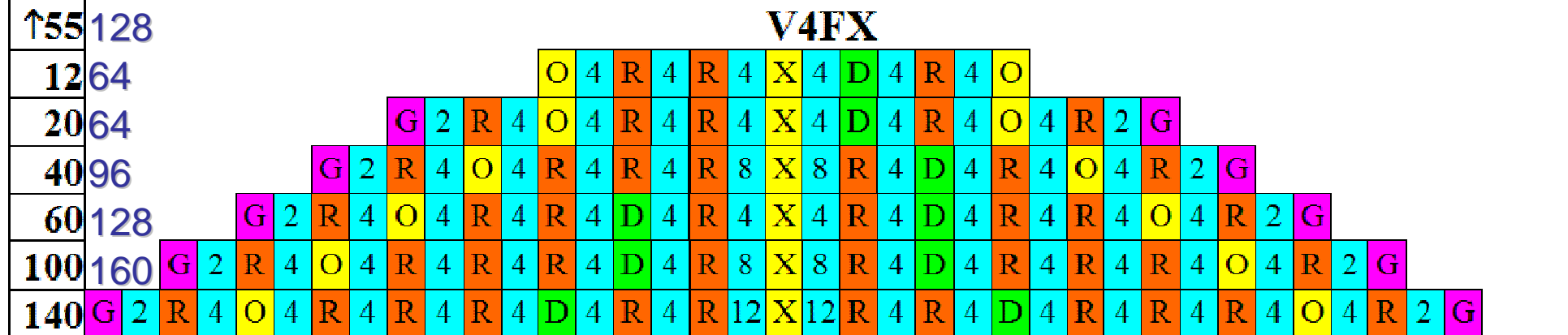
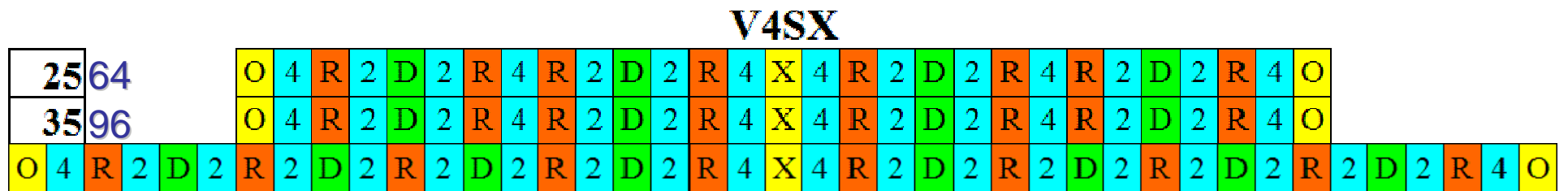
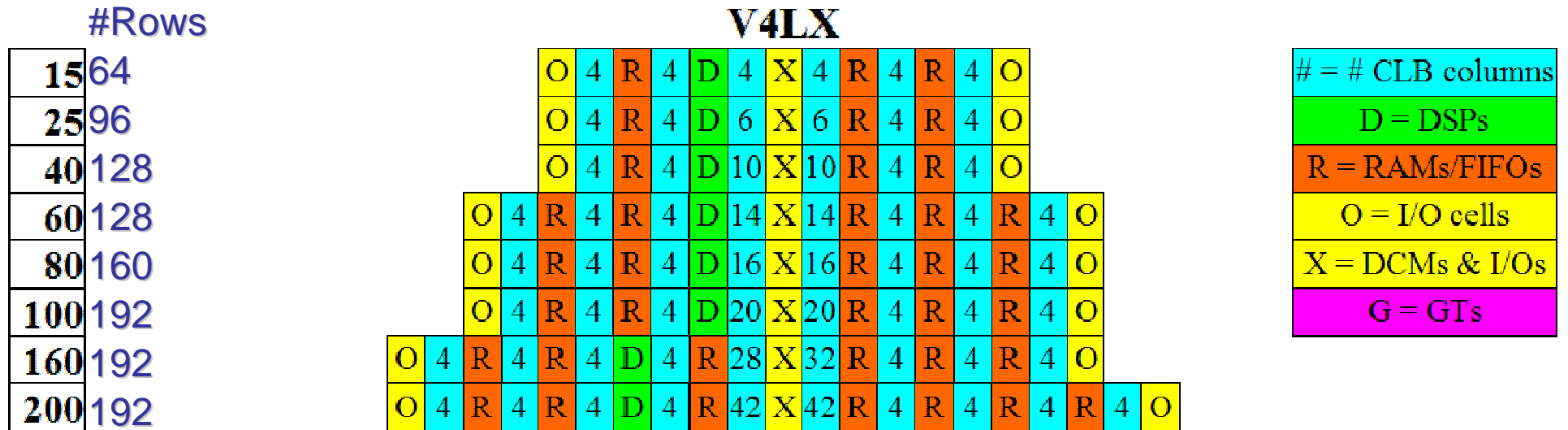
➤ Major address for column, minor address for frame

Frame Length



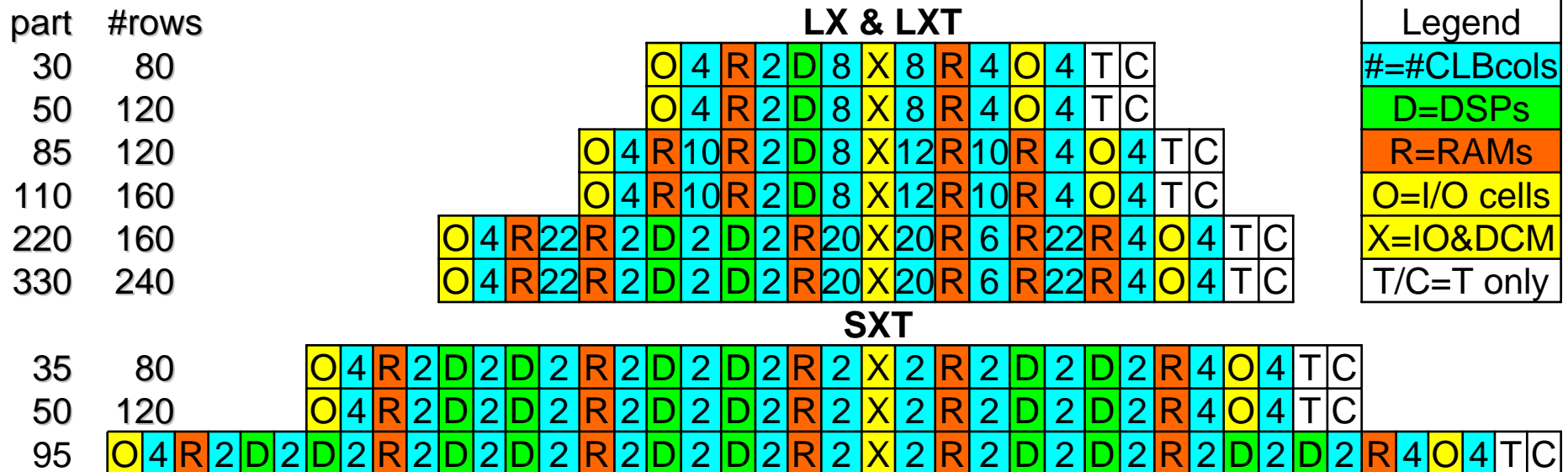


Virtex-4 Architectures



Virtex-5 Architectures

- ❑ Similar architecture, frame structure and order
 - ❖ I/O cells not along outside column on right side
 - ❖ “Center” column (Xs) not in center of array
 - ✓ More columns to right side of “center” column
 - ❖ Similar top/bottom and config row format
 - ❖ 41 words (32-bit) per frame
 - ✓ Hamming bits in middle word of frame



Legend
#=#CLBcols
D=DSPs
R=RAMs
O=I/O cells
X=IO&DCM
T/C=T only

FPGA Editor

- Viewing & editing low-level designs

- ❖ Provides detailed control of design

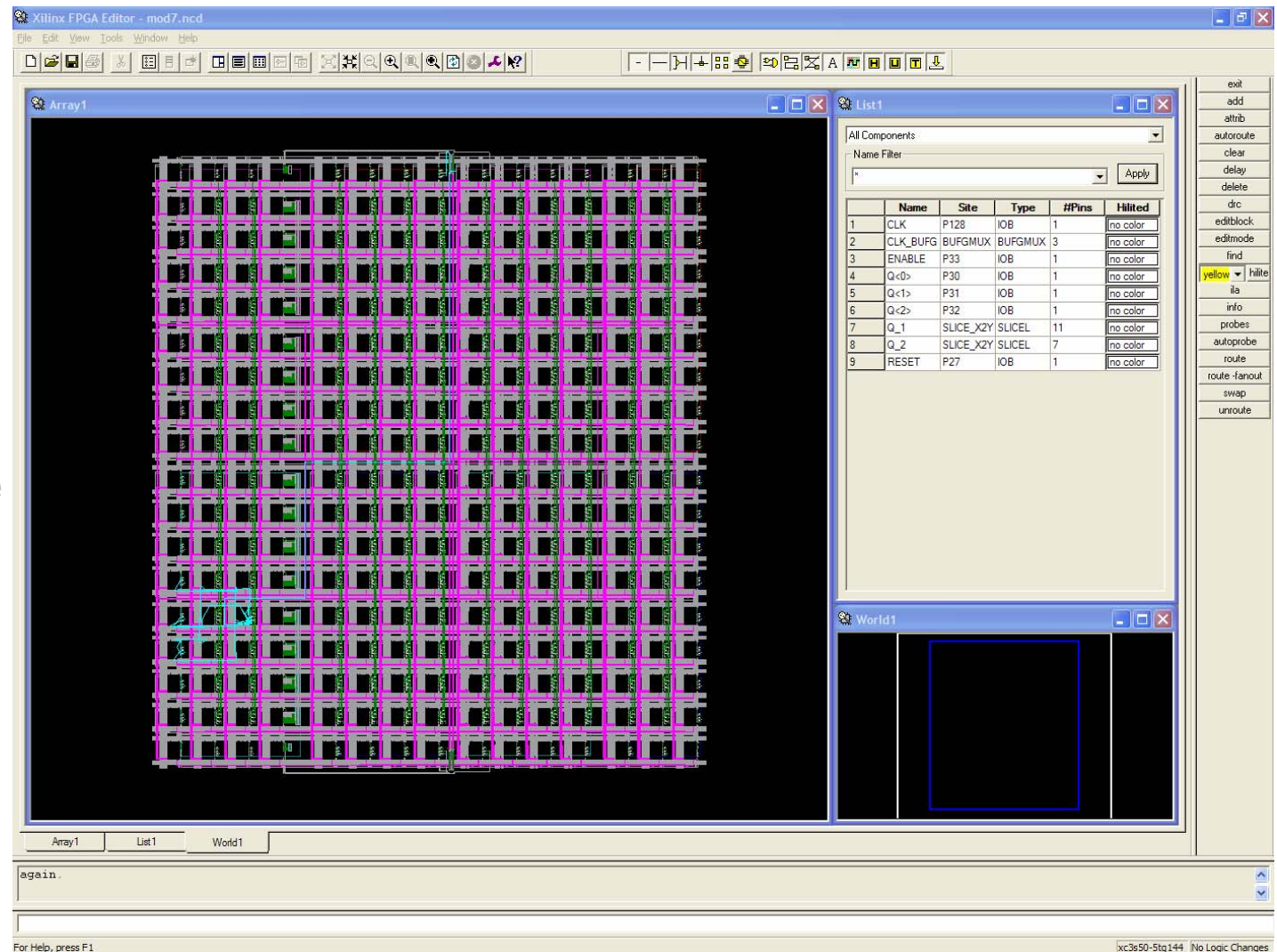
- Access via:

- ❖ ISE tool suite menu
- ❖ Command line prompt

- Excellent for architectural information

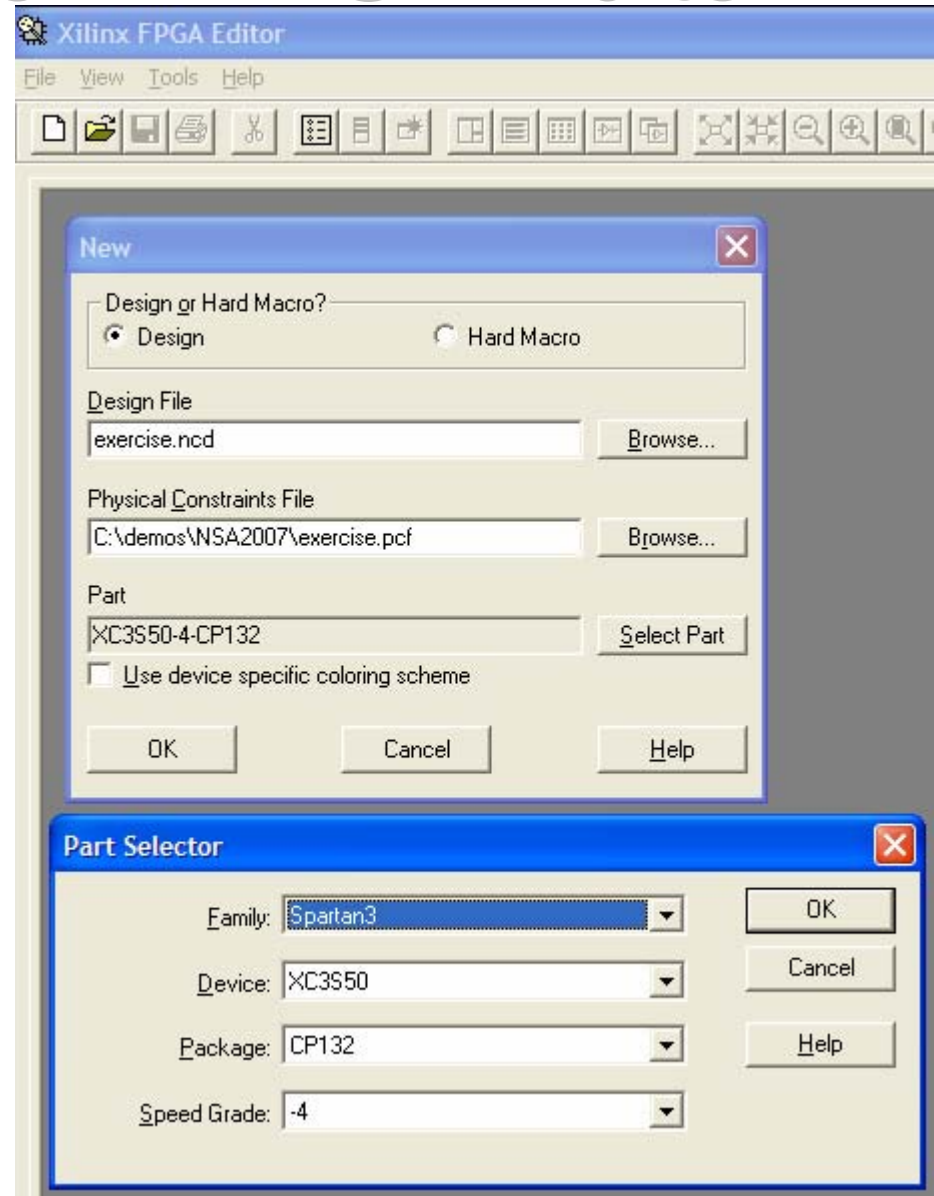
- ❖ Logic
- ❖ Routing

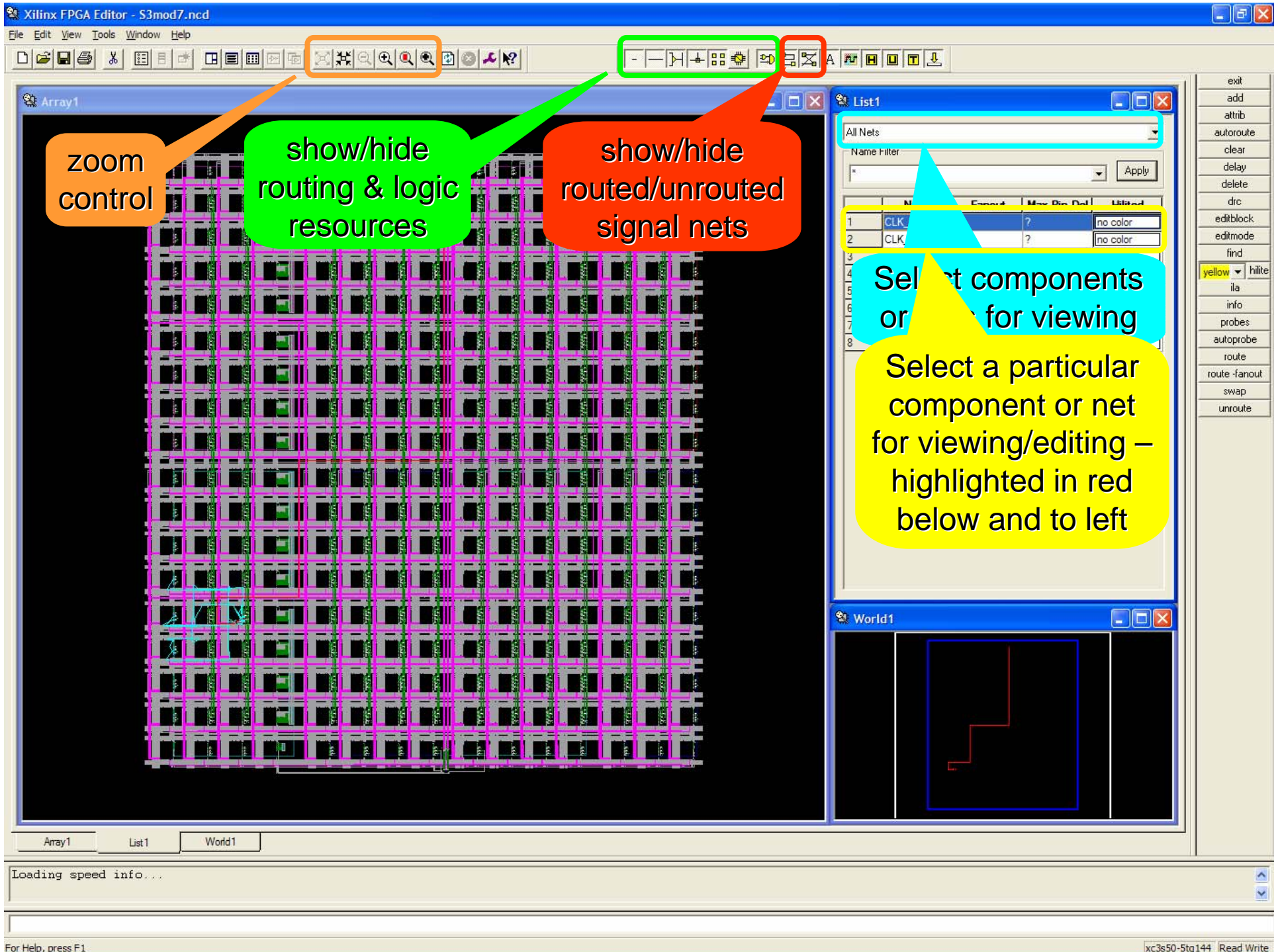
C. Stroud 9/07

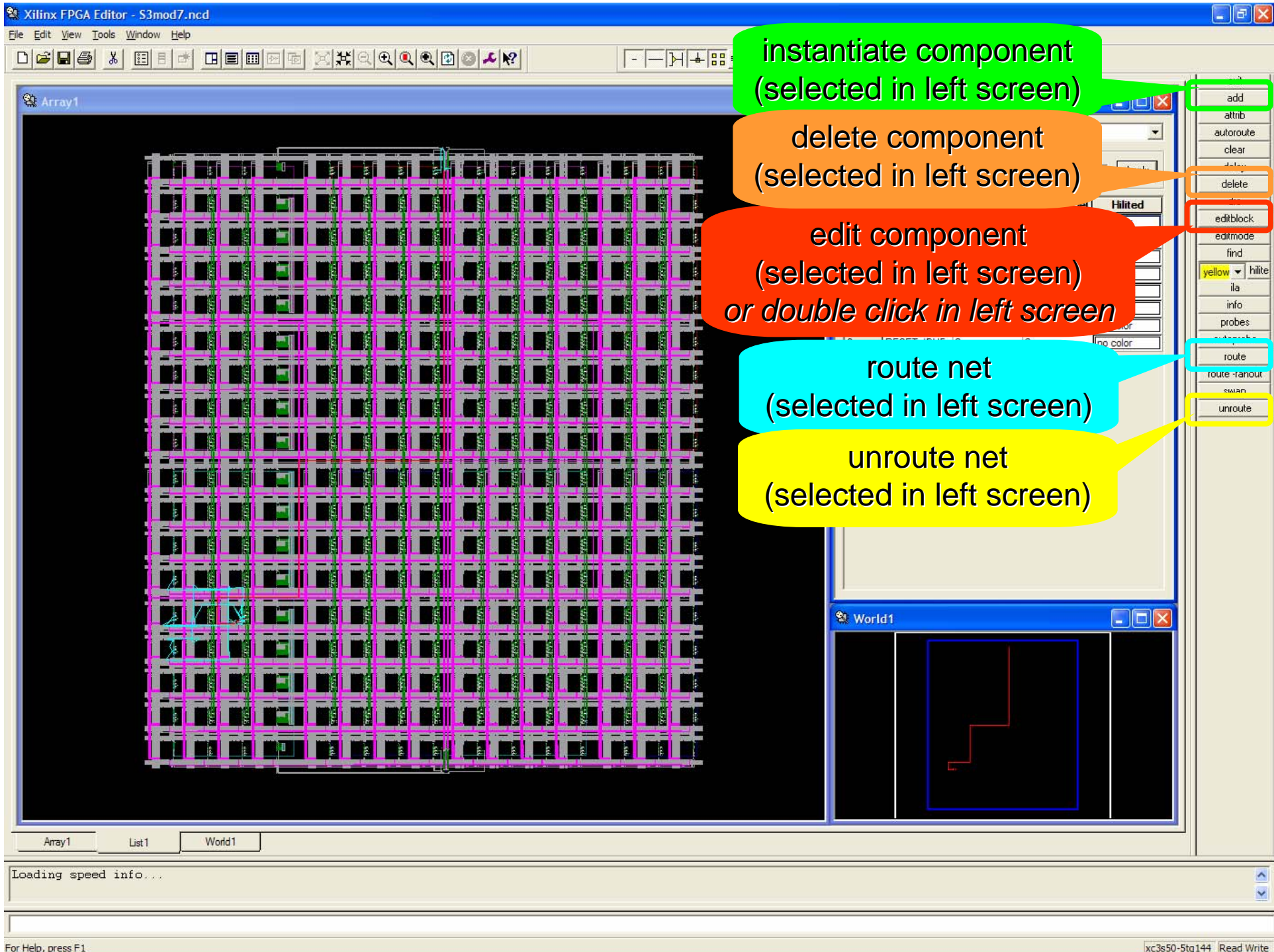


View Architecture in FPGA Editor

- ❑ Open FPGA Editor
- ❑ Select “File” → “New”
- ❑ Click “Select Part”
 - ❖ In “Part Selector” choose:
 - ✓ Spartan3 from “Family”
 - ✓ XC3S50 from “Device”
 - ❖ Click “OK”
- ❑ Type “Design File” name as ‘exercise’
- ❑ Click “OK”







instantiate component
(selected in left screen)

delete component
(selected in left screen)

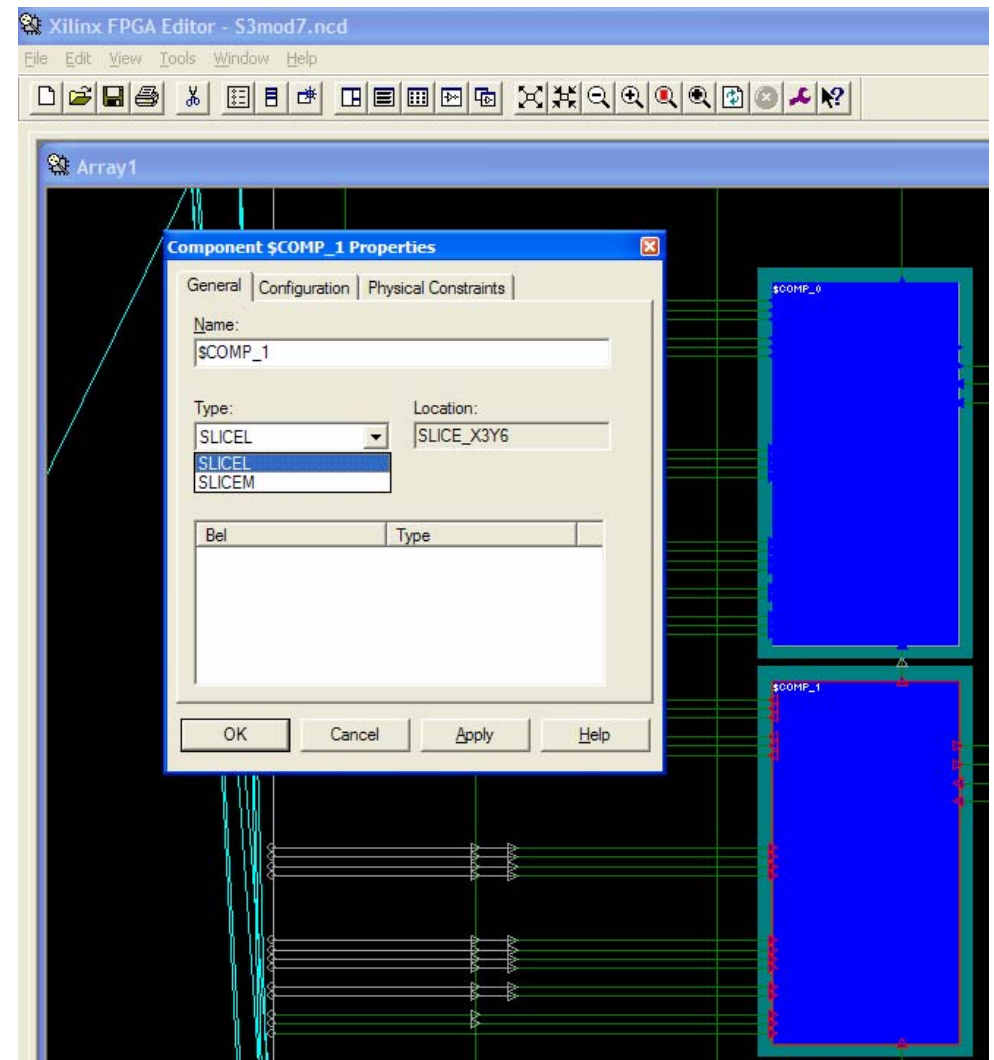
edit component
(selected in left screen)
or double click in left screen

route net
(selected in left screen)

unroute net
(selected in left screen)

Adding Component in FPGA Editor

- ❑ Select component
- ❑ Click “Add” button in right column
 - ❖ Or right click component and select “Add”
- ❑ Select “type” from menu if applicable
- ❑ Assign instantiation name
 - ❖ Or take default
- ❑ Click OK
- ❑ Now block can be edited



Xilinx FPGA Editor - slop.ncd - [Block1 - Edit Site CLB_R8C1.S1 with Comp R8C1s1]

File Edit View Tools Window Help

apply edits (runs DRC)

edit config options

property buttons and windows to view/edit configuration options

selectable configuration options

The screenshot displays the Xilinx FPGA Editor interface. The main workspace shows a logic diagram of a CLB slice (CLB_R8C1.S1) with various components like LUTs, RAMs, ROMs, and flip-flops. Callouts highlight specific features: a green callout points to the 'edit config options' button in the toolbar; an orange callout points to the 'apply edits (runs DRC)' button; a blue callout points to the 'property buttons and windows to view/edit configuration options' area; and a yellow callout points to the 'selectable configuration options' for a flip-flop component. The bottom panel shows the component name 'R8C1s1' and its configuration string 'BX:MUX:BX_B:KINV:0 DX:MUX:0 FFX:#FF'. The status bar at the bottom indicates 'comp "R8C1s1", site "CLB_R8C1.S1", type = SLICE'.

Name R8C1s1

Config BX:MUX:BX_B:KINV:0 DX:MUX:0 FFX:#FF

F

G

Array1 List1 World1 Block1

comp "R8C1s1", site "CLB_R8C1.S1", type = SLICE

exit
add
attrib
autoroute
clear
delay
delete
drc
editblock
editmode
find
yellow hilite
ila
info
probes
autoprobe
route
route -fanout
swap
unroute

xc2s15-5qcs144 Read Write

ISE Synthesis Overview

Open ISE

❖ In "File" menu select "New Project"

✓ Set project name to 'mod7'

➤ Click Next

✓ In "Device Properties" window select

➤ Family = spartan3

➤ Device = xc3s50

➤ Package = Tq144

➤ Click Next

➤ In "New Source" window click Next

➤ In "Add Existing Source"

➤ Click "Add Source"

➤ Got to directory and select "mod7.vhd"

➤ Click Next

➤ Click Finish

➤ Click OK in next window about adding source

❖ In Processes Window right click "Implement Design" and select "Run"

✓ When synthesis is complete, "Design Summary" should look like background

✓ NCD file has been created

Project Status

Project File:	mod7.ise	Current State:	Placed and Routed
Module Name:	mod7	Errors:	No Errors
Target Device:	xc3s50-5tq144	Warnings:	2 Warnings
Product Version:	ISE 8.2.01	Updated:	Sep 10 16:53:07 2007

MOD7 Partition Summary

Device Utilization Summary

Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Flip Flops	3	1,536	1%	
Number of 4 input LUTs	6	1,392	1%	
Logic Distribution				
Number of occupied slices	2	768	1%	
Number of slices containing only related logic	2	2	100%	
Number of slices containing unrelated logic	0	2	0%	
Total Number of 4 input LUTs	3	1,536	1%	
Number of bonded I/Os	6	97	6%	
Number of DSI LUTs	1	8	12%	
Total equivalent gate count for design	45			
Additional JTAG gate count for IOBs	288			

Performance Summary

Routing Results:	All Signals Completely Routed	Pinout Data:	Pinout Report
Timing Constraints:	All Constraints Met	Clock Data:	Clock Report

Detailed Reports

Report Name	Status	Generated	Errors	Warnings	Infos
Synthesis Report	Complete	Tue Sep 4 11:20:10 2007	0	0	1 Info
Translation Report	Current	Tue Sep 4 11:20:22 2007	0	0	0
Map Report	Current	Tue Sep 4 11:20:32 2007	0	2 Warnings	3 Infos
Place and Route Report	Current	Tue Sep 4 11:20:43 2007	0	0	2 Infos

Project Properties

- Enable Enhanced Design Summary
- Enable Message Filtering
- Display Incremental Messages

Enhanced Design Summary Contents

- Show Partition Data
- Show Errors
- Show Warnings
- Show Failing Constraints
- Show Clock Report

Processes

- Implement Design
 - Run
 - Rerun
 - Rerun All
 - Stop
 - Open Without Updating
 - Properties...

Started : "Launching Design Summary".

Run highlighted process