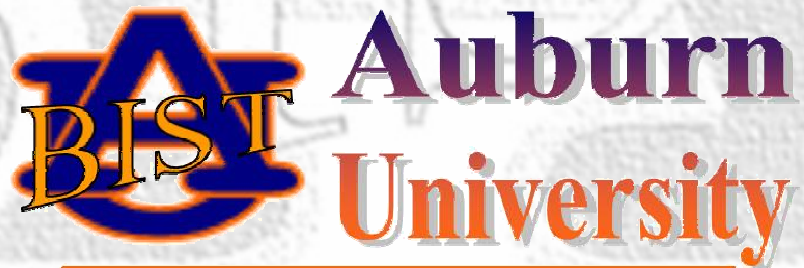


FPGA Built-In Self-Test and Diagnosis

Chuck Stroud

Electrical and Computer Engineering
Auburn University



Built-In Self-Test

BIST for FPGAs

- **Basic idea:** reprogram FPGA to test itself
 - ❖ No area overhead or performance penalties
- Applicable to all levels of testing
 - ❖ Application independent testing
 - ✓ A generic test approach for a generic component
 - ❖ Good diagnostic resolution
- **Cost:**
 - ❖ Memory to store BIST configurations
 - ✓ **Goal:** minimize number of configurations
 - ❖ Download time to execute BIST configurations
 - ✓ **Goal:** minimize downloads and/or download time

BIST for FPGAs

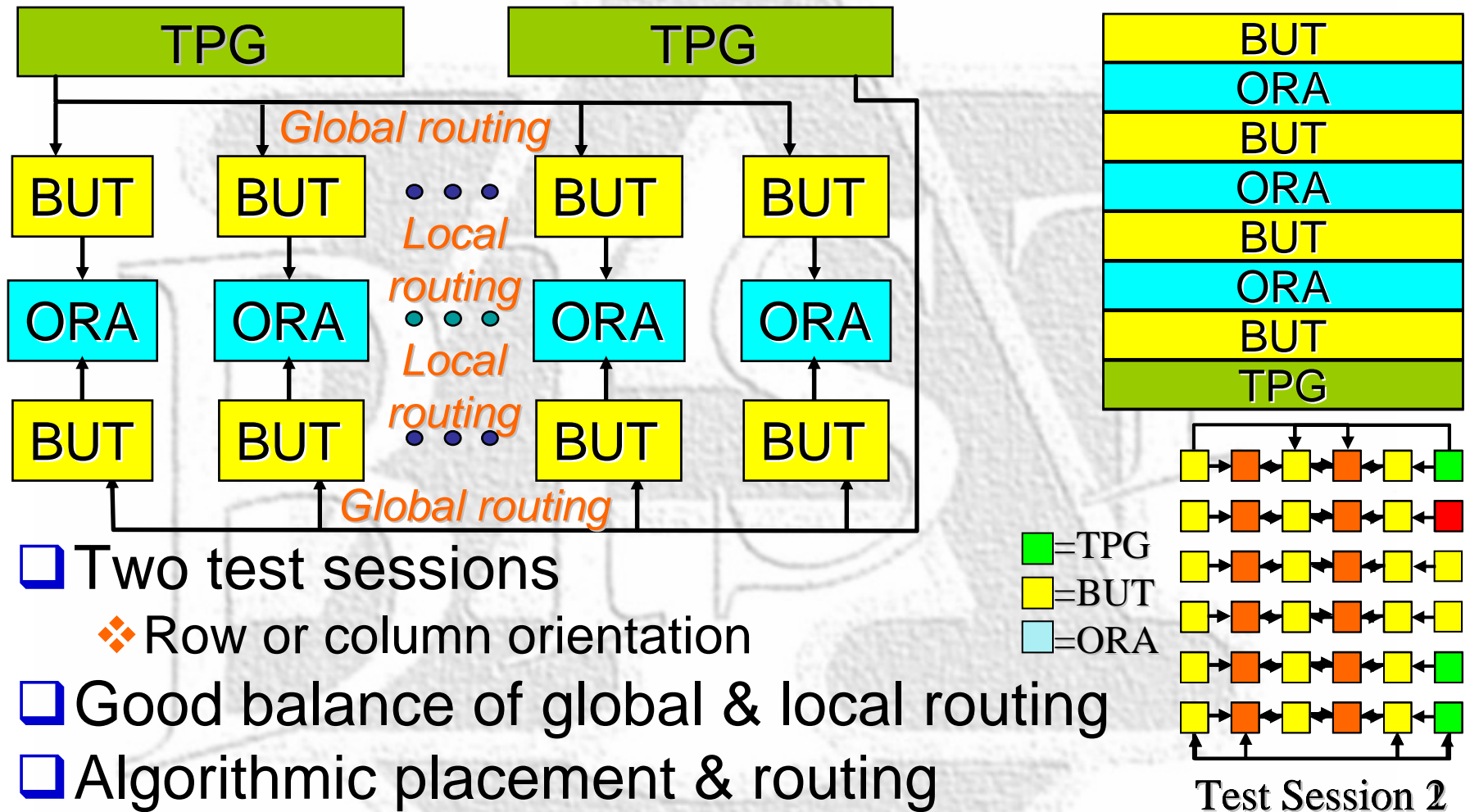
- ❑ Program some of the CLBs as Test Pattern Generators (TPGs)
- ❑ Program some of the CLBs as Output Response Analyzers (ORAs)
- ❑ Program resources to be tested
 - ❖ Logic BIST – configure logic blocks under test (BUTs)
 - ❖ Routing BIST – configure wires under test (WUTs)
 - ❖ Testing logic and routing resources typically separated
 - ✓ But you need one to test the other

FPGA BIST Configurations

FPGA		Logic	Routing	Total
ORCA	2C	9	27 (48)	66
	2CA	14	48	76
Atmel	AT94K/40K	4	56	64
Cypress	39K	20	419	459
Xilinx	4000E/Spartan	12	128	152
	4000XL/XLA	12	206	230
	Virtex-I/Spartan-II	12	283	307
	Virtex-4	15	?	?

Notes: Logic BIST configurations applied twice for total
Configurations for embedded cores not included

BIST for Programmable Logic



- ❑ Two test sessions
 - ❖ Row or column orientation
- ❑ Good balance of global & local routing
- ❑ Algorithmic placement & routing
 - ❖ Good for dynamic partial reconfiguration
 - ❖ Easily scalable with XDL

Pathological Case

To escape detection **all** of the following must be true:

- X & Y have same position in both TPGs in row 1
- V & Z have same position in both TPGs in row 8
- X & Y have equivalent faults
- V & Z have equivalent faults
- X & Y cause TPGs in row 1 to skip patterns that detect V & Z
- V & Z cause TPGs in row 8 to skip patterns that detect X & Y

								Row
	X					Y		1
								2
								3
								4
								5
								6
								7
	V					Z		8

But rotating test sessions will detect these faults!!

Diagnosis Based on BIST Results

Step 1: Record ORA results

Step 2: Mark BUTs good between consecutive ORAs with 0s

Step 3: Mark BUTs good for every two adjacent 0s followed by empty cell

Step 4: Mark BUTs bad for every consecutive 0 and 1 followed by empty cell

Step 5: Inconsistencies mean fault in ORA or in routing resources

Step 6: Unique diagnosis if all BUTs marked faulty or fault-free

row	B ₁	O ₁₂	B ₂	O ₂₃	B ₃	O ₃₄	B ₄	O ₄₅	B ₅	O ₅₆	B ₆
1	0	0	0	0	0	1	0	0	0	0	0
2	0	0	0	0	0	0	0	1	1	1	
3	0	0	0	0	0	1	1	1		1	
4	1	0	1	1	0	0	0	0	0	1	1
5		0		1	1	1	0	0	0	0	0
6	1	1	0	0	0	0	0	1	1	1	

Note:

Row 4: BUTs 1 & 2 have equivalent faults

Ambiguities:

Row 2: BUT 6 may be faulty or fault-free

Row 6: BUT 6 may be faulty or fault-free

Row 3: BUT 5 and/or BUT 6 is faulty

Row 5: BUTs 1 & 2 may be fault-free or faulty (with equivalent faults)

rotate BIST 90° to remove ambiguities

ORA Designs

- ORAs have big impact on BIST architecture

- ❖ Need to retrieve BIST results from ORAs
- ❖ Both logic and routing BIST
- ❖ Bigger impact than TPGs

- Original design

- ❖ Large comparators with output to IO pins

- Iterative ORA

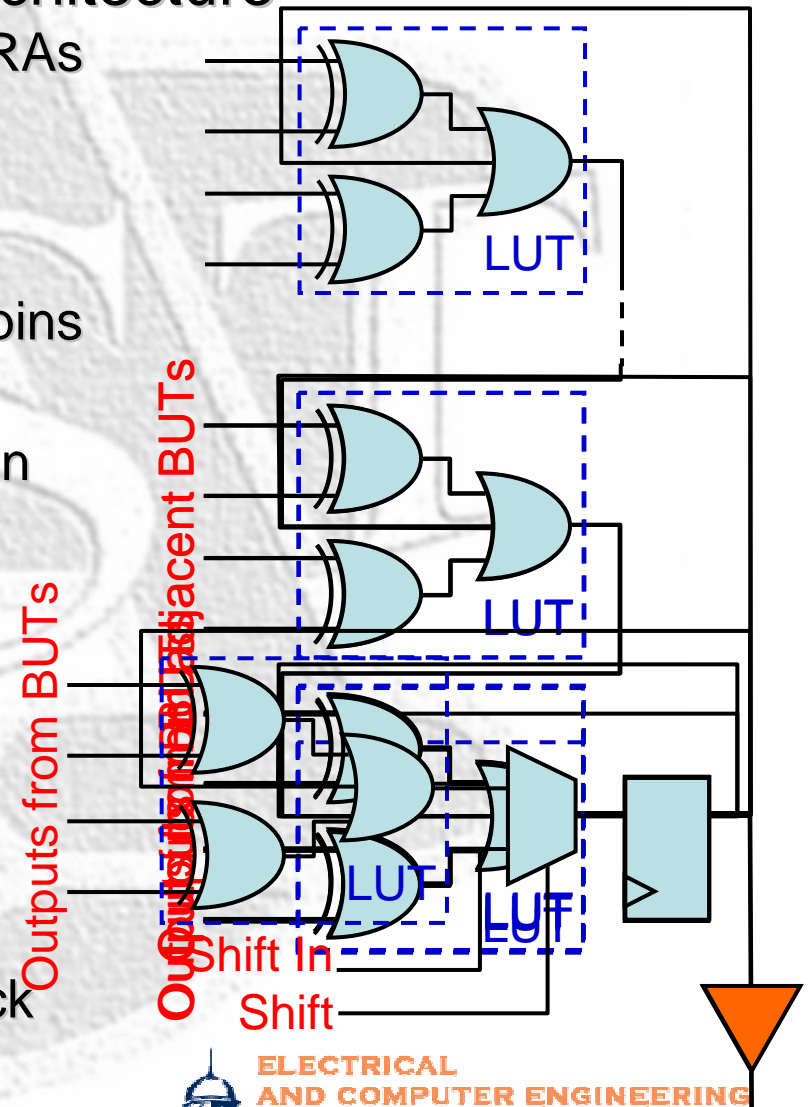
- ❖ All ORAs in row/column fed to 1 IO pin
- ❖ Poor diagnostic resolution

- Integrated ORA/scan chain

- ❖ Higher diagnostic resolution
- ❖ More logic for scan chain
- ❖ Interface to Boundary Scan

- Configuration memory readback

- ❖ No logic for scan chain
- ❖ Partial configuration memory readback
- ❖ Readback from embedded processor



Virtex-4 BIST (Circular Comparison)

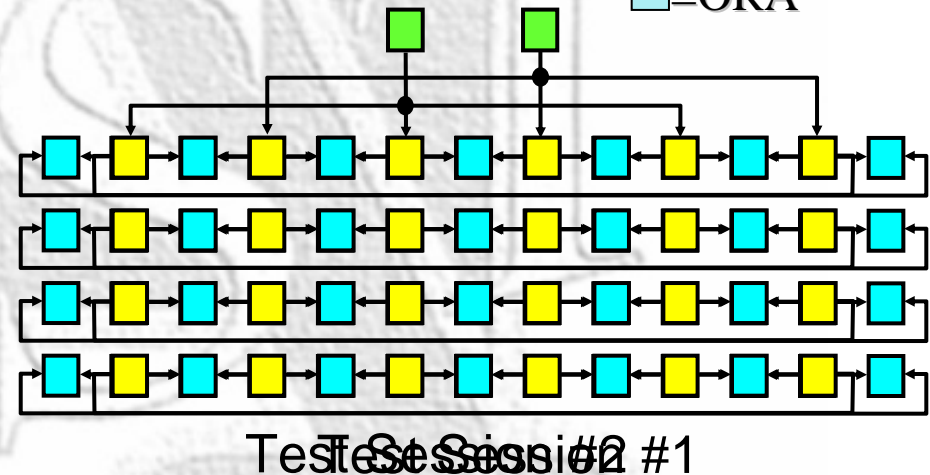
□ Circular comparison of BUTs

- ❖ Better diagnostic resolution
- ❖ Possibly better fault detection

□ Need TPGs

- ❖ Embedded processor
 - ✓ DSP
 - ✓ Embedded RAM
 - DSP counter reads
 - RAM (ROM) with test patterns

■ = TPG
■ = BUT
■ = ORA



□ Need sufficient routing resources

- ❖ For circular comparison
- ❖ Available in Virtex II & 4

Circular Comparison Diagnosis

Step 1: Record ORA results

Step 2: Mark all CUTs associated with two or more consecutive ORAs with 0s (0=fault-free)

Step 3: Recursively mark CUTs with 1 (1=faulty) for every consecutive 0 and 1 followed by empty cell

Step 4: Inconsistencies mean fault in CUT-to-ORA routing resources or in ORAs it they have not been tested and known to be fault-free

Step 5: Unique diagnosis if all CUTs marked faulty or fault-free

Notes:

- ❖ No loss of diagnostic resolution at edge of array - *there are no edges*
- ❖ C3 and C4 have equivalent faults

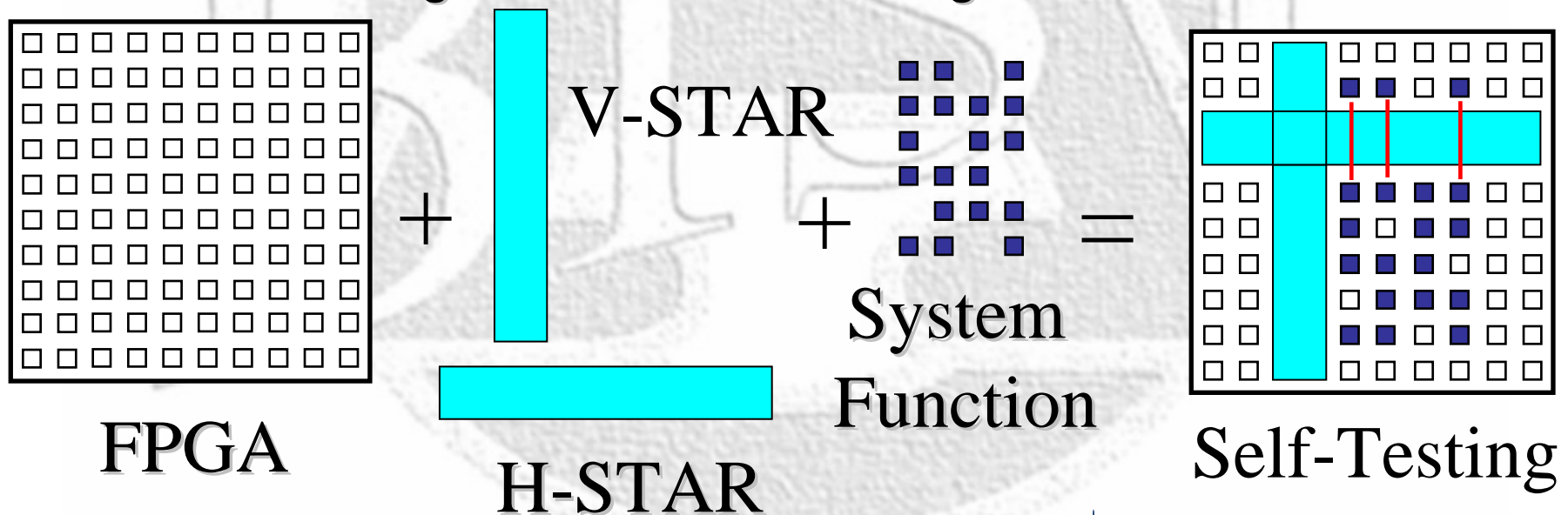
O ₉₁	C ₁	O ₁₂	C ₂	O ₂₃	C ₃	O ₃₄	C ₄	O ₄₅	C ₅	O ₅₆	C ₆	O ₆₇	C ₇	O ₇₈	C ₈	O ₈₉	C ₉	O ₉₁
0	0	0	0	1	1	0	1	1	0	0	0	0	0	1	1	1	0	0

CUT=Circuit Under Test (CLBs, DSPs, RAMs, etc.)

On-line BIST, Diagnosis & FT

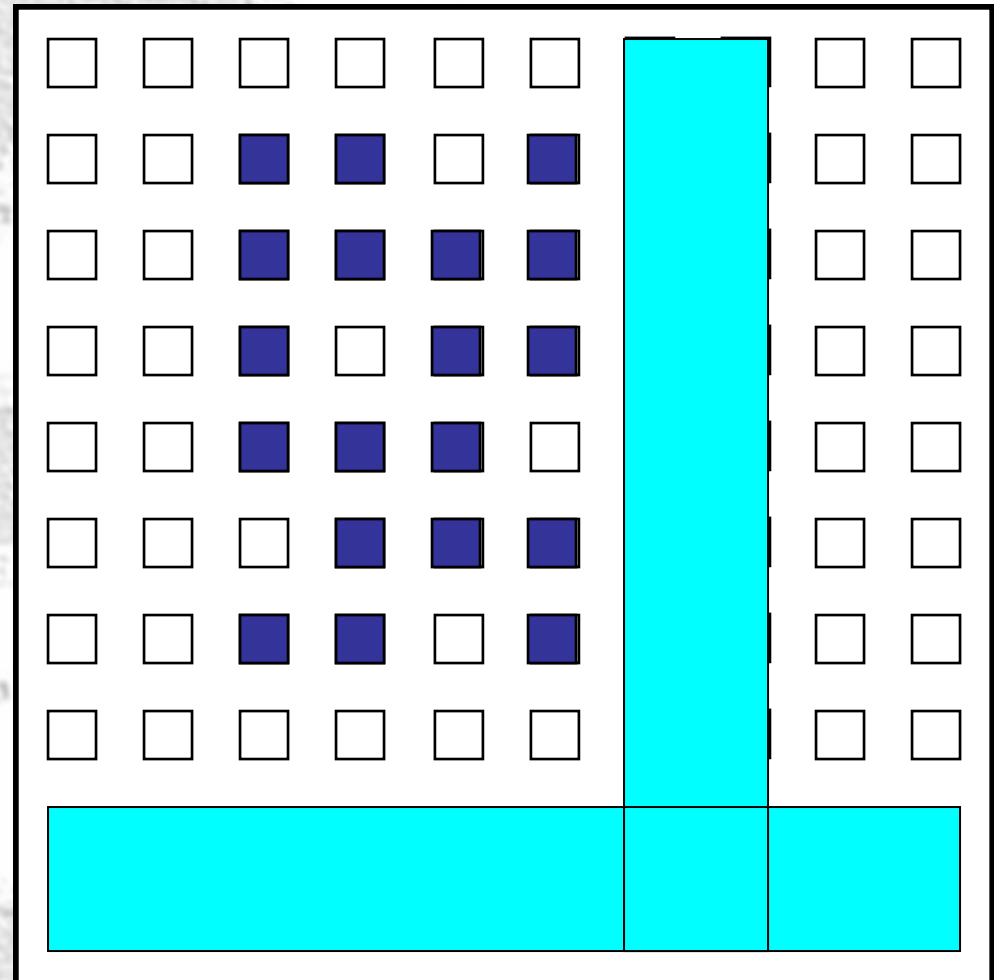
□ Roving Self Testing AREAs (STARs)

- ❖ Test programmable logic & interconnect in FPGA
- ❖ Horizontal STAR (roves up and down FPGA)
 - ✓ Tests horizontal routing resources
- ❖ Vertical STAR (roves across FPGA)
 - ✓ Tests logic and vertical routing resources



On-Line BIST, Diagnosis, & FT

- ❑ Exploits dynamic partial reconfiguration
- ❑ STARs rove across FPGA performing BIST
- ❑ Diagnosis when faults are detected
- ❑ Reconfiguration of system function to avoid faults when STAR moves to new position



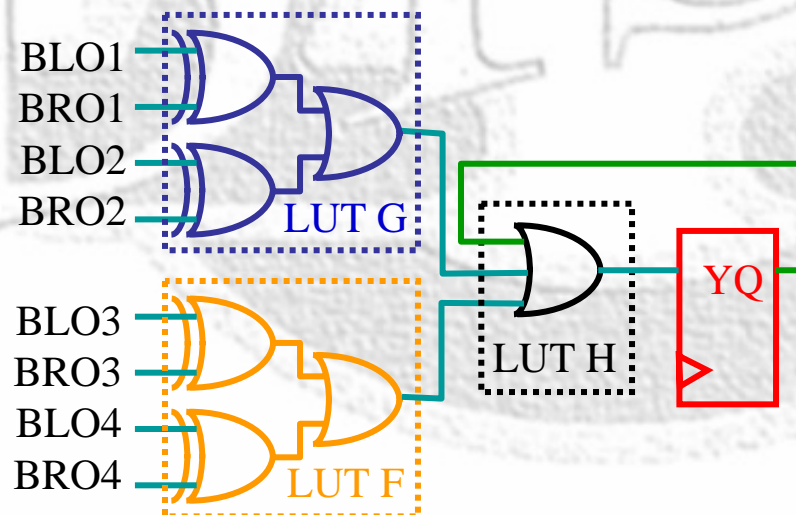
XDL Example of an ORA

```
inst "ora_1_1" "CLB" , placed R1C1 CLB_R1C1 ,
cfg "F::#LUT:F=(F1@F2)+(F3@F4) F4MUX::F4I
G::#LUT:G=(G1@G2)+(G3@G4) G3MUX::G3I G2MUX::G2I
H::#LUT:H=F+G+H1 H0::G H1::C4 H2::F
CLKY::CLK DY::H YQMUX::QY SRY::RESET FFY::#FF
SRX::RESET FFX::#FF " ;
```

```
net "ora_1_1_yq" ,
  outpin "ora_1_1" YQ ,
  inpin "ora_1_1" C4 ,
; pip R1C1 CENTER_GYQ -> CENTER_GYQ_VERT ,
  pip R1C1 CENTER_GYQ_VERT -> CENTER_H2R ,
  pip R1C1 CENTER_H2R -> CENTER_C4 ,
;
```

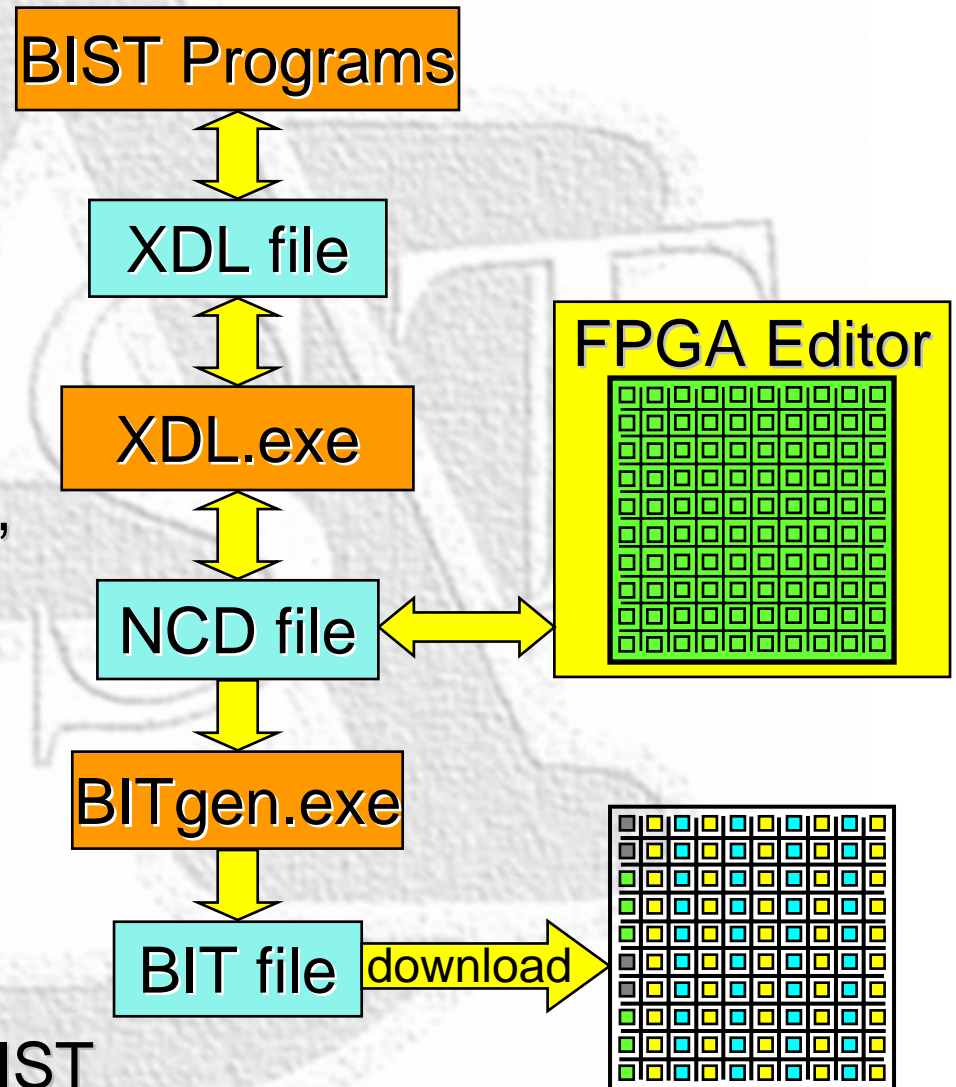
unrouted net in XDL

routed net in XDL



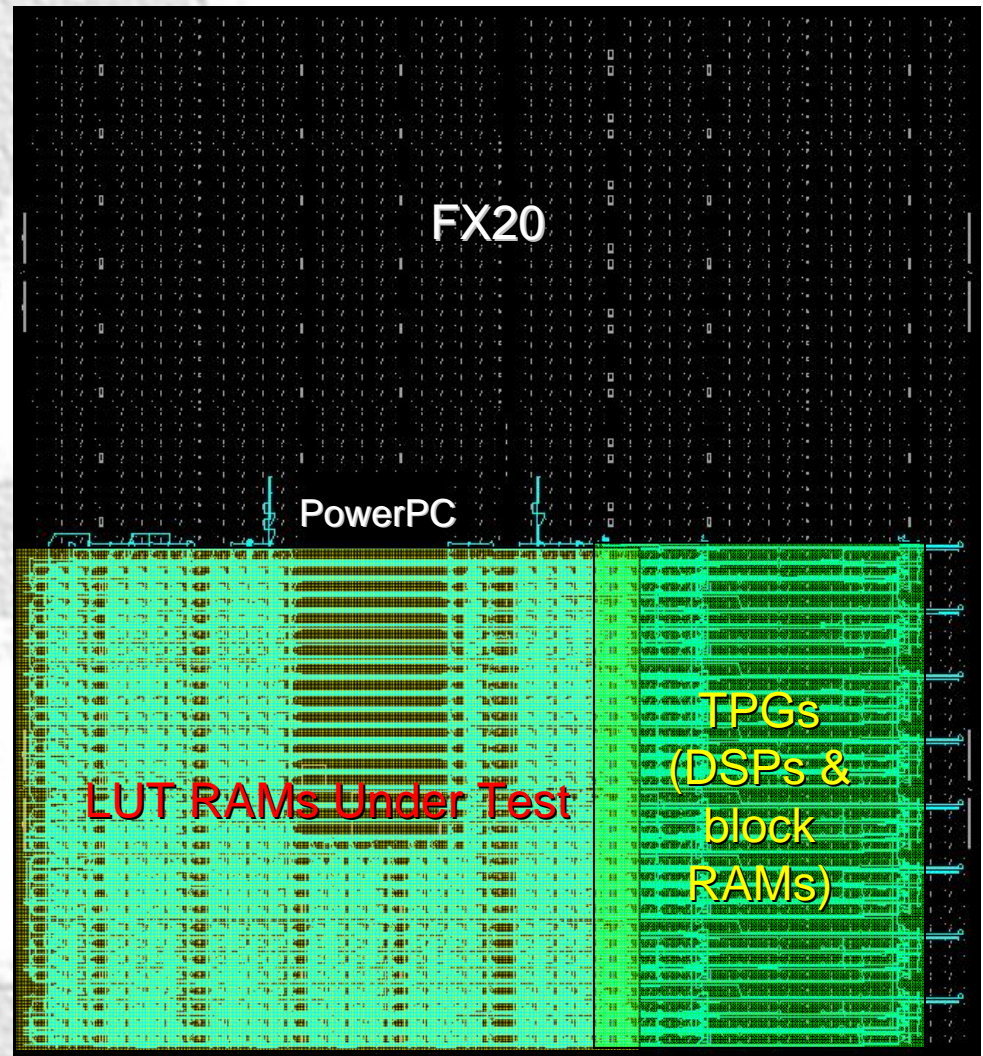
Automated BIST Configurations

- ❑ C program generates .XDL file
- ❑ .XDL to .NCD
 - ❖ `xld -xdl2ncd bist.ncd`
- ❑ FPGA Editor
 - ❖ Design Rule Check
 - ❖ Route with “no pin swap” option
- ❑ .NCD to .BIT
 - ❖ BitGen
 - ❖ Download into FPGA
- ❑ .NCD to .XDL
 - ❖ Modify for subsequent BIST configs in test session



BIST Generation Programs

- User specifies portion of array to test
 - ❖ Start row & column
 - ❖ End row & column
- TPGs generated & located based on rows
- XDL is not routed
 - ❖ FPGA Editor used for routing with “no swap”
 - ✓ PAR use to support “no pin swap” option
- Additional program modifies routed XDL for subsequent BIST configurations



Reducing Test Time

□ Orient BIST architecture to configuration memory

- ❖ Keep routing constant between configurations

□ Downloading BIST configurations

❖ Partial reconfiguration

- ✓ Reduce # frames written between configurations
 - Keep routing constant between configurations
 - Optimize ordering of BIST configurations

□ Retrieving BIST results

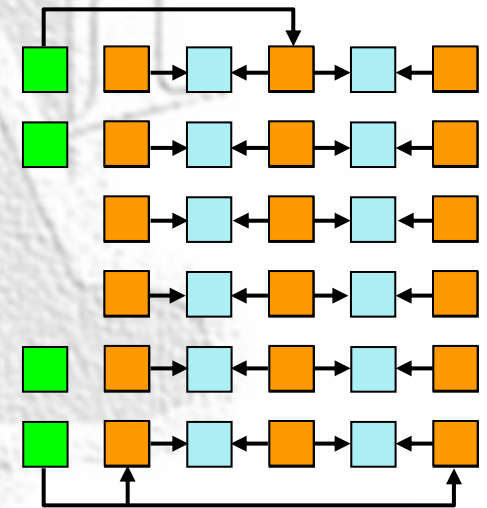
❖ Partial configuration memory readback

- ✓ Eliminates ORA logic for scan chain
 - Allows concurrent testing of more resources

- ✓ Reduce # frames read

❖ Dynamic partial reconfiguration

- ✓ Read BIST results after a series of BIST configurations
 - Slight loss of diagnostic resolution



Reducing Test Time

