Center for
Reliable
Computing

Relative IDDQ Testing and Fault Location for FPGAs

Erik Chmelar
Center for Reliable Computing
Stanford University
December 1, 2003

Motivation

- Devices not perfect
  - Manufacturing defects
- Model defect as fault
  - Stuck-at, stuck open, delay, bridging, etc.
- CMOS circuits
  - Bridging faults occur
- Need effective fault detection technique
  - Relative IDDQ

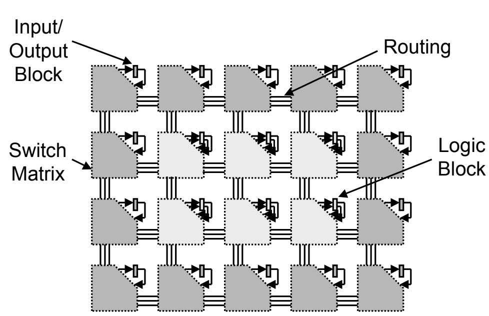
Outline

- ***FPGA***
- Relative IDDQ
- Partitioning
- Fault Location
- Conclusion

FPGA

- Field-programmable Gate Array
  - Configurable hardware platform
    - Programmable logic
      - Combinational and sequential
    - Programmable routing network
      - Interconnects logic
  - Implements arbitrary logic design

Basic Structure



Types

- Application-dependent
  - Fixed configuration
  - Used resources tested
  - Unused resources may be faulty
- Application-independent
  - Configuration unknown
  - All resources tested

Configuration

- Logic design mapped to FPGA
  - Program LUT logic functions, F_{LUT}
  - Set initial conditions of bistables, $init_{bistable}$
  - Program routing network
- Used resource
  - Used in configuration
- Unused resource
  - Not used in configuration

Interconnect Logic Values

- Used interconnect
  - Set by driving LUT or bistable
  - For logic-0: $F_{LUT} = 0$ or $init_{bistable} = 0$
  - For logic-1: $F_{LUT} = 1$ or $init_{bistable} = 1$
- Unused interconnect
  - Never left floating
  - Driven to logic-1 by hardware
    - Xilinx and Altera

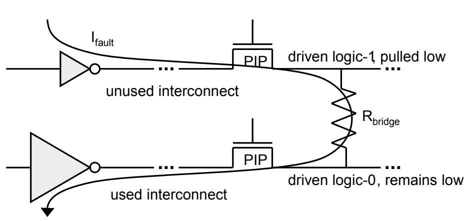
Outline

- FPGA
- ***Relative IDDQ***
- Partitioning
- Fault Location
- Conclusion

Used-unused Bridging Fault

- Bridge joining used interconnect to unused
- Difficult to detect
  - Strong logic-0 dominates weak logic-1
  - Used interconnect dominates unused
    - PIP implemented as NMOS pass transistor
      - Drives strong logic-0
    - Unused interconnects do not float
      - Driven by hardware to weak logic-1

Example



Standard IDDQ

- Activate bridging fault
  - Set internal node logic values
- Deactivate switching of nodes
- Measure leakage current, IDDQ
  - Includes fault current
  - IDDQ > threshold?
    - Yes, fail
    - No, pass

Relative IDDQ
Application-dependent FPGA

- Measure reference current
  - Bridging fault not activated
- Measure total current
  - Bridging fault activated
- Calculate signature current
  - Compare to threshold

Reference Current

- $IDDQ_{ref}$
- Sum of leakage currents
  - e.g. source-to-substrate leakage
  - e.g. sub-threshold drain current
- Set interconnect logic values
  - Used: logic-1
  - Unused: logic-1
  - Bridging fault not activated

Total Current

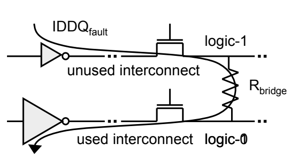
- $IDDQ_{tot} = IDDQ_{ref} + IDDQ_{int} + IDDQ_{fault}$
- Set interconnect logic values
  - Used: logic-0
  - Unused: logic-1
  - $IDDQ_{int}$
    - Leakage between used and unused interconnects
  - $IDDQ_{fault}$
    - Activated bridging fault current

Signature Current

- $IDDQ_{sig} = IDDQ_{tot} – IDDQ_{ref}$
- $= IDDQ_{int} + IDDQ_{fault}$
  - Leakage current cancelled
- $IDDQ_{sig}$ > threshold?
  - Yes, fail
  - No, pass

Example

- Measure $IDDQ_{ref}$
- Measure $IDDQ_{tot}$
- Calculate $IDDQ_{sig}$



Relative IDDQ
Application-independent FPGA

do
  generate test configuration
    set used and unused interconnects
  test as application-dependent FPGA
until
  adequate used-unused fault coverage

Metrics

- Detectability ratio
  - Fault current / current compared to threshold
  - Standard IDDQ
    - $d_s = IDDQ_{fault}/IDDQ_{tot}$
  - Relative IDDQ
    - $d_r = IDDQ_{fault}/IDDQ_{sig}$
- Improvement ratio
  - $i = d_r/d_s = IDDQ_{tot}/IDDQ_{sig}$

Experiment

- Emulate bridging fault using PIP
  - Turn on unused PIP
- Small FPGA
  - XC3S50
- 90 nm manufacturing process

Experimental Results

Dev	IDDQ						d_s	d_r	i
	ref	Fault-free		Faulty		fault			
		tot	sig	tot	sig				
1	1.66	**1.81**	0.15	2.09	0.42	0.27	0.13	0.64	5.0
2	1.72	**1.91**	0.19	2.20	0.48	0.29	0.13	0.60	4.6
3	1.53	**1.66**	0.12	1.93	0.40	0.27	0.14	0.69	4.9
4	**1.90**	**2.07**	**0.17**	**2.35**	**0.45**	**0.28**	**0.12**	**0.63**	**5.2**
5	1.60	**1.74**	0.14	2.01	0.41	0.27	0.14	0.66	4.9
units	mA	mA	mA	mA	mA	mA	-	-	-

Outline

- FPGA
- Relative IDDQ
- ***Partitioning***
- Fault Location
- Conclusion

Scaling and IDDQ

- IDDQ increases with FPGA size
  - $IDDQ_{tot}$ increases faster than $IDDQ_{ref}$
    - More PIPs => more leakage
  - $IDDQ_{sig}$ increases
- Detectability ratio decreases
  - More difficult to detect fault
- Need scalable IDDQ technique
  - Partitioning

Definitions

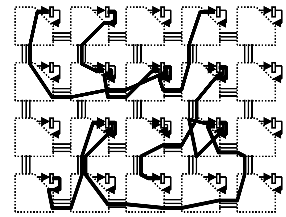
- Net
  - Routing resources between logic elements
    - Interconnect, PIP, MUX, buffer, via
- Suspected fault net
  - Net possibly bridged to another net
- S
  - Set of suspected faulty nets
  - e.g. $S = \{net_1, net_2, ..., net_M\}$

Partitioning

- Divide S into N partitions, $P_1, \ldots, P_N$
  - $S = \{net_1, net_2, \ldots, net_M\}$
  - $S = \{P_1, P_2, \ldots, P_N\}$
  - Each partition
    - Contains some nets of S
  - Each net of S
    - Must be contained in at least one partition

Example

- S
- P_1, P_2, P_3

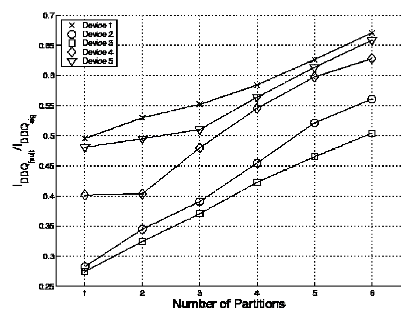


Configurations

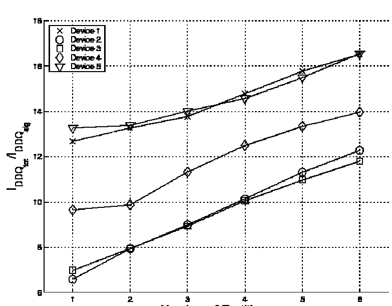
- Measure $IDDQ_{tot-i}$ for each P_i
  - N partitions, N configurations
    - Set interconnect logic values

Cfg.	logic-0	logic-1
1	P_1	$P_2, \ldots, P_N$, unused
2	P_2	$P_1, P_3, \ldots, P_N$, unused
...	...	...
N	P_N	$P_1, \ldots, P_{N-1}$, unused

Detectability Ratio, d_r



Improvement Ratio, i



Outline

- FPGA
- Relative IDDQ
- Partitioning
- ***Fault Location***
- Conclusion

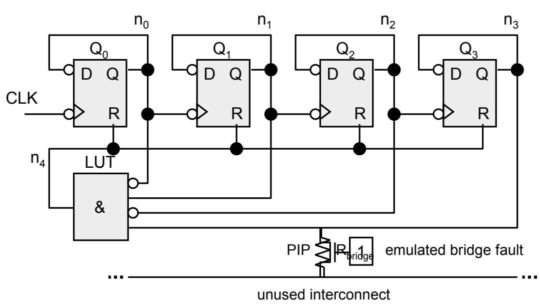
Motivation

- Fault diagnosis
  - Improve manufacturing process
  - Fix design errors
  - Increase yield
- Fault location
  - Necessary for most fault diagnosis
- Need efficient fault location technique
  - Relative IDDQ and partitioning

Fault Location Algorithm

while |S| > 1
  $(P_1 \ldots P_N)$ = partition(S,N)
  foreach P_i
    measure $IDDQ_{sig-i}$
  $IDDQ_{sig-max} = \max(IDDQ_{sig-1} \ldots IDDQ_{sig-N})$
  foreach P_i
    if $IDDQ_{sig-i} < IDDQ_{sig-max}$
      $S = S - P_i$

Example
Decade Counter



Example
Experimental Results

- Binary Search
  - N = 2 partitions per iteration

It.	Partition 1		Partition 2		S After Elimination
	P_1	$IDDQ_{sig-1}$ (mA)	P_2	$IDDQ_{sig-2}$ (mA)	
0	n_0,n_1,n_2,n_3,n_4	0.21	-	-	n_0,n_1,n_2,n_3,n_4
1	n_0,n_1	0.00	n_2,n_3,n_4	0.21	n_2,n_3,n_4
2	n_2	0.00	n_3,n_4	0.21	n_3,n_4
3	n_3	0.21	n_4	0.00	n_3

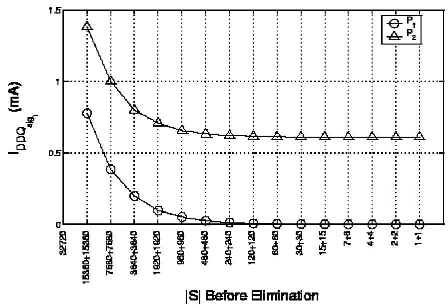
Complexity

- Binary search of S
  - $\lceil \log_2 |S| \rceil$ iterations
- Number of configurations
  - One total to measure $IDDQ_{ref}$
  - Two per iteration to measure $IDDQ_{tot-i}$
    - Set nets of P_1 to logic-0
      - Measure $IDDQ_{tot-1}$
    - Set nets of P_2 to logic-0
      - Measure $IDDQ_{tot-2}$

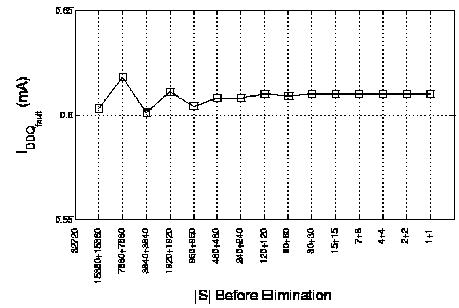
Experimental Results

- Large FPGA
  - XC3S1000
- At start |S| = 30720 nets
  - $\lceil \log_2 30720 \rceil$ = 15 search iterations
  - 2*15+1 = 31 device configurations
- At finish |S| = 1
  - S = {net_i}
    - The faulty net

IDDQ$_{sig-1}$, IDDQ$_{sig-2}$



IDDQ$_{fault}$



Outline

- FPGA
- Relative IDDQ
- Partitioning
- Fault Location
- ***Conclusion***

Conclusion

- Utilize FPGA configurability
- Relative IDDQ and partitioning
  - Detect bridging fault
  - Increase detectability ratio
- Fault location
  - Locate bridging fault
  - Simple
  - Automatic