Placement of Repair Circuits for In-Field FPGA Repair

Mike Wirthlin, BYU

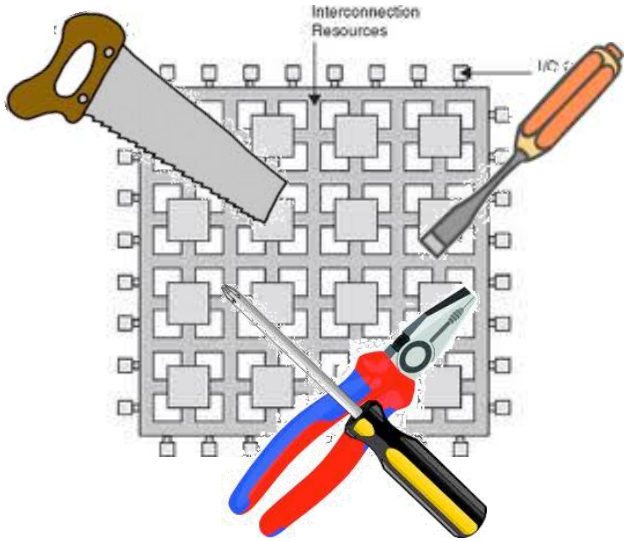
Josh Jensen, BYU

Alex Wilson, BYU

Will Howes, BYU, LLNL

Shi-Jie Wen, Cisco Systems

Rick Wong, Cisco Systems

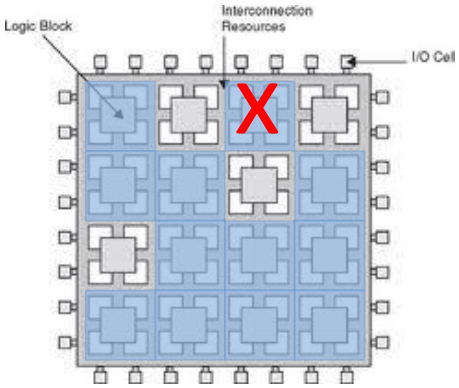


This work was supported by Cisco Systems and the NSF Center for High-Performance Reconfigurable computing (CHREC)

Motivation

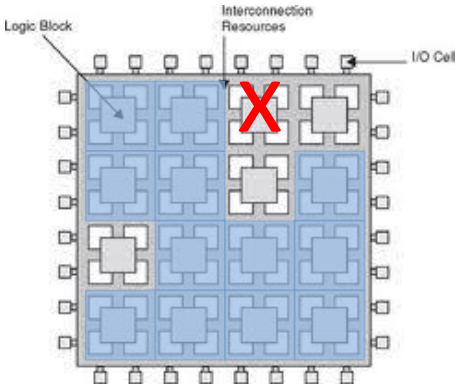
- Difficult to manufacture FPGAs with no faults
- FPGAs may “wear out” (i.e. permanent faults)
  - Electromigration, Hot Carrier Injection
  - Time-dependent dielectric breakdown (TDDB)
- FPGAs can be “repaired”
  - Modify design (bitstream) to avoid faulty resource
  - Exploit idle, unused resources for fault repair

Motivation



- FPGAs can be “repaired”
  - Modify design (bitstream) to avoid faulty resource
  - Exploit idle, unused resources for fault repair

Motivation



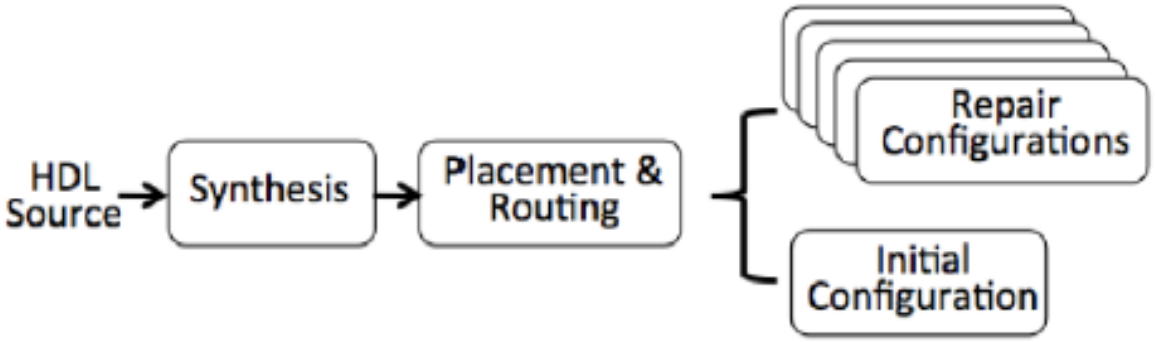
- FPGAs can be “repaired”
  - Modify design (bitstream) to avoid faulty resource
  - Exploit idle, unused resources for fault repair

Related Work

- Yield Enhancement
  - Reserve spare resources for repairing functional faults
    - Rows/columns of spare cells
    - Routing tracks
  - Identify faults at power up using online diagnostics
  - Automatically replace functional faults with spares
- Increment Design Methods
  - Preserve the placement and routing data structures
  - Use placement and routing data to *quickly* generate a repair
- Embedded Bitstreams
  - Embed test structures within the bitstream
  - Embed alternative path configurations within bitstream
  - Make simple decisions on configuration at run-time

Repair Approach

- Create initial bitstream for fault-free device
- Create repair bitstreams for *every* fault
  - Repairs created *before* FPGA deployment
  - Apply repairs *after* FPGA deployment

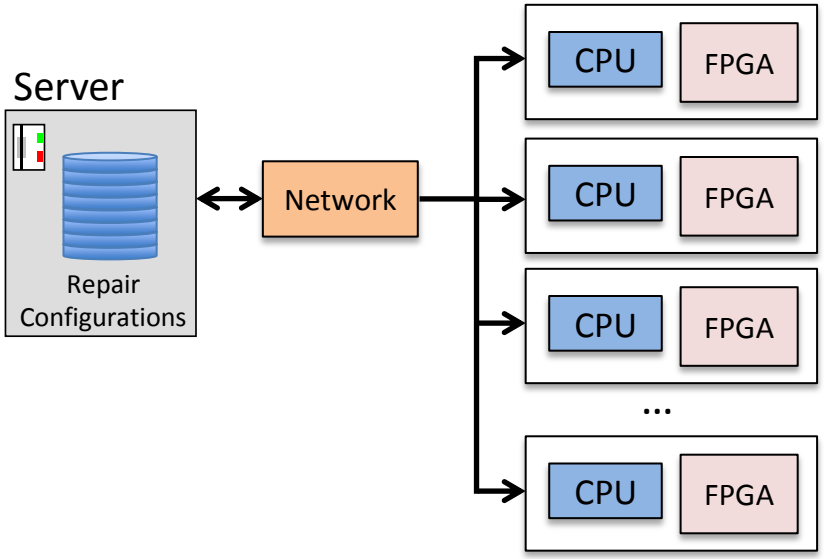


Repair Approach

- Create initial bitstream for fault-free device
- Create repair bitstreams for *every* fault
  - Repairs created *before* FPGA deployment
  - Apply repairs *after* FPGA deployment
- Pros
  - No computation required in field to create repair
  - Repairability determined before deployment
- Cons
  - Requires greater repair bitstream storage
  - Computationally intensive

In-Field Repair Scenario

- Same FPGA design is placed in *many* products
- Product has internet connectivity
- Server can provide FPGA repair bitstreams



BYU
BRIGHAM YOUNG UNIVERSITY

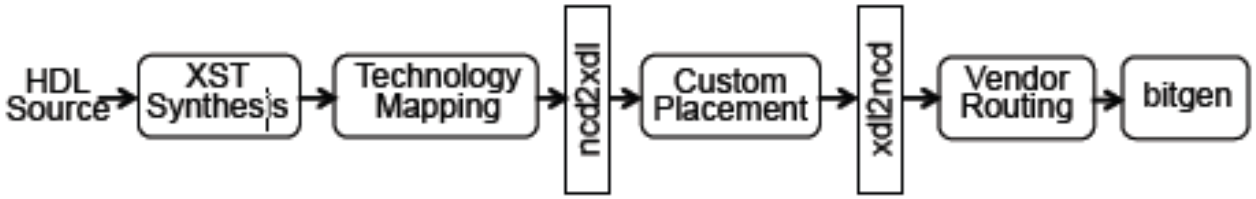
CISCO

Repair Focus: Placement

- Create unique “repair” placement configurations
  - Repair faults within logic (CLB), BRAM, and DSP
- Goals of repair placer:
  - Identify as few placement repairs as possible
  - Preserve placement quality for all repairs
  - Minimize time required to generate repair set
- Three placement algorithms created
  - Naïve, Cost-Repair, and Shadow Placement
- Full FPGA repair will require “repair routing” in addition to placement (ongoing effort)

Baseline placer

- Written in Java and built on RapidSmith

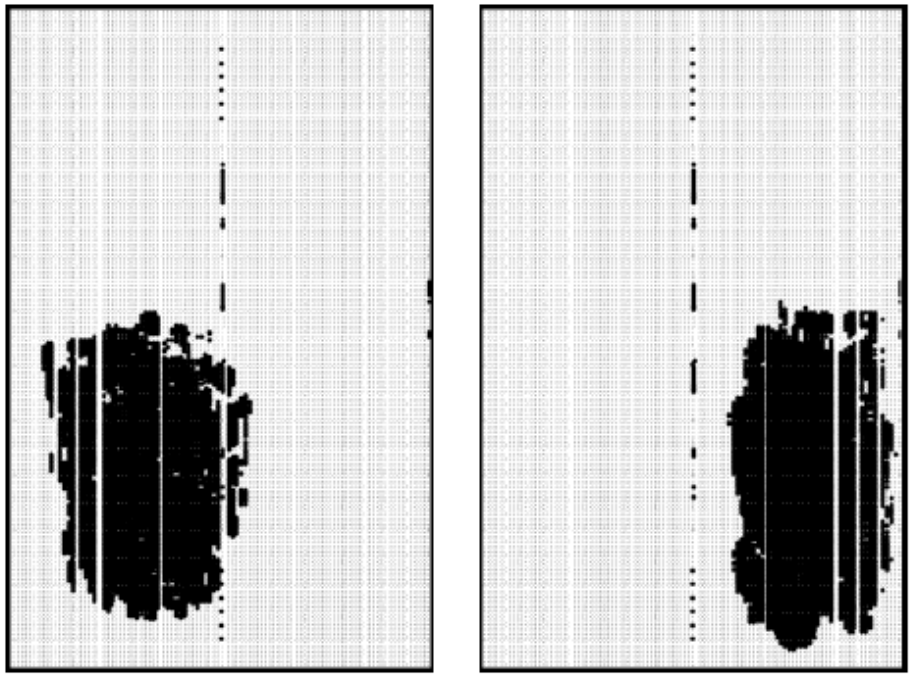


- Targets Xilinx, V4 Architecture (valid bitstreams)
- Based on simulated annealing/VPR

	Baseline RapidSmith			Xilinx		
Design	Time	Cost	ns	Time	Cost	ns
top	.71	200	3.3	7	127	2.9
test0	4.4	5834	7.6	16	6115	6.7
mult18	35.7	9160	2.5	15	10866	2.3
crazy	51.5	73731	4.4	35	54280	3.4
multxor	3937	348792	3.9	110	321098	3.0

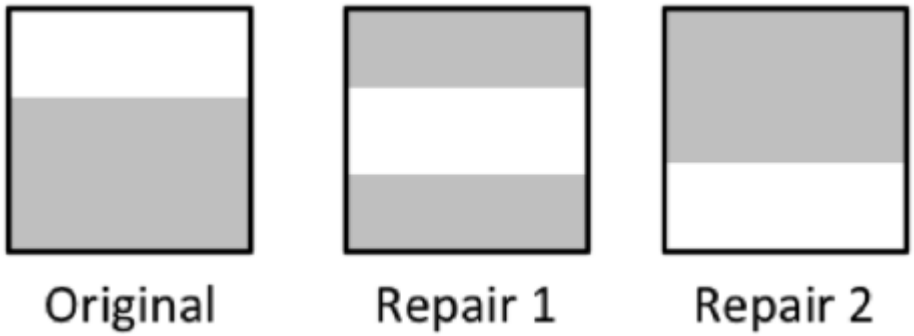
Repair and FPGA Utilization

- The number of repairs depends on the utilization of the design
- Designs with less than 50% utilization only need *one* repair configuration



FPGA Utilization

- Two Repair Circuits for 67% Utilized Design

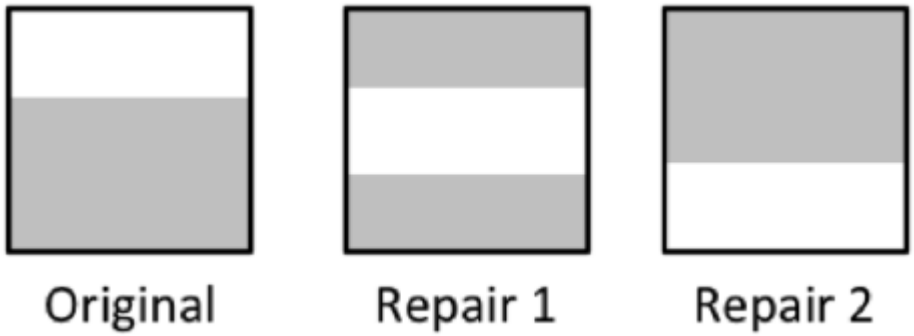


- The higher the utilization, the more repair placements that are required:

$$N_{min} = \left\lceil \frac{A}{I} \right\rceil = \left\lceil \frac{A}{R-A} \right\rceil = \left\lceil \frac{1}{R/A-1} \right\rceil = \left\lceil \frac{u}{1-u} \right\rceil$$

FPGA Utilization

- Two Repair Circuits for 67% Utilized Design



- The higher the utilization, the more repair placements that are required:

Example, u=.95: $N_{min} \geq .95/(.05) = 19$

Benchmark Circuits and Constraints

	Constraint				
Design	A 50%	B 75%	C 90%	D 95%	E 99%
top	4,16	3,16	2,16	N/A	N/A
test0	24,26	16,26	14,25	9,37	11,29
mult18	35,38	25,35	25,29	6,43	20,33
crazy	N/A	47,126	47,126	47,126	47,118
multxor	140,152	99,142	105,112	116,96	115,93

BYU
BRIGHAM YOUNG
UNIVERSITY

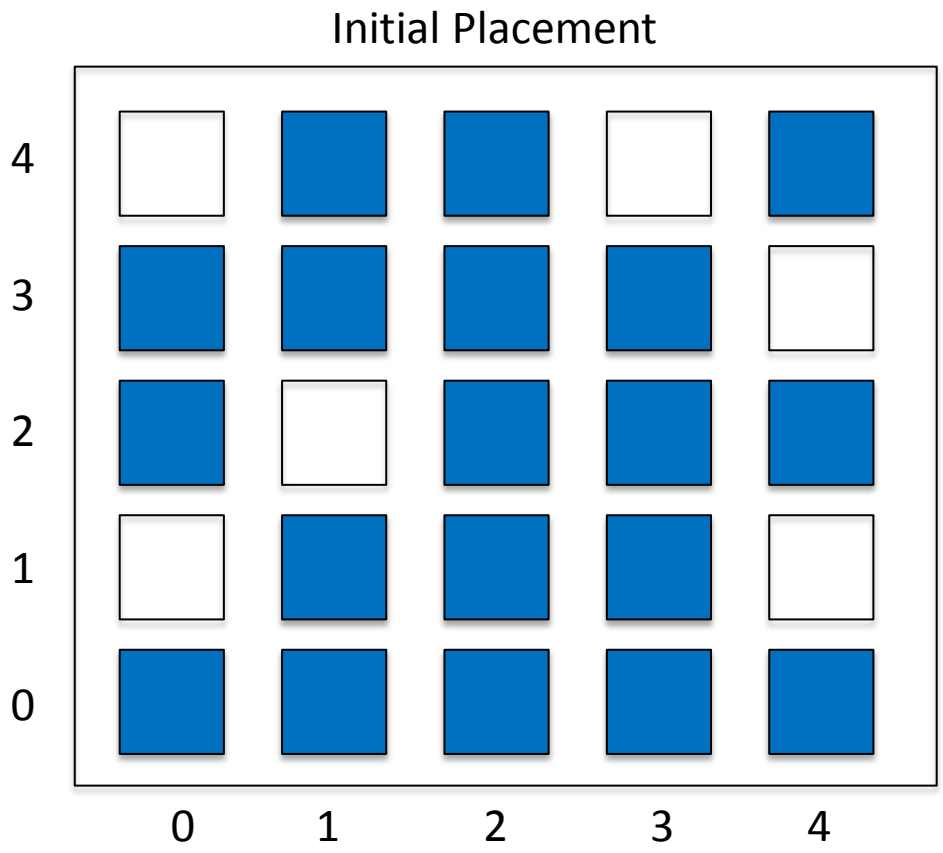
cisco

Naïve Repair Placement

Algorithm 1 Naive Repair Placement

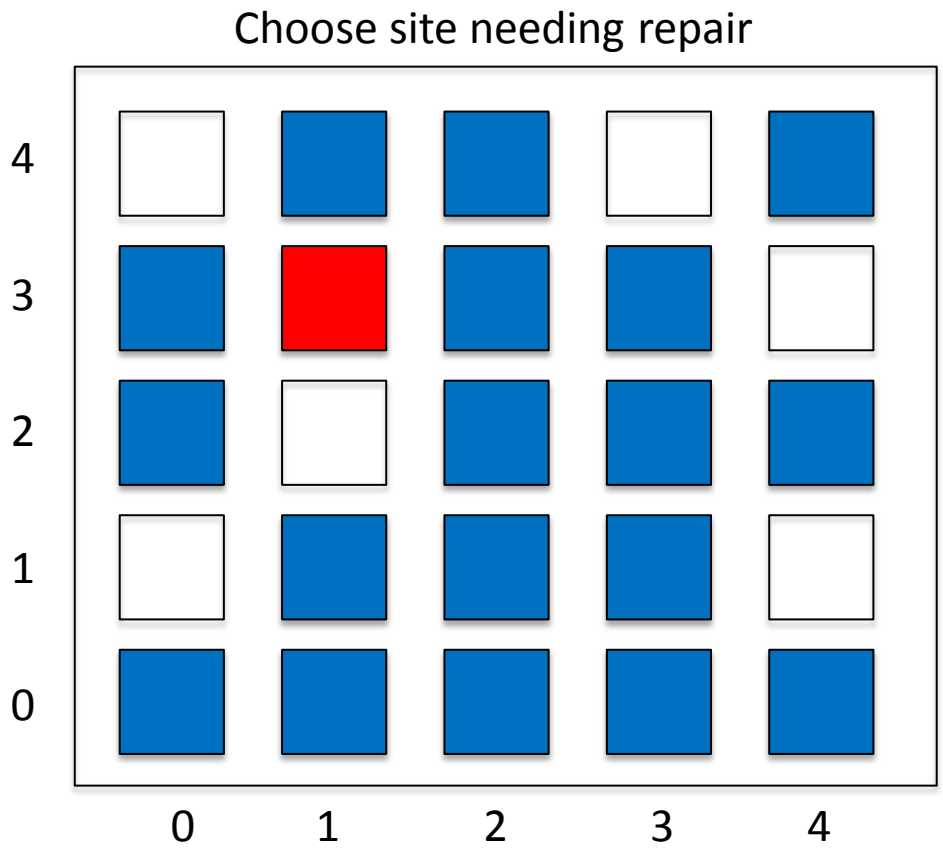
1: $D \leftarrow$ set of all possible placement sites
2: Perform initial placement
3: $S \leftarrow$ set of occupied sites in initial placement
4: **while** $S \neq \emptyset$ **do**
5: choose $s \in S$, remove s from S
6: Remove site s from device database
7: Perform repair placement (site s not available)
8: $R \leftarrow$ set of occupied sites in repair placement
9: $G \leftarrow (D - R) \cap S$ (sites repaired)
10: $S \leftarrow S - G$
11: Add site s into device database
12: **end while**

Naïve Repair Placement



Sites needing repair
(0,0) (0,2) (0,3) (1,0)
(1,1) (1,3) (1,4) (2,0)
(2,1) (2,2) (2,3) (2,4)
(3,0) (3,1) (3,2) (3,4)
(4,0) (4,2) (4,4)

Naïve Repair Placement

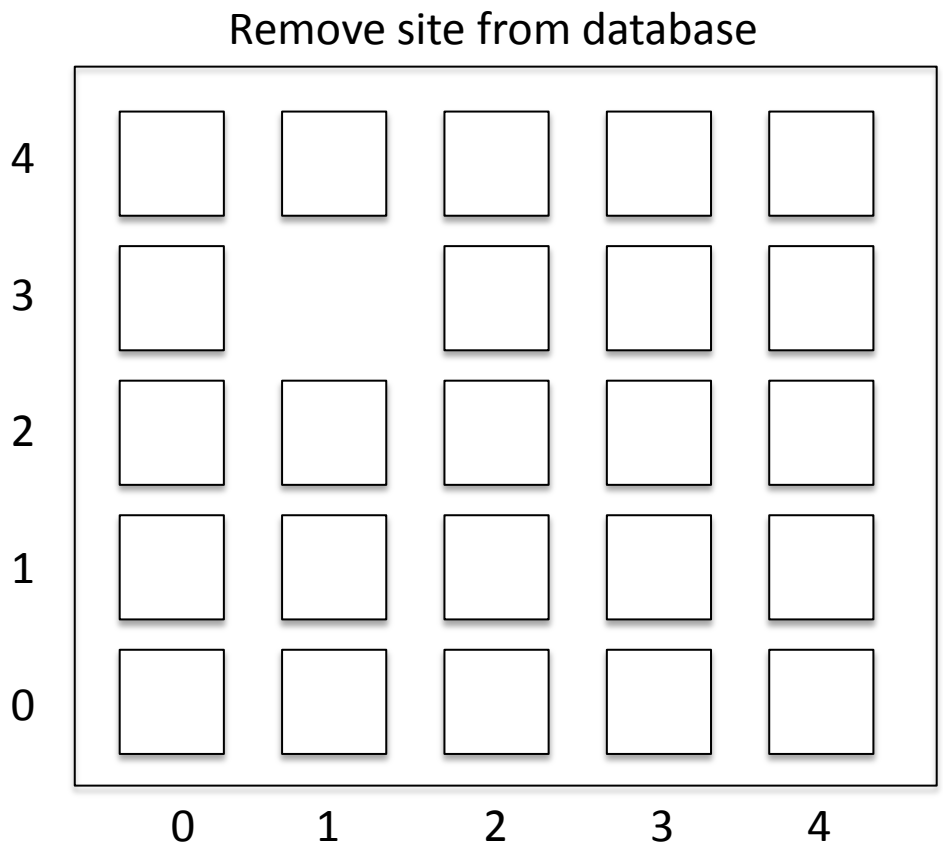


Sites needing repair
(0,0) (0,2) (0,3) (1,0)
(1,1) (1,3) (1,4) (2,0)
(2,1) (2,2) (2,3) (2,4)
(3,0) (3,1) (3,2) (3,4)
(4,0) (4,2) (4,4)

BYU
BRIGHAM YOUNG
UNIVERSITY

CISCO

Naïve Repair Placement

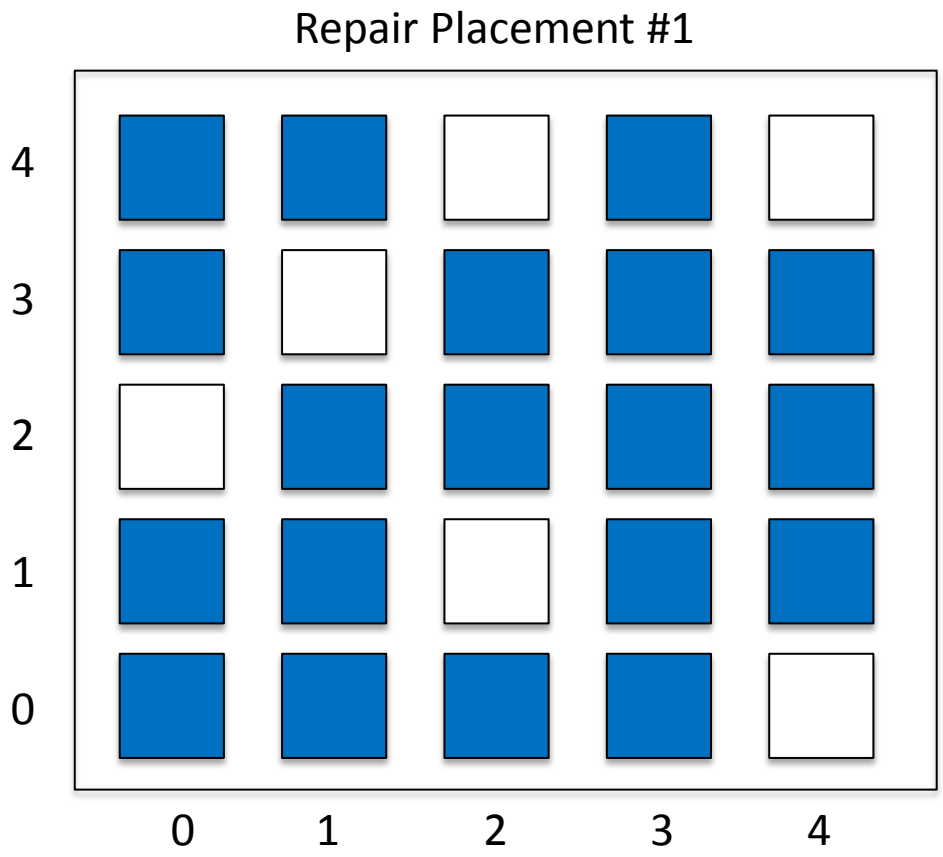


Sites needing repair
(0,0) (0,2) (0,3) (1,0)
(1,1) (1,3) (1,4) (2,0)
(2,1) (2,2) (2,3) (2,4)
(3,0) (3,1) (3,2) (3,4)
(4,0) (4,2) (4,4)

BYU
BRIGHAM YOUNG
UNIVERSITY

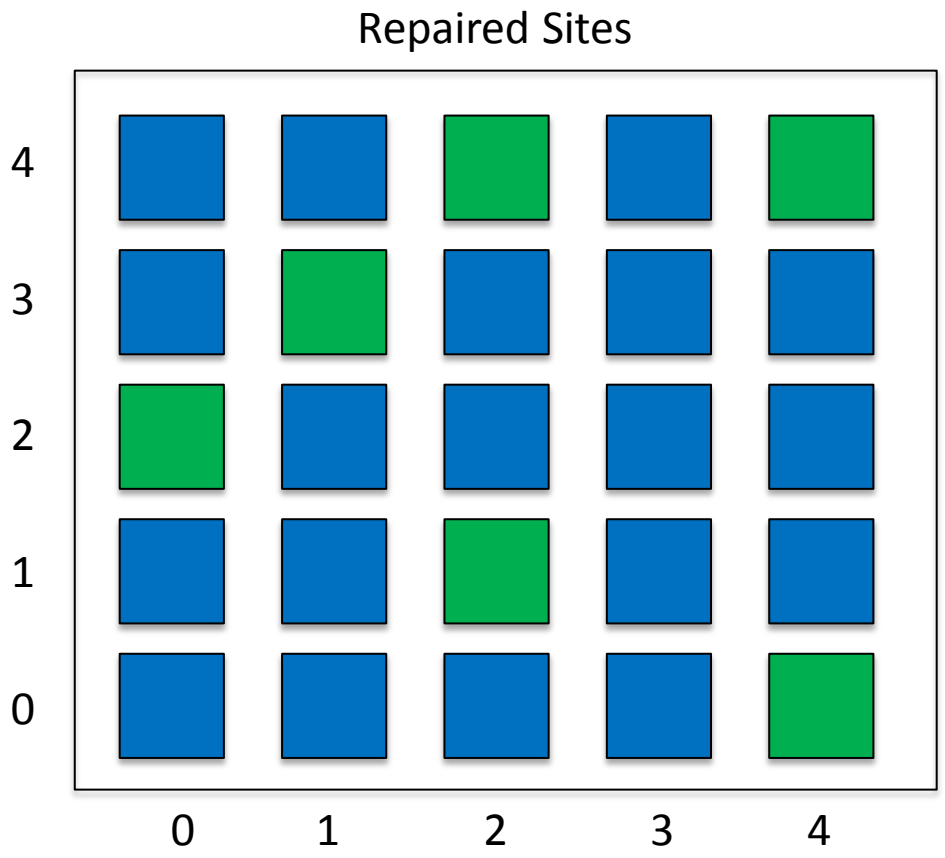
CISCO

Naïve Repair Placement



Sites needing repair
(0,0) (0,2) (0,3) (1,0)
(1,1) (1,3) (1,4) (2,0)
(2,1) (2,2) (2,3) (2,4)
(3,0) (3,1) (3,2) (3,4)
(4,0) (4,2) (4,4)

Naïve Repair Placement



Sites needing repair

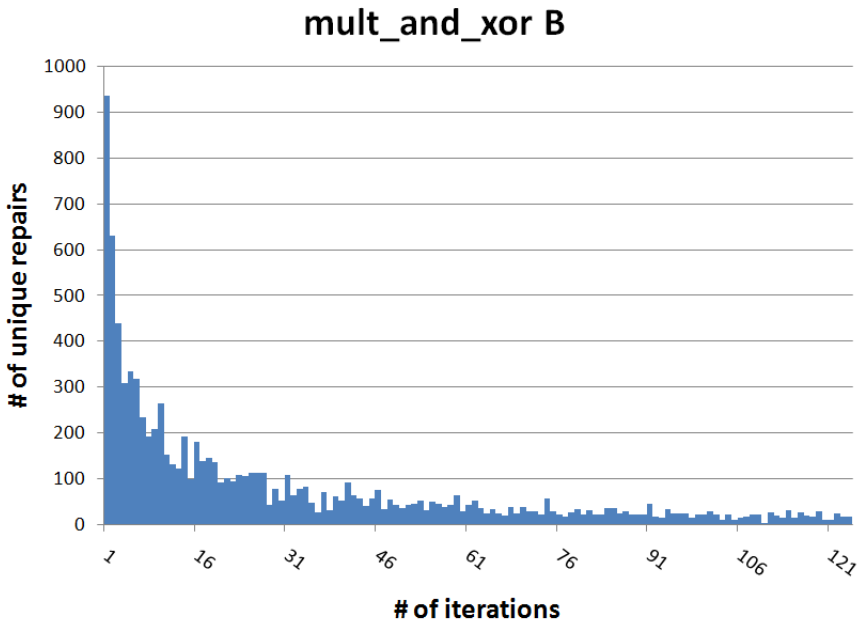
(0,0) (0,3) (1,0)
(1,1) (1,4) (2,0)
(2,2) (2,3)
(3,0) (3,1) (3,2) (3,4)
(4,2)

Naïve Replacement Results

Number of repairs for each constraint

Design	Constraint 0	A	B	C	D	E
top	1	4	6	31	N/A	N/A
test0	10	40	71	112	149	243
mult18	2	51	142	294	340	467
crazy	N/A	76	114	224	413	929
multxor	47	150	441	*	*	*
Time	25x	56x	132x	152x	243x	468x

Number of repairs in each iteration



Cost = 1.02x cost of baseline placer

BYU
BRIGHAM YOUNG
UNIVERSITY

CISCO

Cost Repair Placement

Algorithm 2 Cost Repair Placement

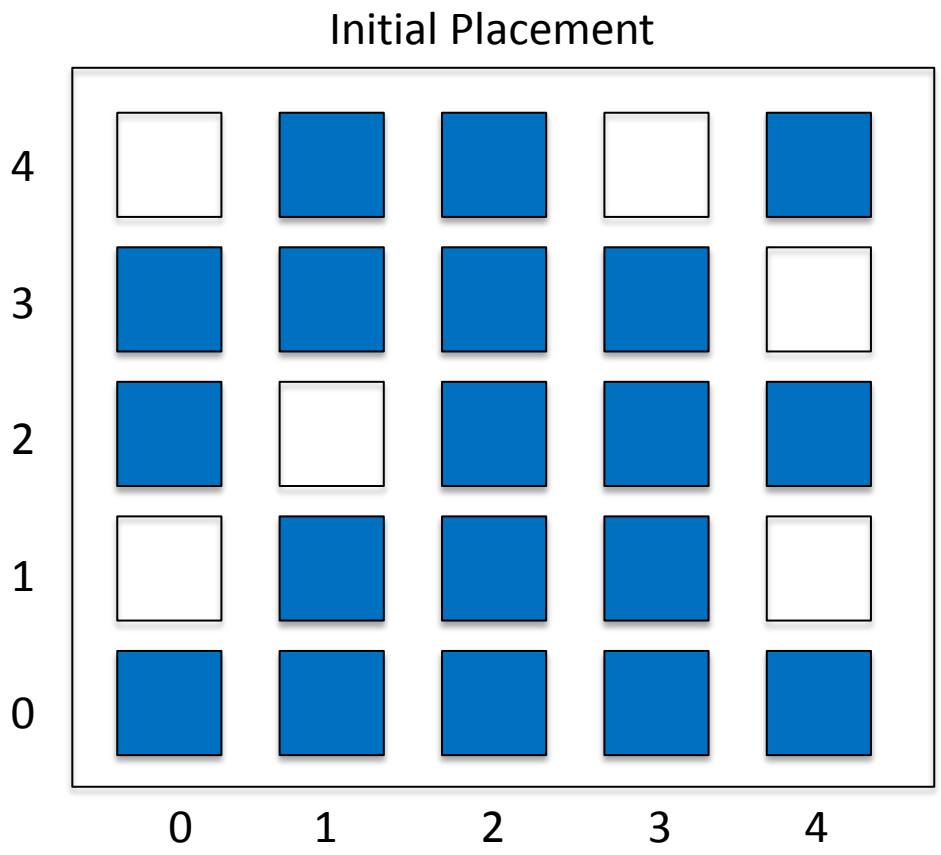
1: $D \leftarrow$ set of all possible placement sites
2: Perform initial placement
3: $S \leftarrow$ set of occupied sites in initial placement
4: **while** $S \neq \emptyset$ **do**
5: choose $s \in S$, remove s from S
6: Remove site s from device database
7: Perform repair placement (site s not available)
8: $R \leftarrow$ set of occupied sites in repair placement
9: $G \leftarrow (D - R) \cap S$ (sites repaired)
10: $S \leftarrow S - G$
11: Increase cost of each site in S
12: Add site s into device database
13: **end while**

$$Cost = \sum_{i \in AllNets} q(i) \cdot (bb_x + bb_y) + \sum_{j \in Sites} c_j$$

BYU
BRIGHAM YOUNG
UNIVERSITY

cisco

Cost Repair Placement

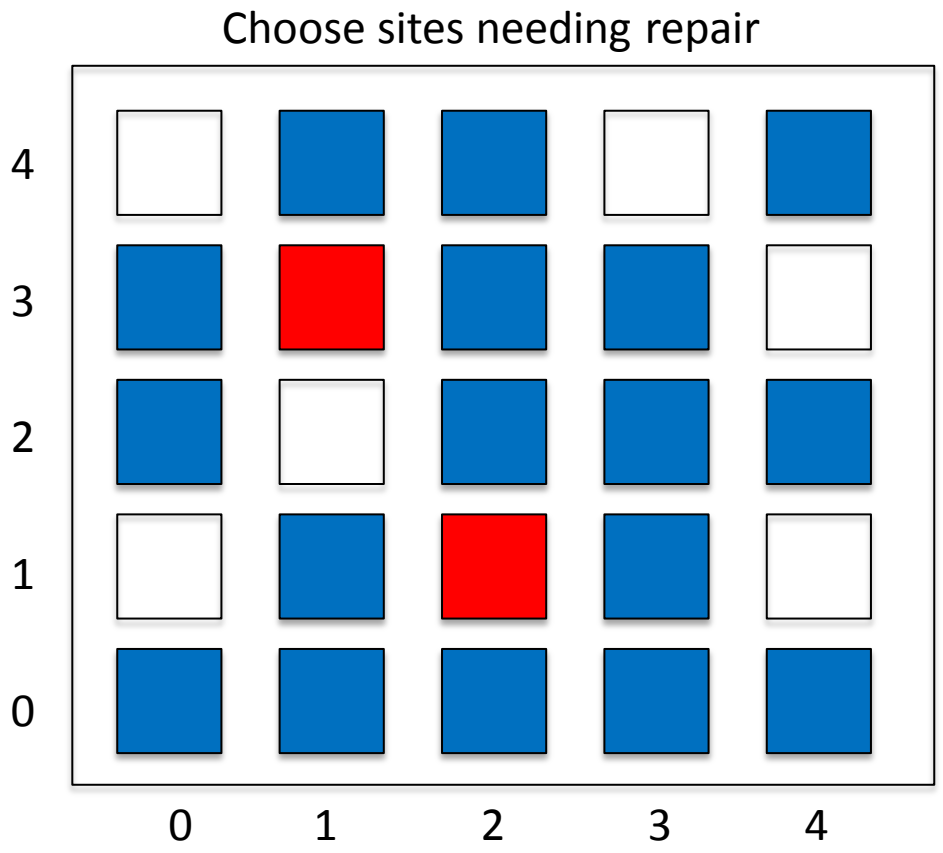


Sites needing repair
(0,0) (0,2) (0,3) (1,0)
(1,1) (1,3) (1,4) (2,0)
(2,1) (2,2) (2,3) (2,4)
(3,0) (3,1) (3,2) (3,4)
(4,0) (4,2) (4,4)

BYU
BRIGHAM YOUNG
UNIVERSITY

CISCO

Cost Repair Placement



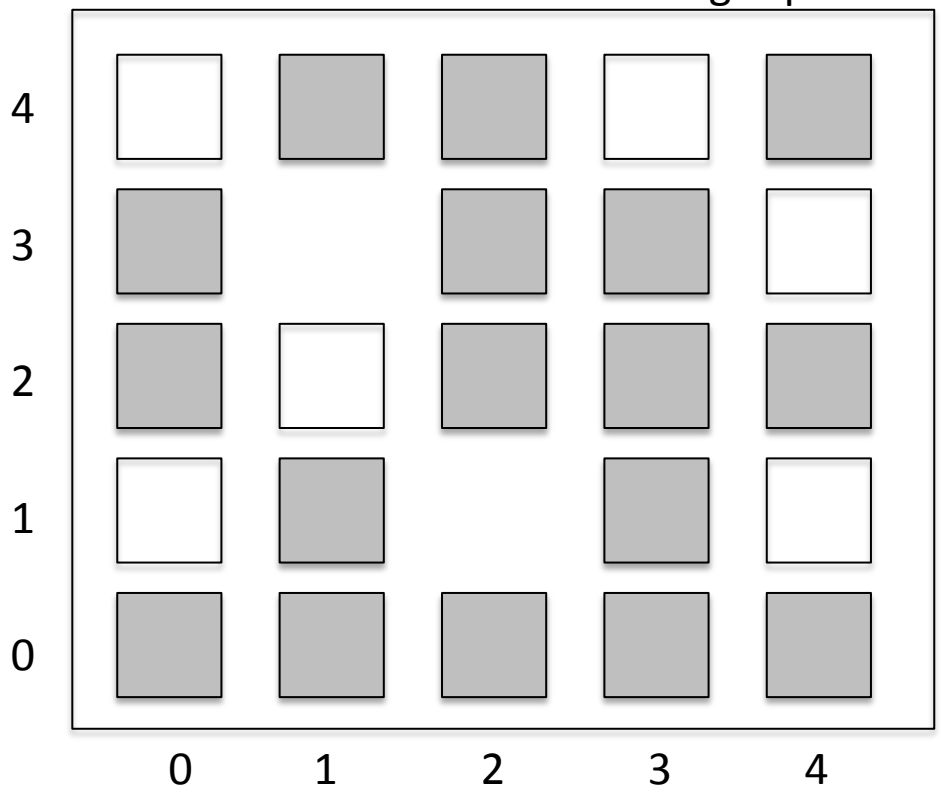
Sites needing repair
(0,0) (0,2) (0,3) (1,0)
(1,1) (1,3) (1,4) (2,0)
(2,1) (2,2) (2,3) (2,4)
(3,0) (3,1) (3,2) (3,4)
(4,0) (4,2) (4,4)

BYU
BRIGHAM YOUNG
UNIVERSITY

CISCO

Cost Repair Placement

Remove sites from database and
increase cost for sites needing repair

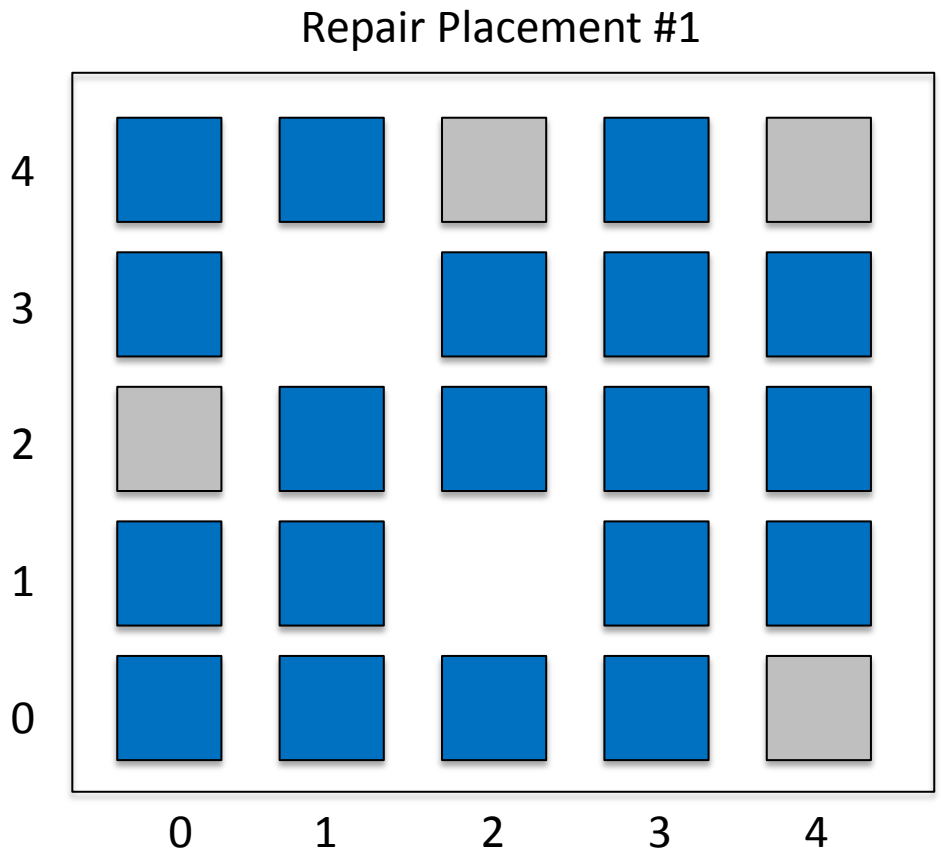


Sites needing repair
(0,0) (0,2) (0,3) (1,0)
(1,1) (1,3) (1,4) (2,0)
(2,1) (2,2) (2,3) (2,4)
(3,0) (3,1) (3,2) (3,4)
(4,0) (4,2) (4,4)

BYU
BRIGHAM YOUNG
UNIVERSITY

CISCO

Cost Repair Placement

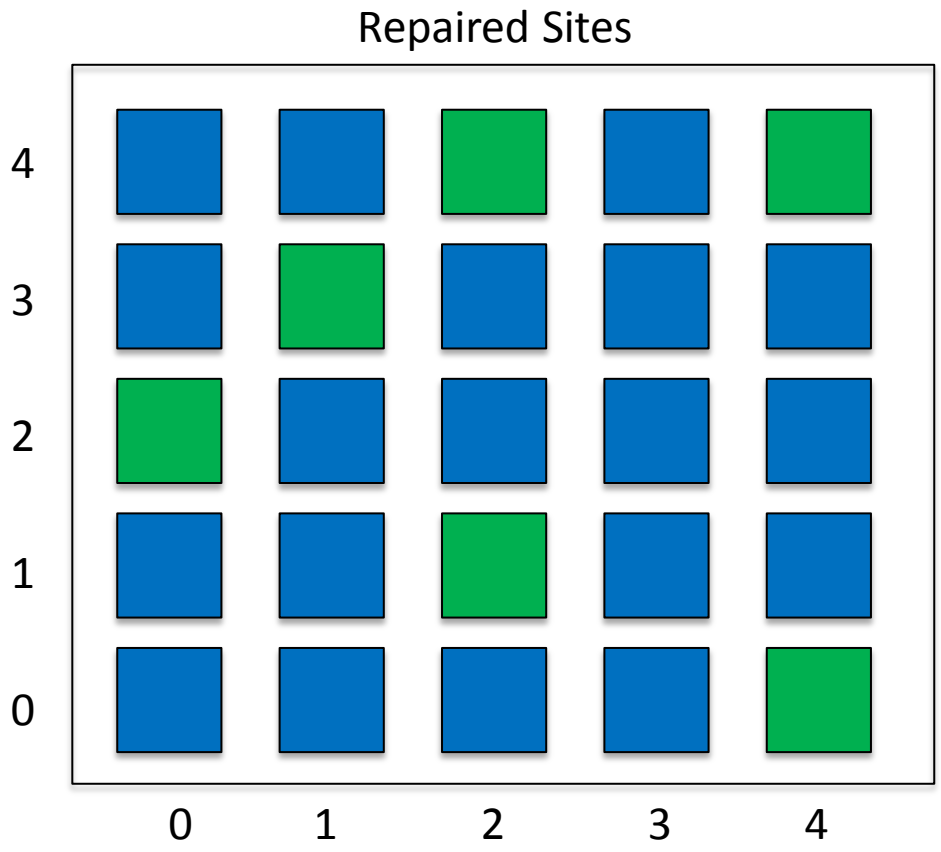


Sites needing repair
(0,0) (0,2) (0,3) (1,0)
(1,1) (1,3) (1,4) (2,0)
(2,1) (2,2) (2,3) (2,4)
(3,0) (3,1) (3,2) (3,4)
(4,0) (4,2) (4,4)

BYU
BRIGHAM YOUNG
UNIVERSITY

CISCO

Cost Repair Placement



Sites needing repair

(0,0) (0,3) (1,0)
(1,1) (1,4) (2,0)
(2,2) (2,3)
(3,0) (3,1) (3,2) (3,4)
(4,2)

BYU
BRIGHAM YOUNG
UNIVERSITY

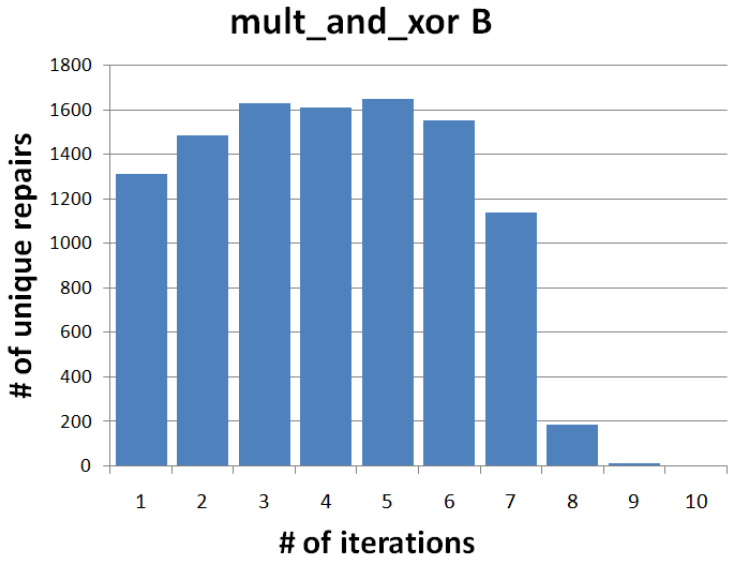
CISCO

Cost Repair Placement Results

Number of repairs for each constraint

Design	Constraint 0	A	B	C	D	E
top	1	2	4	16	N/A	N/A
test0	12	7	11	10	17	85
mult18	4	5	11	11	18	111
crazy	15	14	15	16	21	32
multxor	8	9	10	12	21	108
Time	8.7x	8.2x	12.9x	14.8x	22.1x	28.5x

Number of repairs in each iteration



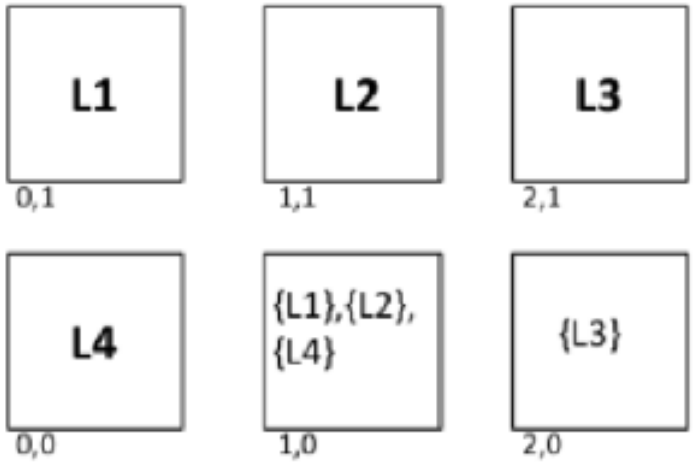
Cost = 1.05x cost of baseline placer

BYU
BRIGHAM YOUNG
UNIVERSITY

CISCO

Shadow Placement

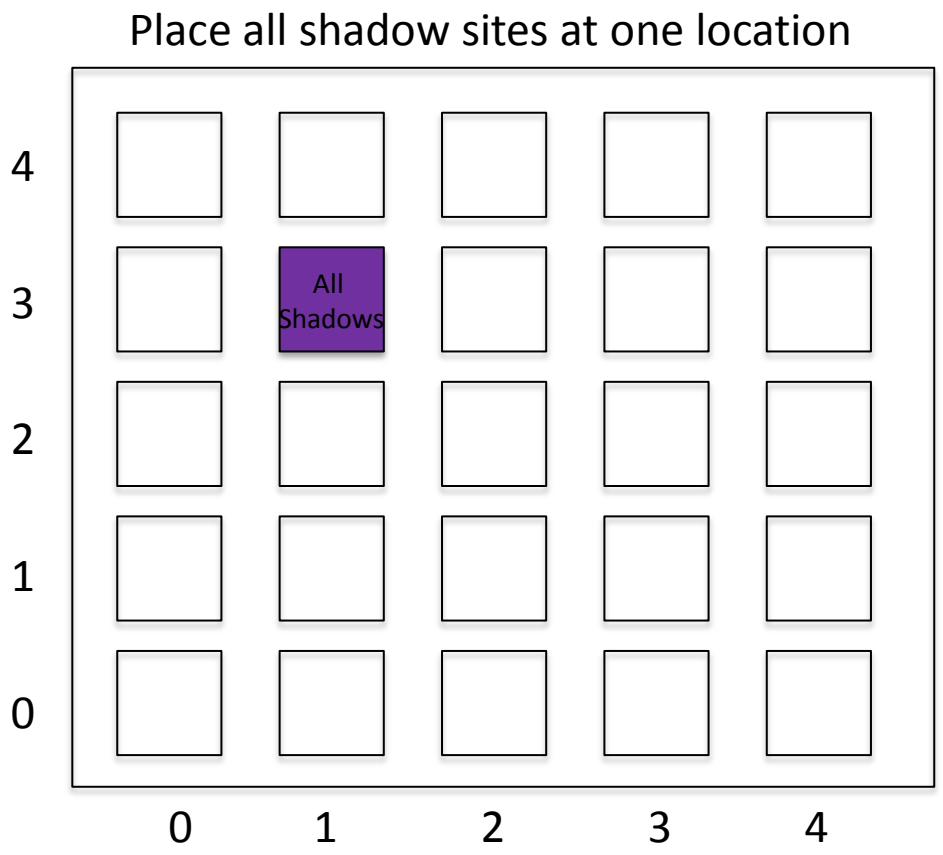
- Goals
  - reduce the number of placement iterations
  - reduce computation time



BYU
BRIGHAM YOUNG
UNIVERSITY

CISCO

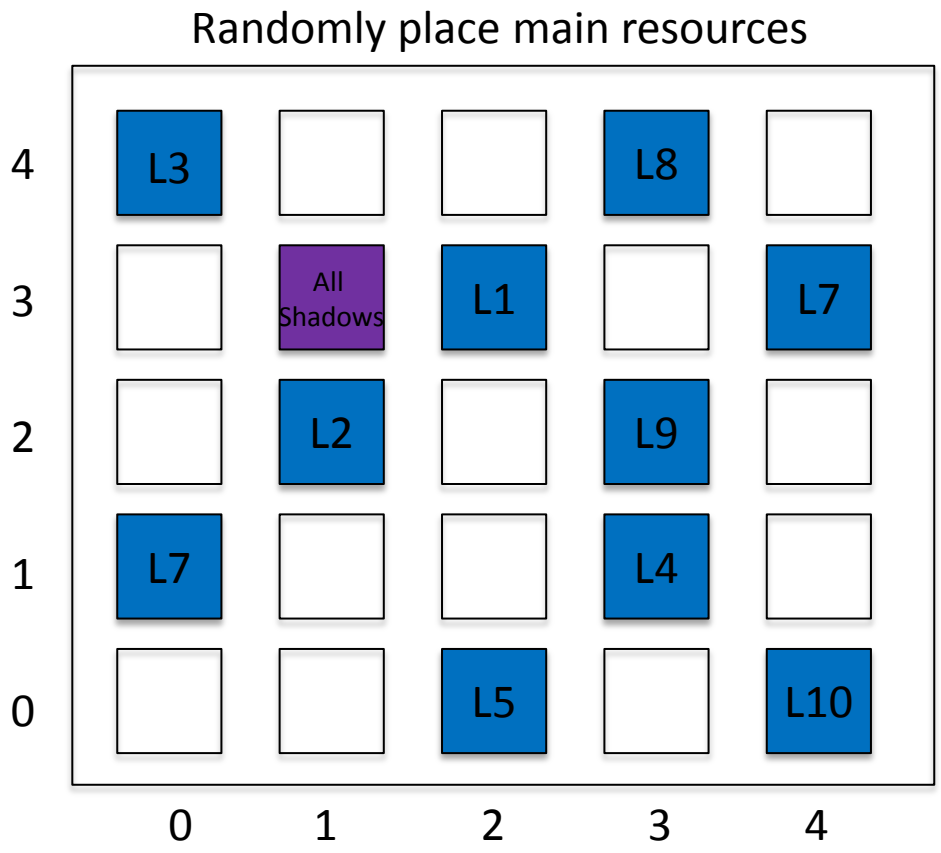
Shadow Placement



BYU
BRIGHAM YOUNG UNIVERSITY

CISCO

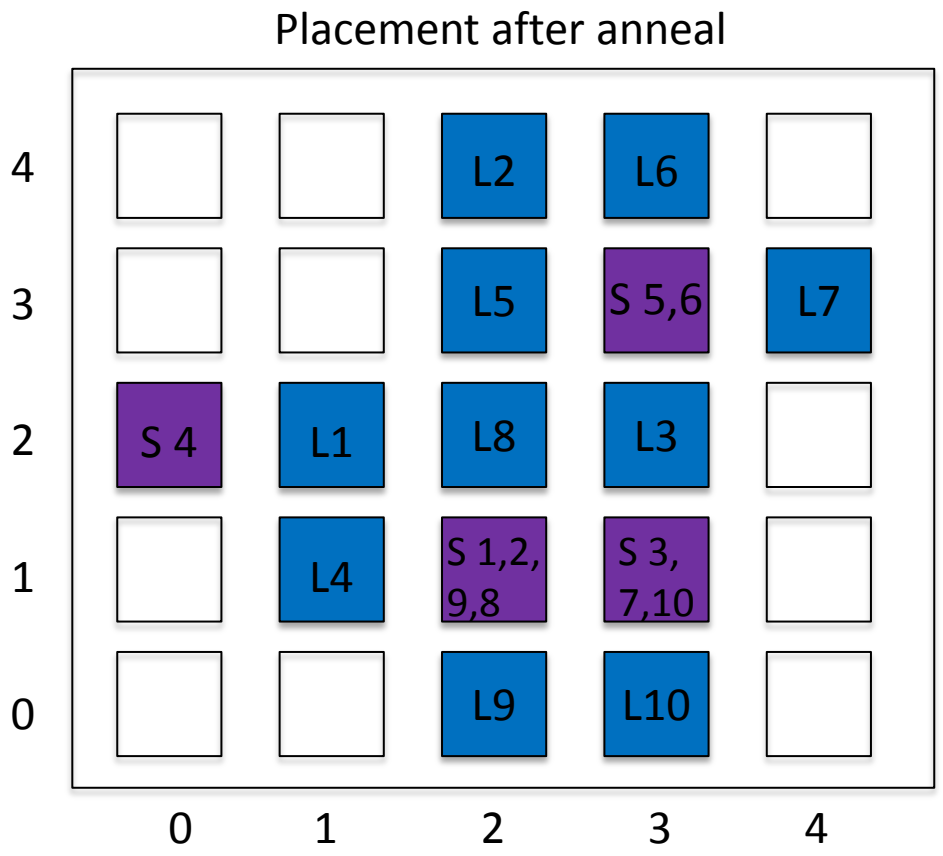
Shadow Placement



BYU
BRIGHAM YOUNG UNIVERSITY

CISCO

Shadow Placement

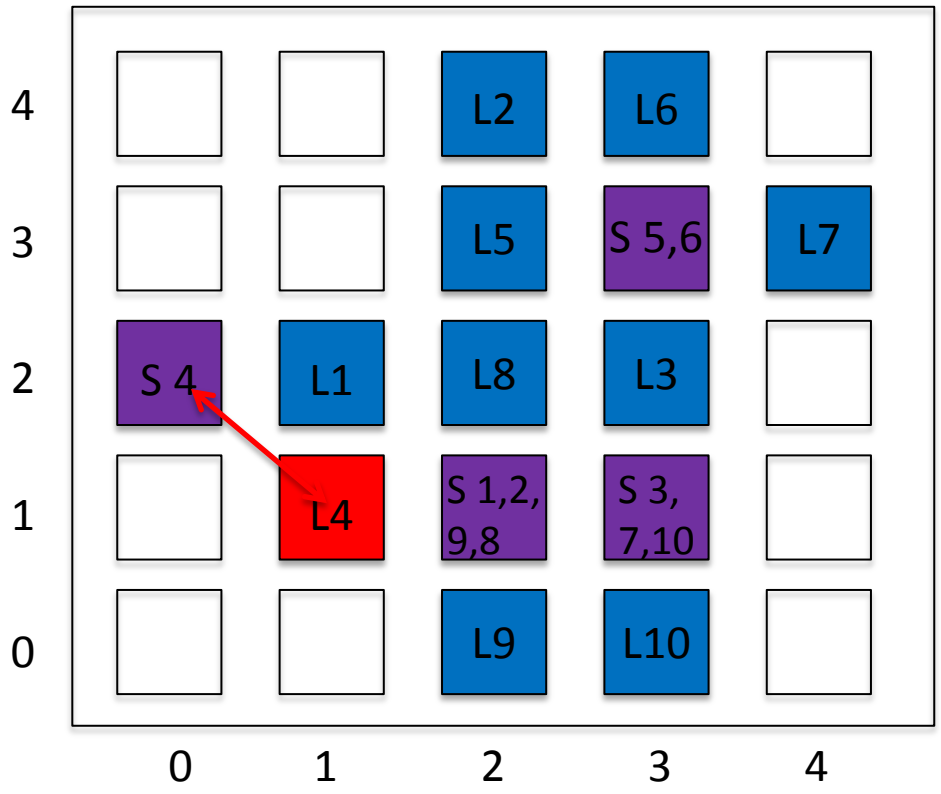


BYU
BRIGHAM YOUNG UNIVERSITY

CISCO

Shadow Placement

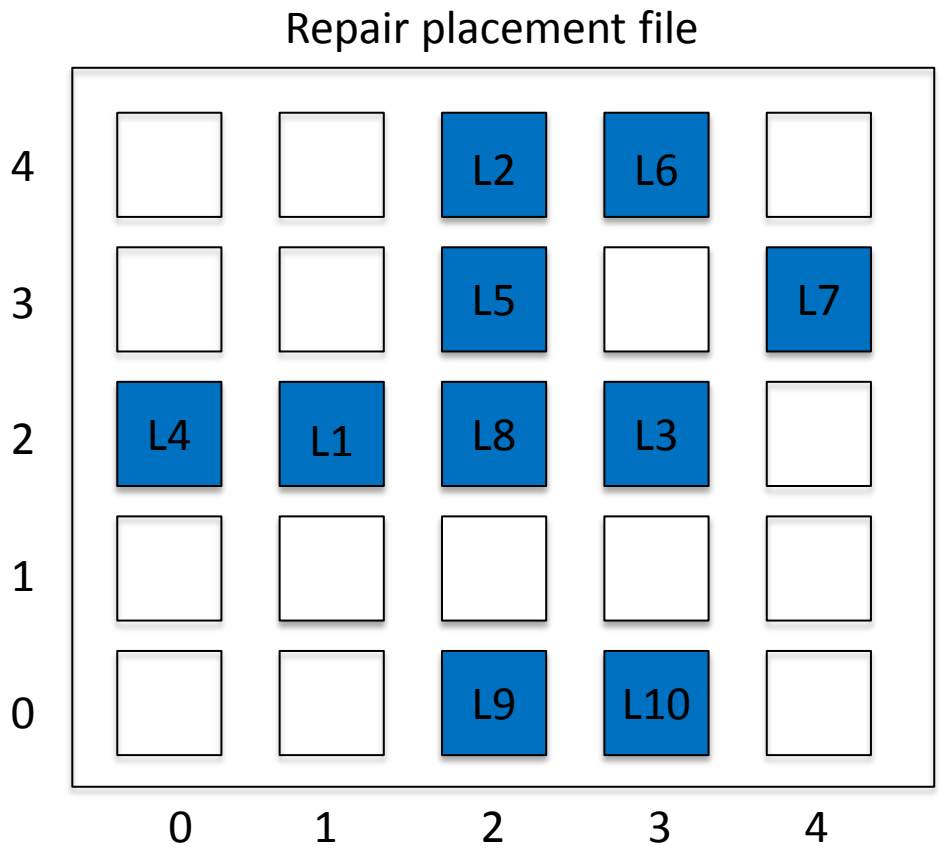
Choose site to repair and swap with shadow



BYU
BRIGHAM YOUNG UNIVERSITY

CISCO

Shadow Placement



BYU
BRIGHAM YOUNG
UNIVERSITY

CISCO

Shadow Placement Results

- Completes in a *single* iteration
- Time does not increase with tighter constraint
- Disadvantage is lower quality of some repairs

Number of shadow sites

	Constraint					
	None	A	B	C	D	E
top	29	23	16	N/A	N/A	N/A
test0	218	122	69	31	19	N/A
mult18	482	167	110	52	26	N/A
crazy	884	878	646	357	175	55
multxor	4094	1773	826	431	282	101
Time	2.12x	3.43x	3.01x	3.1x	2.48x	3.03x

Cost = 1.41x cost of baseline placer

BYU
BRIGHAM YOUNG
UNIVERSITY

CISCO

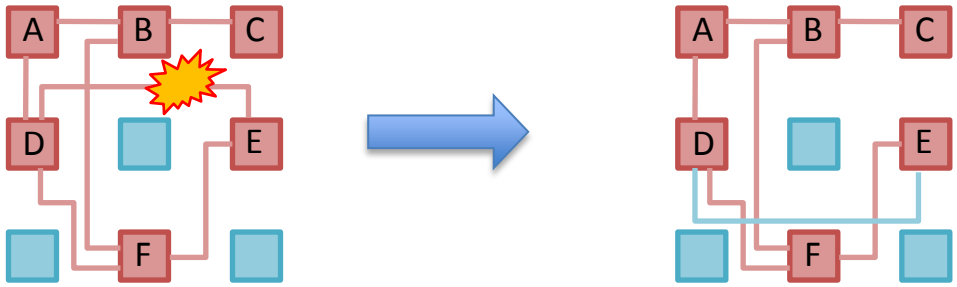
Conclusions

- Multiple placement configurations can be created before deployment for in-field repair
  - Repairs readily available
  - Requires more compute time
- Three algorithms demonstrate a trade-off between run-time and circuit quality

Technique	Run Time	Cost
Naive Repair	210×	1.02
Cost Repair	13.9×	1.05
Shadow Repair	2.9×	1.41

Future Work

- Interconnect repair
  - Repair routing algorithms
  - Integrated routing/placement repair
- Timing driven repair placement and routing



Questions

?