ERROR DETECTION-BASED FAULT-TOLERANCE FOR SPACEBORNE DIGITAL CIRCUITS

Gökçe Aydos

IN THIS TALK

FPGAS AND SPACE

ERROR DETECTION-BASED FAULT-TOLERANCE

COMPARISON WITH LTMR

RELATED WORK AND SUMMARY

FPGAS AND SPACE

RADIATION IN SPACE

- **due to solar wind and cosmic rays**
- **magnetosphere protects us from extraterrestrial radiation**

EXAMPLES OF RADIATION EFFECTS

- short circuits in transistors
- more delay on circuit nets
  due to cumulative dose
- bitflips in circuit flipflops

EFFECTS OF BITFLIPS IN FPGAS

- **configuration memory**
- **application memory (e.g., RAM, flipflops)**

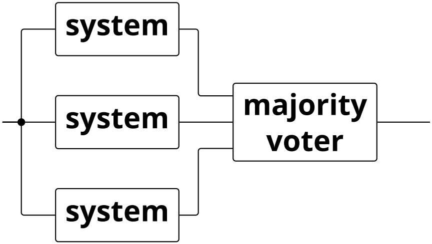
BITFLIPS IN SPACEBORNE FPGAS: AN EXAMPLE

- **one-year mission in space**
- **1.5 million km away between sun and earth**
- **5000 flipflops**
- **8 Kib BlockRAM**

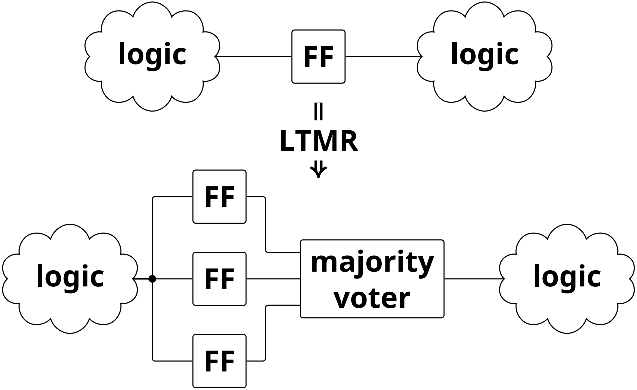
device	conf. mem.	RAM	flipflops
Virtex-4 QV	$\sim$ 300k	$\sim$ 4k	$\sim$ 2k
RT ProASIC3	0	62	4
ATF280	0	0	0

COMMON FAULT-TOLERANCE APPROACH: TRIPLICATION

triple modular redundancy (TMR)

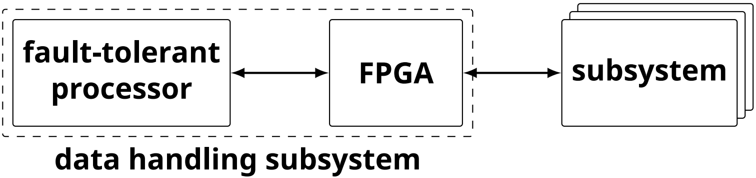


HARDENING AGAINST BITFLIPS IN FLIPFLOPS



ERROR DETECTION-BASED FAULT-TOLERANCE

CASE:
DATA HANDLING ARCHITECTURE

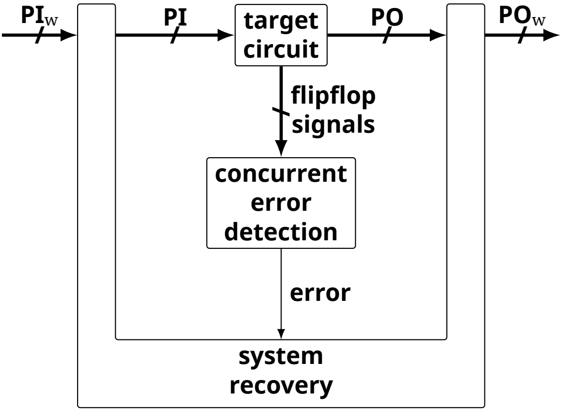


- **circuits on the FPGA are often hardened by triplication of flipflops**
- ***is error detection-based fault-tolerance a good alternative?***

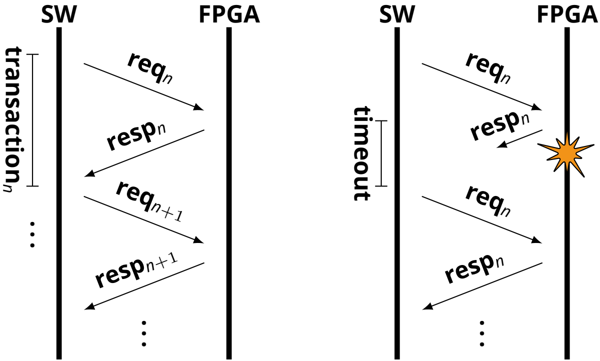
ERROR DETECTION-BASED FAULT-TOLERANCE

- **only error detection on hardware**
- **hardware recovery
using isolation and reset**
- **transaction-based processing**

EDFT APPLIED ON HARDWARE

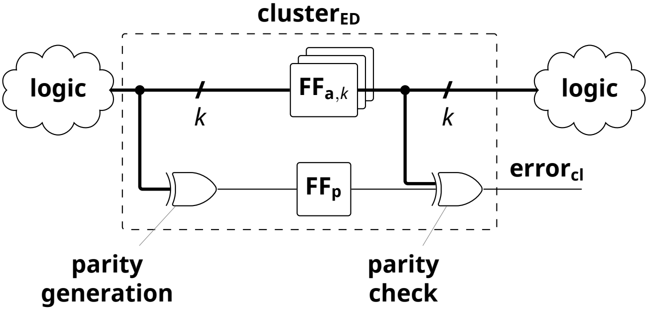


RECOVERY BY RECOMPUTATION

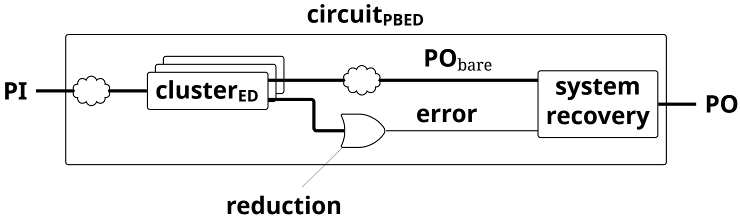


PARITY-BASED ERROR DETECTION

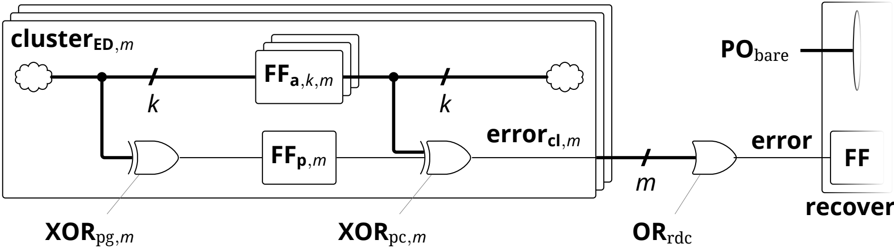
ERROR DETECTION CLUSTER



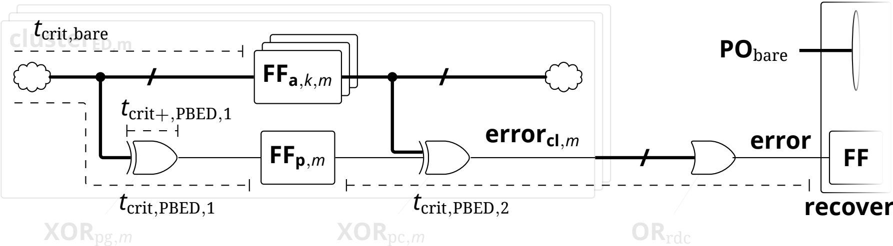
REDUCTION OF CLUSTER ERROR SIGNALS



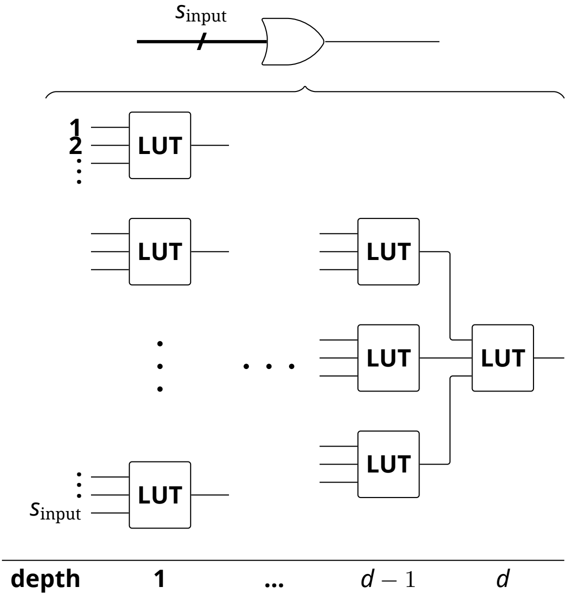
ERROR DETECTION CLUSTER + REDUCTION



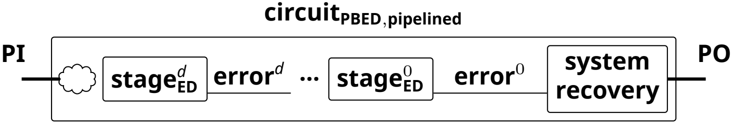
CRITICAL PATHS



LOGICAL OR AS LUT TREE

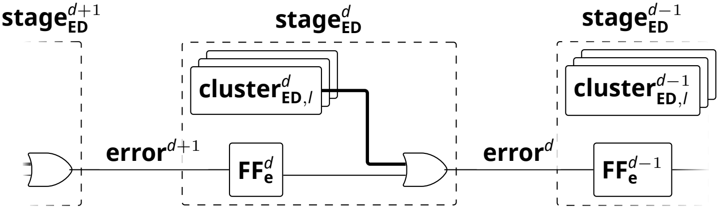


PIPELINED ERROR DETECTION



$$\text{sequential-distance}(\textbf{error}^d, \textbf{PO}) = d$$

PIPELINED ERROR DETECTION II



Data: technology-level netlist, placing try count,
cluster size, partitioning try count
Result: direct PBED applied technology-level netlist

Data: **technology-level netlist, placing try count,**
cluster size, partitioning try count
Result: **direct PBED applied technology-level netlist**
for $t = 1$ **to placing try count do**
    placer seed $= t$**;**
    place & route the netlist;
end

Data: **technology-level netlist, placing try count, cluster size, partitioning try count**
Result: **direct PBED applied technology-level netlist**
for $t = 1$ **to placing try count do**
  placer seed $= t$**;**
  place & route the netlist;
end
pick the netlist with the shortest critical path;
extract FF coordinates from this netlist;

Data: **technology-level netlist, placing try count,**
cluster size, partitioning try count

Result: **direct PBED applied technology-level netlist**

for $t = 1$ **to placing try count do**

  placer seed $= t$**;**

  place & route the netlist;

end

pick the netlist with the shortest critical path;

extract FF coordinates from this netlist;

foreach ***FF*** **do**

  if ***has enable input || has negated output*** **then**

    eliminate enable input and negated output;

  end

Data: technology-level netlist, placing try count, cluster size, partitioning try count
Result: direct PBED applied technology-level netlist

```
for t = 1 to placing try count do
    placer seed = t;
    place & route the netlist;
end
pick the netlist with the shortest critical path;
extract FF coordinates from this netlist;
foreach FF do
    if has enable input || has negated output then
        eliminate enable input and negated output;
    end
    categorize according to clock- and reset-signal;
end
```

if *location-aware partitioning* **then**

foreach *FF category* **do**

if *location-aware partitioning* **then**
    foreach *FF category* **do**
        ...;
        for $i = 1$ **to partitioning try count do**
            while *there are unclustered FFs* **do**
                master $=$ **pick a random FF;**
                neighbors $=$ **pick the nearest** $s_{\mathrm{cl}} - 2$ **FFs;**
                new cluster $= \{$*master*$,$ *neighbors*$\}$**;**
                total dist$.$ *for this try* $+=$
                  *distances from the master to each neighbor***;**
            end

```
if location-aware partitioning then
    foreach FF category do
        ...;
        for i = 1 to partitioning try count do
            while there are unclustered FFs do
                master = pick a random FF;
                neighbors = pick the nearest s_cl − 2 FFs;
                new cluster = {master, neighbors};
                total dist. for this try += 
                  distances from the master to each neighbor;
            end
            if total dist. for this try < min. total dist. then
                mark this partitioning;
            end
        end
```

```
if location-aware partitioning then
    foreach FF category do
        ...;
        for i = 1 to partitioning try count do
            while there are unclustered FFs do
                master = pick a random FF;
                neighbors = pick the nearest s_cl − 2 FFs;
                new cluster = {master, neighbors};
                total dist. for this try += 
                  distances from the master to each neighbor;
            end
            if total dist. for this try < min. total dist. then
                mark this partitioning;
            end
        end
    end
else // random partitioning
    ...
end
add parity-generation and -check circuitry;
```

Result: pipelined PBED applied netlist

Result: pipelined PBED applied netlist

⋮

foreach *primary output (PO)* **do**

  build a FF dataflow graph with this PO as sink vertex;

  annotate the FFs with sequential distance to this PO;

end

Result: pipelined PBED applied netlist

⋮

foreach *primary output (PO)* **do**

    build a FF dataflow graph with this PO as sink vertex;
    annotate the FFs with sequential distance to this PO;

end

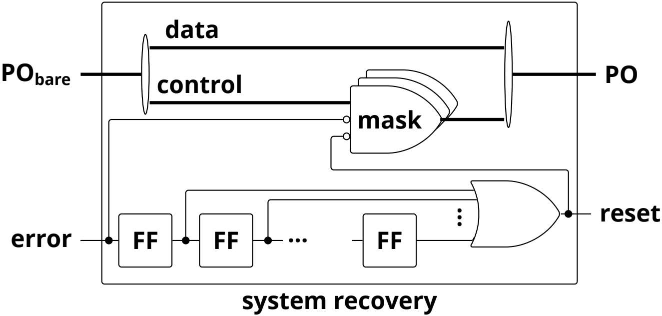
foreach *FF* **do**

    ⋮
    determine *min. sequential distance to all POs*;
    categorize according to … and *min. sequential distance to all POs*;

end

add parity-generation and -check circuitry;

RECOVERY EXAMPLE



COMPARISON WITH LTMR

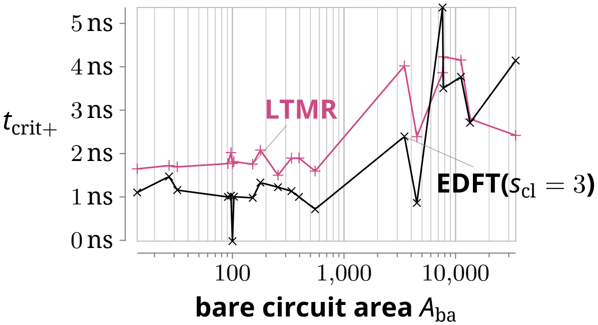
ERROR DETECTION-BASED FAULT-TOLERANCE VS. TMR

- hardware overhead
  - area
  - timing
- processing time overhead
- software overhead

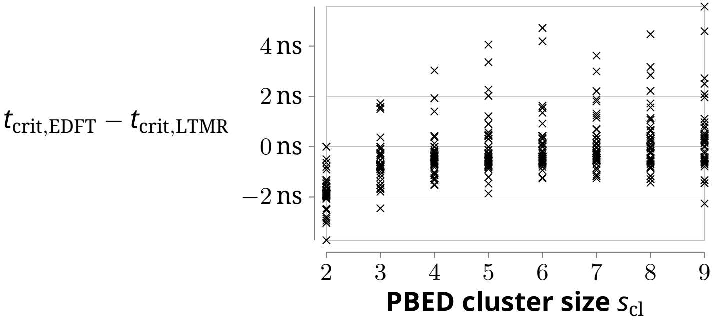
EXPERIMENTAL COMPARISON

- I99T benchmark circuits
- synthesis settings
- ProASIC3 FPGA as target architecture

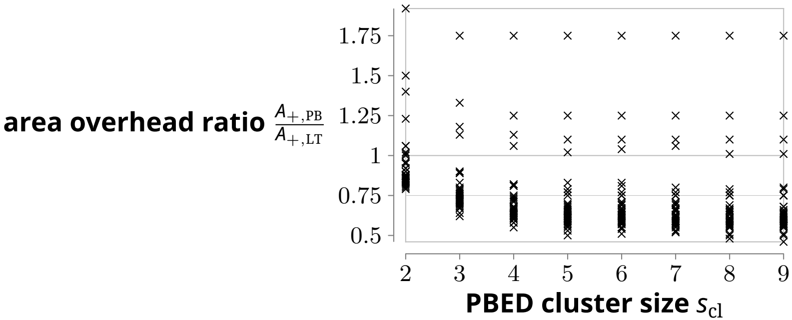
EDFT VS LTMR - CRITICAL PATH



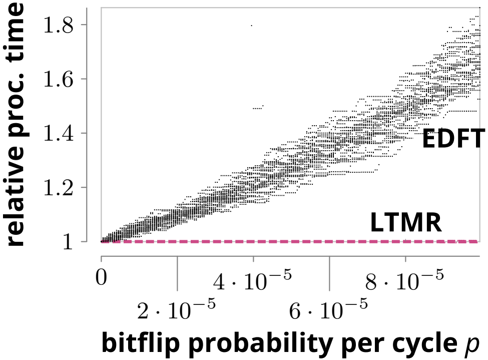
EDFT VS LTMR - CRITICAL PATH - CLUSTER SIZE



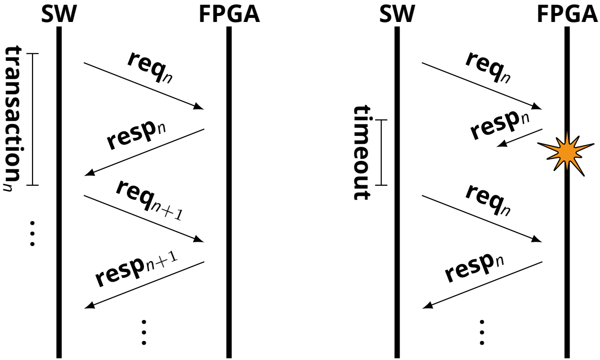
EDFT VS LTMR - AREA OVERHEAD RATIO - CLUSTER SIZE



EDFT VS LTMR - PROCESSING TIME



EDFT VS LTMR - SOFTWARE OVERHEAD



RELATED WORK

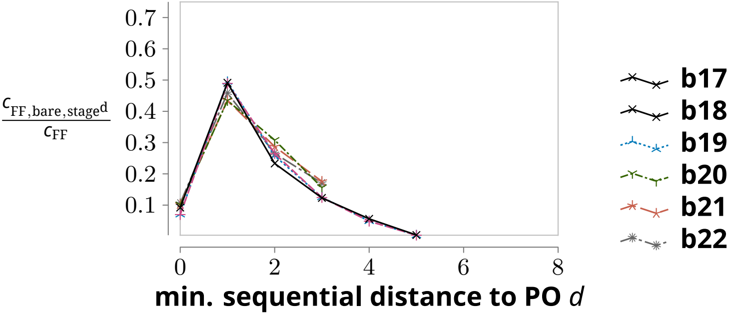
RELATED WORK

- cross layer end-to-end fault-tolerance solution
- parity-based error detection with recomputation on a known spaceborne FPGA
- on application level SW-only techniques are not sufficient
- cross-layer techniques achieve better results

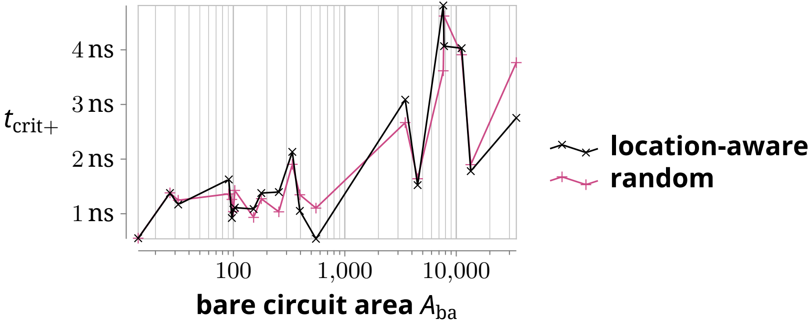
SUMMARY

BACKUP SLIDES

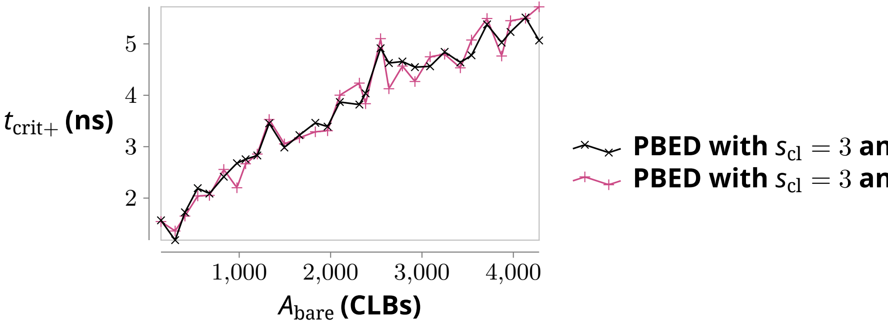
SEQUENTIAL DISTANCE DISTRIBUTION



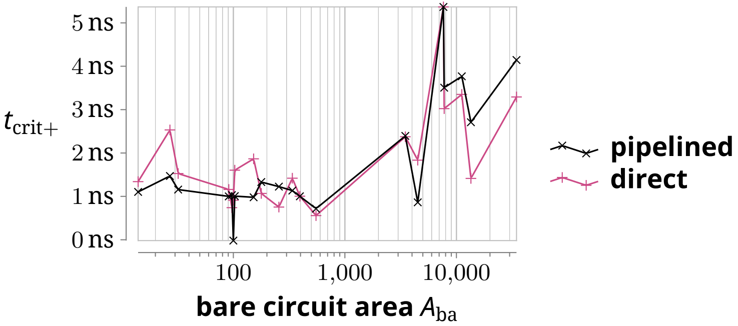
LOCATION-AWARE VS RANDOM PARTITIONING I



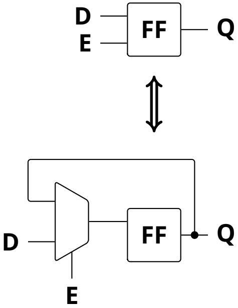
LOCATION-AWARE VS RANDOM PARTITIONING II



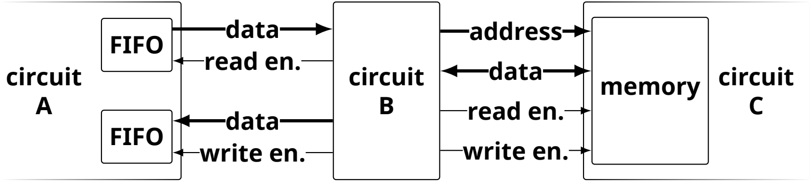
PIPELINED VS DIRECT PBED - CRITICAL PATH



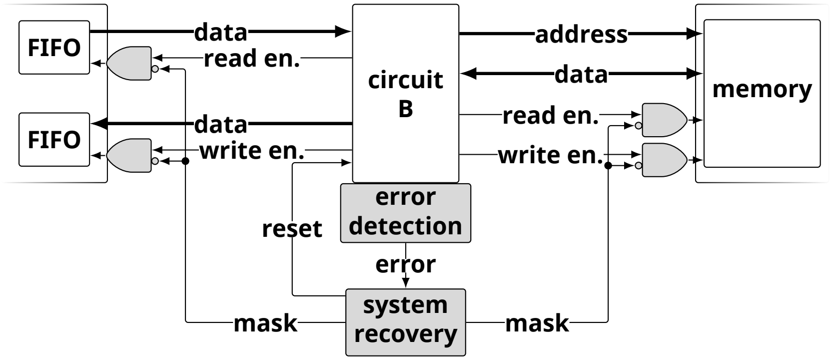
ENABLE FLIPFLOP ELIMINATION



CONTROL SIGNAL MASKING I

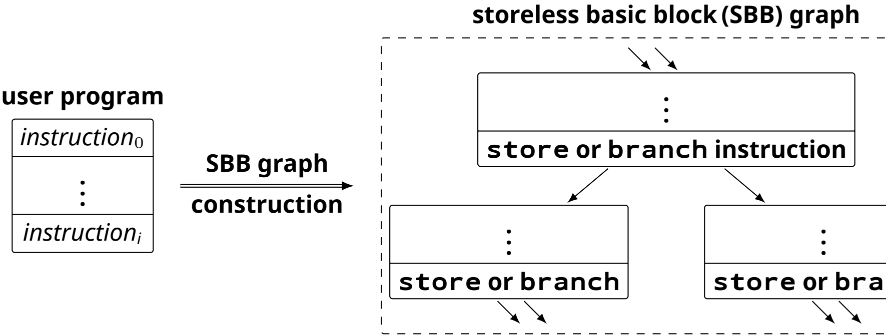


CONTROL SIGNAL MASKING II

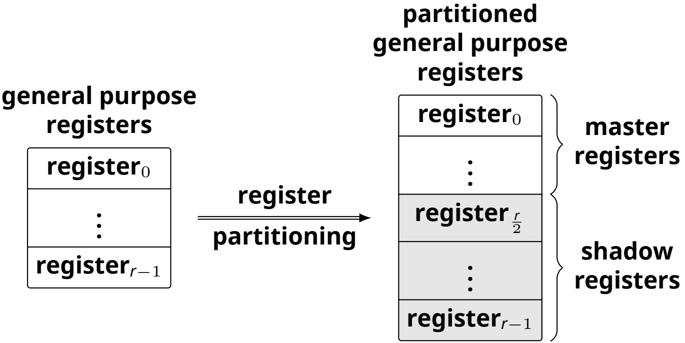


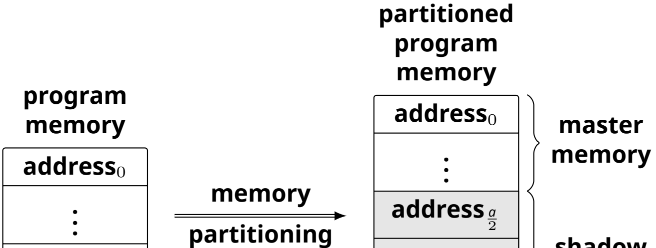
Related Work

STORELESS BASIC BLOCK



REGISTER AND MEMORY PARTITIONING





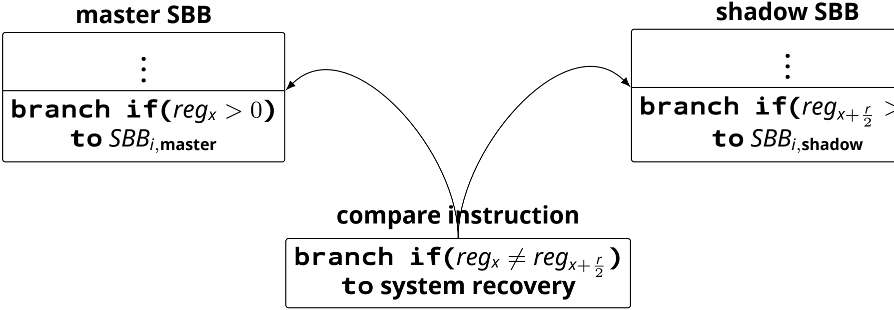
INSTRUCTION DUPLICATION

master SBB
$\vdots$
$\textbf{reg}_z \leftarrow \textbf{mem}[m]$
$\textbf{reg}_x \leftarrow \textbf{reg}_y + \textbf{reg}_z$
$\vdots$

$$\xrightarrow[\textbf{duplication}]{\textbf{instruction}}$$

shadow SBB
$\vdots$
$\textbf{reg}_{z+\frac{r}{2}} \leftarrow \textbf{mem}[m + \frac{a}{2}]$
$\textbf{reg}_{x+\frac{r}{2}} \leftarrow \textbf{reg}_{y+\frac{r}{2}} + \textbf{reg}_{z+\frac{r}{2}}$
$\vdots$

COMPARISON BEFORE EACH BRANCH



EDDI

- ED coverage .98-.99 vs .54-.93 unhardened
- but fault inj. on FF-level results in .86
- motivation: superscalar architectures
- processing overhead .45-1.14 on a 4 inst. per cycle arch.
- can also be implemented on source code level ...

VARIABLE DUPLICATION

user program

```
int a, b;
:
a = b+5;
:
```

hardened program

```
int a, b,
  a_dupl, b_dupl;
:
a = b+5;
a_dupl = b_dupl+5;

if (a != a_dupl)
  recovery();
:
```

BASIC BLOCK SIGNATURES

user program

basic block$_x$

hardened program

move *signature*$_x$ **to** *register*

basic block$_x$

branch if (*signature*$_x \neq$ *register***) to** *recovery*

INVERTED BRANCHES

user program

	$\vdots$
$a+0$	**branch if (***condition***) to** x
$a+1$	$instruction_{a+1}$
$a+2$	$instruction_{a+2}$
	$\vdots$
$x+0$	$instruction_x$
	$\vdots$

hardened program

$a+0$	$\vdots$
$a+1$	**branch if (***condition***) to** $x-1$
$a+2$	**branch if (***condition***) to** *recovery*
$a+3$	$instruction_{a+1}$
	$instruction_{a+2}$
	$\vdots$
$x-1$	**branch if (**$\overline{condition}$**) to** *recovery*
$x+0$	$instruction_x$
	$\vdots$

Performance

- fault inj. on seq. and comb. of a processor
  - 0.77 to 0.84 for EDDI
  - 0.04 to 0.09 for basic blocks signatures
  - 0.01 for inverted branches
- undetected errors due to jumps from a BB to the same BB
- full error coverage unlikely [Aza+11]

Cross-layer FT techniques

[Che+16]

- processors in terrestrial environments
- a combination of low- and high-level techn. proposed
- fault inj. on synthesized and layouted circuits
- silent data corruption (SDC): SW terminates, but error in output
- detected but uncorrected error (DUE): SW does not terminate, restart req.
- ~~error coverage~~
- $impr = \frac{\sum erroneous\ outcomes\ unhardened}{\sum erroneous\ outcomes\ hardened}$
- because not all bitflips lead to a failure, e.g., 40% do not lead to a failure, e.g., branch prediction

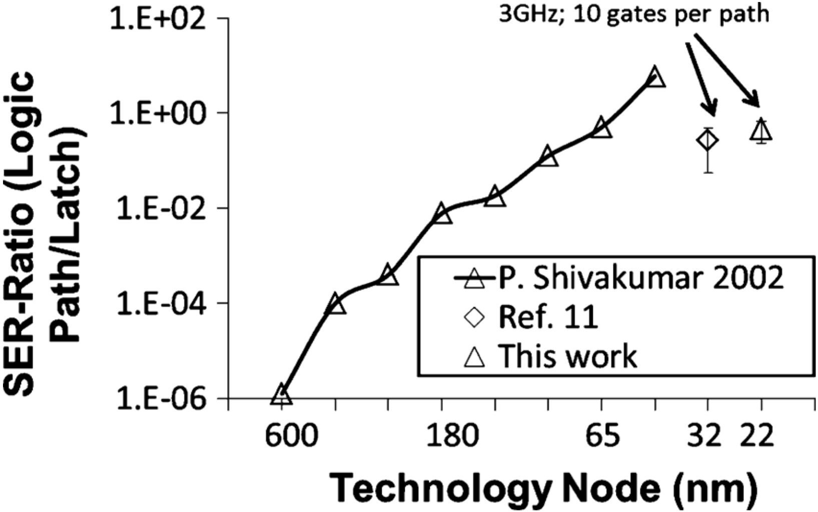
SEU vs SET

DIRECT BITFLIPS VS TRANSIENTS ON COMBINATORICS

- **electrical pulses on combinational nets (SET)**
- **direct bitflip in a sequential element (SEU)**

- **ProASIC3: bitflips mainly caused by SEUs.**
- **32 nm:** $\frac{\textit{error rate}_{\mathrm{SET}}}{\textit{error rate}_{\mathrm{SEU}}} < 30\%$
- **22 nm: very small increase**

SOFT ERROR RATE COMPARISON IN 22 NM NODE



[Sei+12]

Microsemi RTG4

- 65 nm
- TMR'ed flipflops
- SET filter in flipflops
- error rate 1000x better than SmartFusion2 FPGA

Cross section

- **SEU cross section** $= \frac{\textbf{error count}}{\textbf{fluence}}$
- **Fluence [**$\frac{\textbf{particle}}{\textbf{cm}^2}$**]**
- **calculated for different particle spectrums (linear energy transfer (LET))**

FAULT TOLERANCE CLASSIFICATION

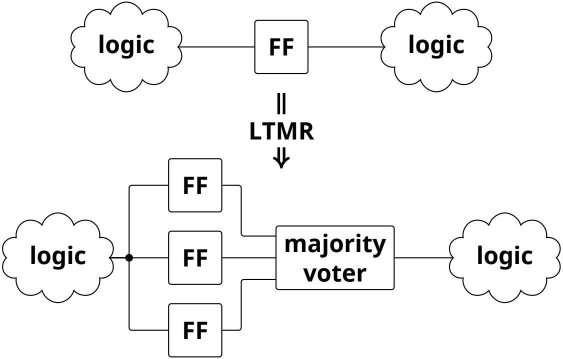
- error detection
  - concurrent detection
  - preemptive detection
- recovery
  - error handling
    - compensation
    - rollback
    - rollforward
  - fault handling
    - diagnosis
    - isolation
    - reconfiguration
    - reinitialization

FT TECHNIQUES AGAINST BITFLIPS

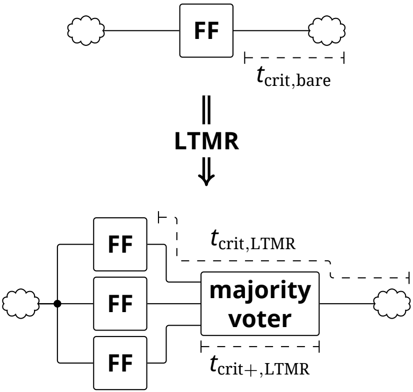
- fabrication process level
- chip layout level
- logic level
- architecture level
- software level
- algorithm level

TMR Techniques

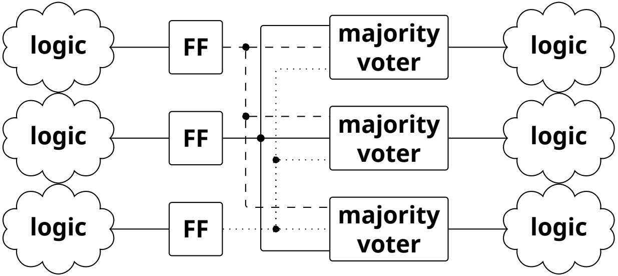
LOCAL TMR



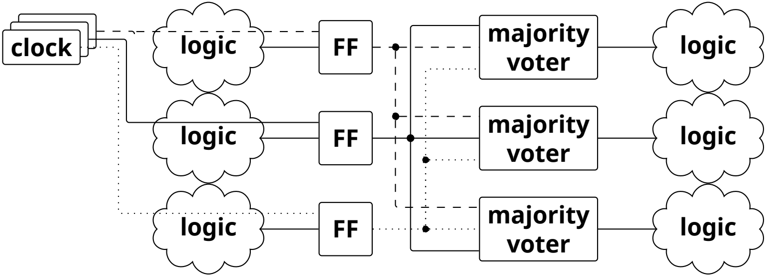
LOCAL TMR - CRITICAL PATH



DISTRIBUTED TMR



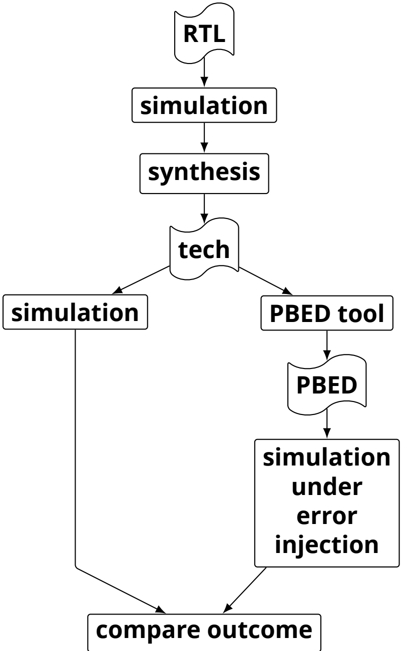
GLOBAL TMR



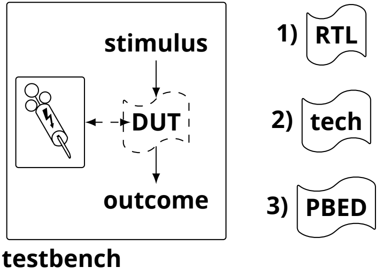
clock
logic
logic
logic
FF
FF
FF
majority voter
majority voter
majority voter
logic
logic
logic

Verification

SIMULATION FLOW

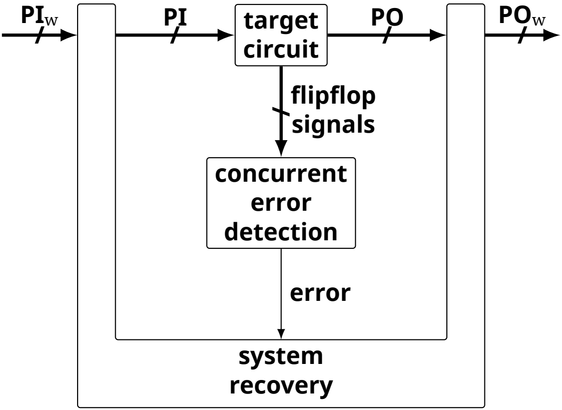


TESTBENCH OVERVIEW

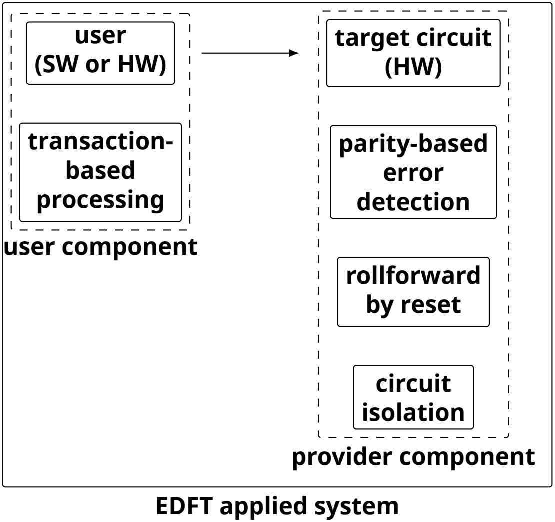


EDFT

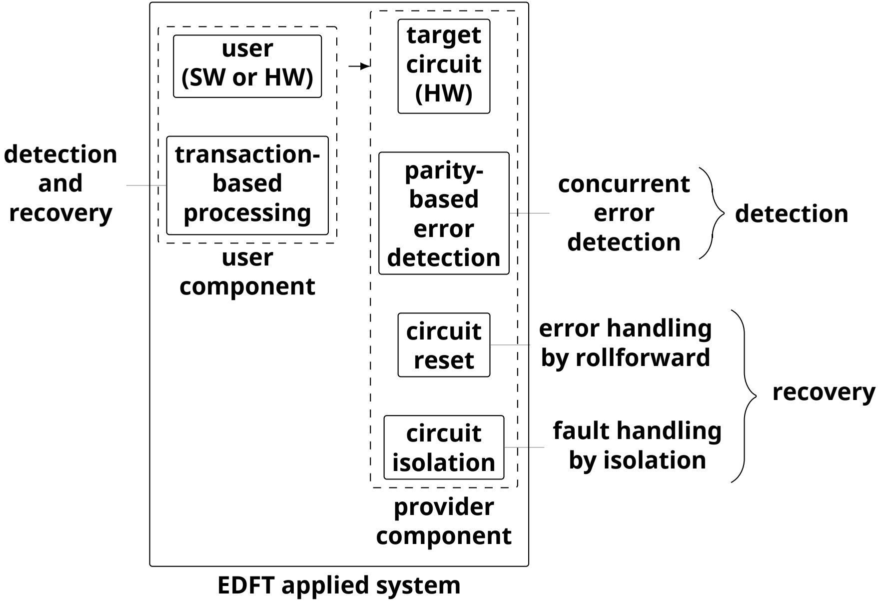
EDFT APPLIED ON HW



EDFT APPLIED ON THE REFERENCE DESIGN A

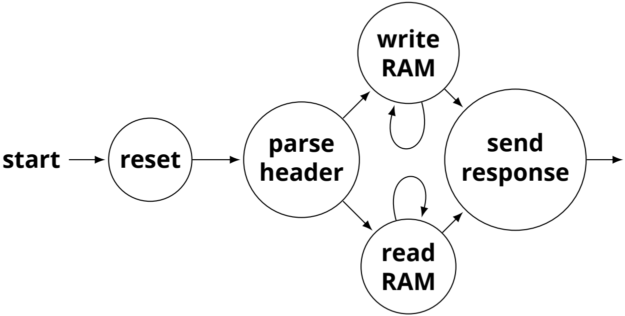


EDFT APPLIED ON THE REFERENCE DESIGN B

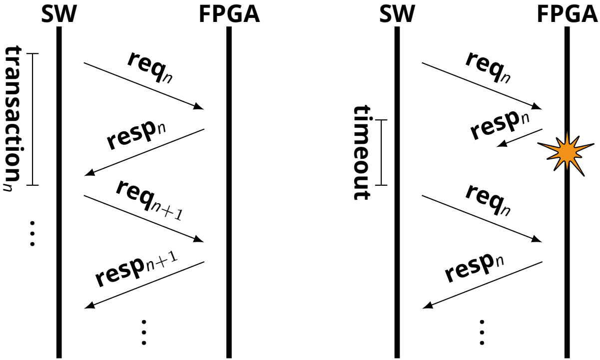


FSM of Circuit B

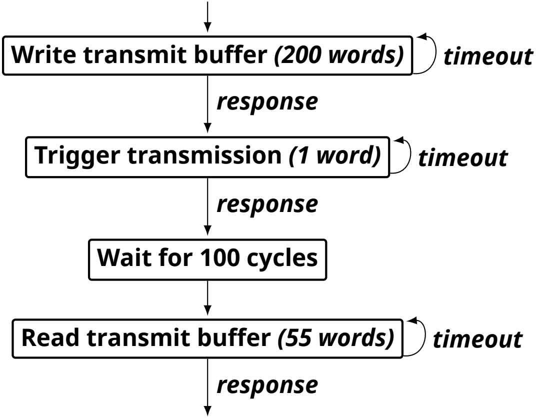
STATE MACHINE OF CIRCUIT B



REFERENCE DESIGN PROTOCOL DIAGRAM

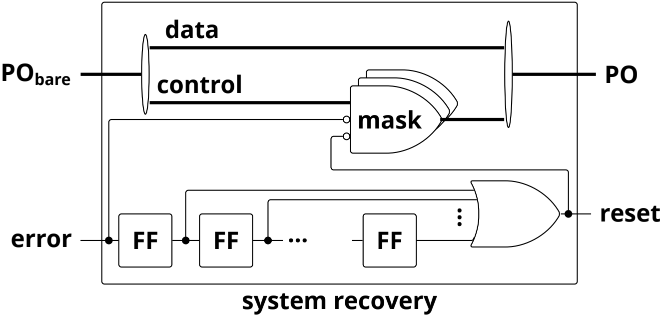


FAULT INJ. TESTBENCH SW FLOWCHART

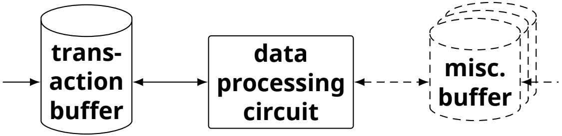


System Recovery

RECOVERY EXAMPLE



PROCESSING MODEL

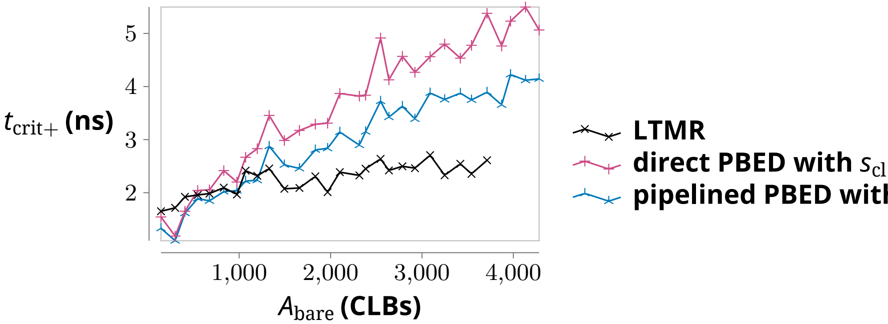


TRANSACTION ON CYCLE LEVEL

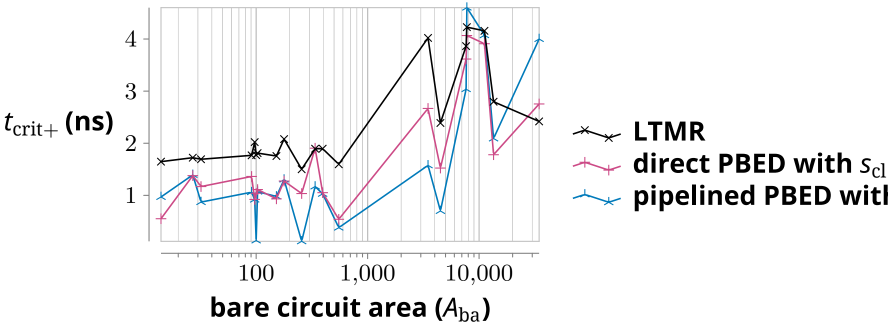
$cycle_{\mathbf{clk}}$	t	$t+1$	$\cdots$	$t+q$	$\cdots$	$t+1+p$
tb. i/	$\mathbf{req}_n(0)$	$\mathbf{req}_n(1)$	$\cdots$	$\mathbf{req}_n(q)$		
tb. /o		$\mathbf{resp}_n(0)$	$\cdots$		$\cdots$	$\mathbf{resp}_n(p)$
mb. i/o		*data*	$\cdots$		$\cdots$	*data*

$\mathbf{transaction}_n$

PP COMPARISON - FSM - CRITICAL PATH OVERHEAD



PP CRITICAL PATH - I99T - VARIABLE CLUSTER SIZE



REFERENCES I

- N. Battezzati, L. Sterpone, and M. Violante, *Reconfigurable Field Programmable Gate Arrays for Mission-Critical Applications*. Springer, 2011. DOI: 10.1007/978-1-4419-7595-9.
- J. R. Azambuja, S. Pagliarini, L. Rosa, and F. L. Kastensmidt, **“Exploring the limitations of software-based techniques in SEE fault coverage,”** *J Electron Test*, vol. 27, no. 4, pp. 541–550, Apr. 2011. DOI: 10.1007/s10836-011-5218-7.
- E. Cheng, P. Bose, S. Mitra, S. Mirkhani, L. G. Szafaryn, C.-Y. Cher, H. Cho, K. Skadron, M. R. Stan, K. Lilja, and J. A. Abraham, ***CLEAR: Cross-layer exploration for architecting resilience - combining hardware and software techniques to tolerate soft errors in processor cores***, version 2, Jun. 23, 2016. arXiv: 1604.03062v2 [cs.AR].

REFERENCES II

- C. Poivey, M. Grandjean, and F. X. Guerre, "Radiation characterization of microsemi proasic3 flash fpga family," in *2011 IEEE Radiation Effects Data Workshop (REDW)*, Jul. 2011, pp. 1–5. DOI: 10.1109/REDW.2010.6062510.
- B. Gill, N. Seifert, and V. Zia, "Comparison of alpha particle and neutron-induced combinational and sequential logic error rates at the 32nm technology node," in *2009 IEEE International Reliability Physics Symposium*, Institute of Electrical and Electronics Engineers (IEEE), Apr. 2009, pp. 199–205. DOI: 10.1109/irps.2009.5173251.

REFERENCES III

N. Seifert, B. Gill, S. Jahinuzzaman, J. Basile, V. Ambrose, Q. Shi, R. Allmon, and A. Bramnik, **"Soft error susceptibilities of 22 nm tri-gate devices,"** *IEEE Transactions on Nuclear Science*, vol. 59, no. 6, pp. 2666–2673, Dec. 2012, ISSN: 0018-9499. DOI: `10.1109/tns.2012.2218128`.