Automatic Time-Redundancy Transformation for Fault-Tolerant Circuits

Dmitry Burlyaev
Pascal Fradet & Alain Girault

02/24/15, Monterey, California, USA
Our Objectives Here

Develop a new *automatic* logic-level transformation for fault-tolerance:

1. technologically independent (*eg*, no clock line control).

2. SET correction with double time redundancy and less HW overhead than in TMR.

3. input/output transparent behavior to SET.

4. formally provable with a proof assistant.
Considered Fault-Models

- Single-Event Upset (SEU)
  - a bit-flip

- Single-Event Transient (SET)
  - a current spike on a wire → SEU(s)
Automated TMR Schemes
Triple-Time Redundancy

\[ \vec{p}_{i_1} \rightarrow \#1 \rightarrow \vec{p}_{o_1} \]

combinatorial circuit
Triple-Time Redundancy

\[ \pi_2 \]

\[ \text{combinatorial circuit} \]

\[ \text{#2} \]

\[ \text{po}_2 \]

\[ \text{D} \quad \text{Q} \]

\[ \text{CLK} \]
Triple-Time Redundancy

\[ \vec{p}_3 \]

#3

\[ \vec{p}_3 \]

combinatorial circuit

\[ \begin{align*}
\text{D} & \quad \text{Q} \\
\text{CLK} \\
\text{D} & \quad \text{Q} \\
\text{CLK} \\
\text{D} & \quad \text{Q} \\
\text{CLK}
\end{align*} \]
Double Time Redundancy Transformation

- micro checkpointing-rollback
- speed-up mode (switching-off time-redundancy)
- input/output buffers (input/output transparency)
- tolerance to at most one SET in 10 clock cycles
Transformation DTR

Input stream upsampling $\times 2$ (original throughput/2)

1) Memory Cell $\leftarrow$ Memory Block
2) Control Block Introduction
3) Input/Output Buffers Insertion
Memory Block: Working Cycle

rollBack

save

si

muxA

muxB

EQ

fail

0

C

1

C

0

1

0
Memory Block: Working Cycle

0 rollBack

0 save

a1

EQ ≠ fail
Memory Block: Working Cycle

0 \text{ rollBack}

1 \text{ save}

a_2

a_1

E
E

1
0
C
1
C
0

EQ
≠

fail
Memory Block: Working Cycle

0 rollback

0 save

b_1

a_2

E

a_1

E

1
0 C

1 C
0

EQ ≠ fail 0

a_1 = a_2
Memory Block: Working Cycle

0 rollBack

1 save

b₂

a₂

b₁

a₂

EQ

fail?

b₁? a₂
Memory Block: Working Cycle

0 rollBack

0 save

C1

b2

a2

b1

b2 = b1

EQ

fail 0

b2 = b1
Memory Block: Working Cycle

0 rollBack

1 save

\[ c_2 \]

\[ E \]

\[ b_2 \]

\[ a_2 \]

\[ 1 \]

\[ 0 \]

\[ c_1 \]

\[ b_2 \]

\[ EQ \]

\[ \neq \]

\[ c_1 \neq b_2 \]

fail?

Cycle: 1
Memory Block: Error-detection

Cycle: 2
Memory Block: Rollback

Cycle: 3
Memory Block: Speed-up

Cycle: 4
Memory Block: Output synchronization

1 rollBack

0 save

e1

d1

c3

d2

c2

Cycle: 5
Memory Block: Output synchronization

1 rollBack

0 save

f1

Cycle: 6
Memory Block: Speed-up OFF

0 rollBack

1 save

f2

f1

c3

e2

E


c2

d2

E


1

1

C

C

0

0

1

1

fail

\[ f_1 \oplus e_1 \]

\[ f_2 \]

Cycle: 7

\[ f_1 \oplus e_1 \neq \text{fail} \]
Memory Block: Working Cycle- no FT

0 rollBack

0 save

g1

Cycle: 8

f2

f1

f2

f1

f2 = f1

fail 0

EQ

f2 = f1
Memory Block: Working Cycle- no FT

0 rollBack

1 save

\[
g_2
\]

Cycle: 9
Memory Block: Working Cycle

Cycle: 10

0 rollBack

0 save

h1

Cycle: 10

fail 1

g1 = g2

E

Q

g2

f2

E

Q

g2

f2

E
Transformation DTR

Input stream upsampling $x2$ (original throughput/2)

1) Memory Cell $\leftarrow$ Memory Block
2) Control Block Introduction
3) Input/Output Buffers Insertion
Control Block

start → norm1

fail \neq 1

save = 1

rollBack = 1

rB = 1
save = 1
rollBack = 1
subst = 1

error

norm1 → norm2

fail = 0

norm2

reco1

rB = 1
rollBack = 1
subst = 1

recov1 → reco2

reco2

reco3

rB = 1
subst = 1

rollBack = 1
Input Buffer

- added to each primary input
- keeps last 2 bits **For** recovery recalculation
- keeps last 2 bits **During** recovery for transparency
Output Buffer

- added to each primary output
- adds a delay to emit correct values during recovery
- fault-tolerant by itself to SET
Experimental Results
area - in terms of Core Cells for Actel ProASIC3 FPGA
Synthesised in Synopsys Synplify Pro’2009
DTR = original x1.39-2.0
TMR = original x3.4-3.9
TMR = DTR x1.9-2.5
Synthesis Results: Small <100 mem. cells

The advantage is smaller
Summary
Automatic circuits transformation for time-redundancy:

can be formally proved:

\[ \forall C : \text{circuit}, \forall i : \text{inputs}, \forall o : \text{outputs}, \quad C \ i \rightarrow o \quad \Rightarrow \quad \neg \exists \text{faulty} \] 

applicable to existing synthesis tools

(Mentor Graphics, Xilinx, Synopsys)

1.9-2.5 smaller than TMR
(with double throughput loss)

stream processing, technologically independent,
Thank you for your attention!

Your Questions/Feedbacks are WELCOMED

dmitry.burlyaev @ inria.fr