Quantifying Robustness by Symbolic Model Checking

S. Baarir C. Braunstein E Encrenaz J-M. Ilié T. Li
I. Mounier D. Poitrenaud S. Younes

HWVW 2010, July 15, 2010

Outline

Motivation

Dependability Analysis

Dependable circuit to transient faults

Soft error (SET or SEU) is and will be even more a major concern of embedded hardware designers.

- Critical applications(space mission ...) submitted to particle strikes or electromagnetic interferences
- Many other applications (video stream, phones ...) submitted to crosstalk coupling and/or high temperature

Early analyses to evaluate the impact of faults

- Improve the confidence of a design
- Early identification $\Rightarrow$ less \$ or € for modifications
  - ➢ Identify the precise locations to be protected
  - ➢ Choose between different architectures of a design

Robustness evaluation

Analysing robustness with respect to soft errors

Huge state-space exploration

- soft error may come for bit-flip or erroneous latched signals
- bit-flip may occurred different location and time
- circuits have hundred of thousands flip-flops

Fault occurrences may cause tons of possible error configurations

Our approach

- Working at RTL level
- Handling time and space multiple faults simultaneously (vs. simulation/injection)
- Relaxing the strict equivalence to a golden model or a specification

Self-stabilization evaluation

After a period of particles strikes, how to insure that the circuit returns to a *safe configuration*?

Analysing the self-healing capabilities of circuits

Concerns of our measures:

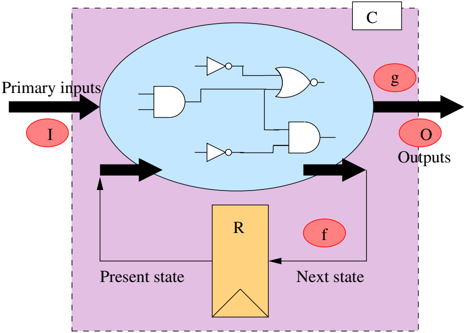
1. Rates of reparation ability
   - ➙ Number of *potentially* and *eventually* repairable states
2. Reparation velocity
   - ➙ Bounds of the reparations sequences

This allows designers to

- Choose part of design to be hardened
- Choose between implementations of the same design

Preliminaries

Circuit



Reachable States and Sequences

- $\mathbf{r} \in 2^R$: a state of C
- $\mathbf{R}_0$: the set of initial state:
- $\mathbf{i}_1.\mathbf{i}_2 \ldots \mathbf{i}_{n-1}$: an input sequence
- $f(\mathbf{i}_1.\mathbf{i}_2 \ldots \mathbf{i}_{n-1}, \mathbf{r})$: a state sequence
- $g(\mathbf{r}, \mathbf{i}_1.\mathbf{i}_2 \ldots \mathbf{i}_{n-1})$: an output sequence
- $reach(C)$: the set of reachable states of C from $\mathbf{R}_0$

Our robustness proposition

Fault Model

Fault Model

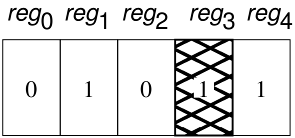
Type of faults

- Errors appear as *bit-flips* on register elements.
- There exists a set of *protected* register elements $P \subseteq R$ (this set may be empty).

Fault occurrences

- Occurrence of Multiple Faults – Multiple Units, except in protected registers.
- Several faults may occur at different time instants.

Circuit functioning with fault occurrences

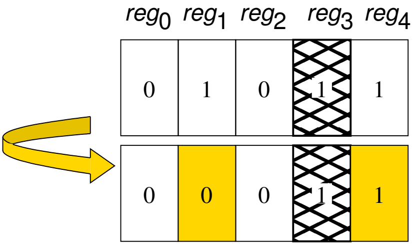


Reachability set with fault occurrences

$Error(C, P)$, is the smallest subset of 2^R satisfying:

- $\mathbf{R}_o \subseteq Error(C, P)$
- $\mathbf{r} \in Error(C, P) \Rightarrow \{\mathbf{r}' \in 2^R \mid \forall p \in P, \mathbf{r}'[p] = \mathbf{r}[p]\} \subseteq Error(C, P)$

Circuit functioning with fault occurrences

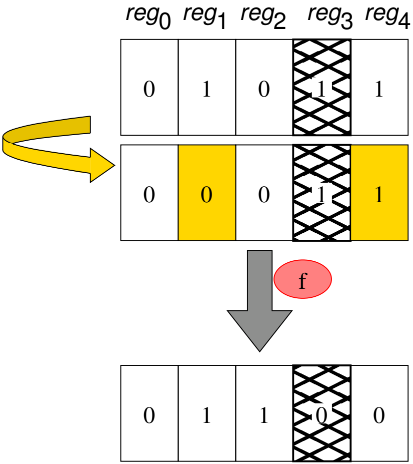


Reachability set with fault occurrences

$Error(C, P)$, is the smallest subset of 2^R satisfying:

- $\mathbf{R}_o \subseteq Error(C, P)$
- $\mathbf{r} \in Error(C, P) \Rightarrow \{\mathbf{r}' \in 2^R \mid \forall p \in P, \mathbf{r}'[p] = \mathbf{r}[p]\} \subseteq Error(C, P)$
- $\mathbf{r} \in Error(C, P) \Rightarrow \{\mathbf{r}' \in 2^R \mid \exists\, \mathbf{i} \in 2^I, \mathbf{r}' = f(\mathbf{i}, \mathbf{r})\} \subseteq Error(C, P)$

Circuit functioning with fault occurrences



Reachability set with fault occurrences

$Error(C, P)$, is the smallest subset of 2^R satisfying:

- $\mathbf{R}_o \subseteq Error(C, P)$
- $\mathbf{r} \in Error(C, P) \Rightarrow \{\mathbf{r}' \in 2^R \mid \forall p \in P, \mathbf{r}'[p] = \mathbf{r}[p]\} \subseteq Error(C, P)$
- $\mathbf{r} \in Error(C, P) \Rightarrow \{\mathbf{r}' \in 2^R \mid \exists\, \mathbf{i} \in 2^I, \mathbf{r}' = f(\mathbf{i}, \mathbf{r})\} \subseteq Error(C, P)$

Each state in $Error(C, P)$ is called an error state.

Repairing model

Repairing sequences
Introduction

Requirements

When faults do not occur anymore, we want to characterize the set of error state that are "repairable":

- Reach a state considered as "correct"
- The path between the error state and the correct state is "constrained"

Definition (Repairing sequence)

A repairing sequence is a sequence from an error state up to a *correct state*

- when faults do not occur anymore,
- when the sequence respects a **repairing pattern.**

Repairing Sequences
Repairing Pattern

Repairing path

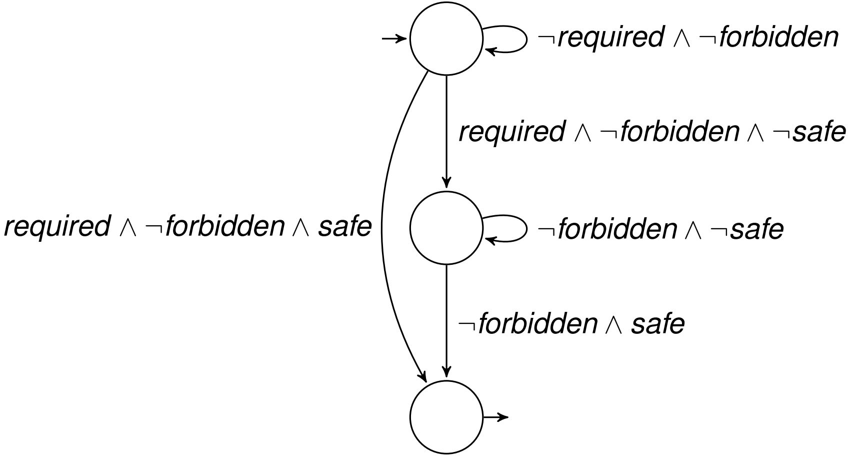
The way to go from an error state to a "correct" configuration (*safe*) may be constrained.

- Some configuration may be avoided (*forbidden*)
- Some configuration may be mandatory (*required*)

Repairing automaton

- Usual way to express constraints on paths: an automaton.
- A *Repairing automaton* for C is defined by $\langle S, T, S_0, F \rangle$ where :
  - S a finite set of states.
  - $T \subseteq S \times 2^R \times S$ a finite set of labeled transitions.
  - S_0 a finite set of initial states.
  - F a finite set of accepting states.

Repairing automaton example 1/2



Repairing automaton example 2/2

How to express set of states ?

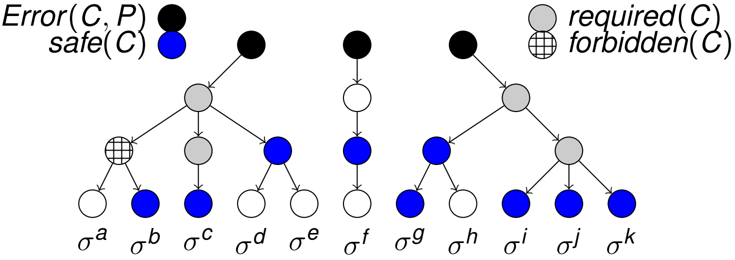
safe(C), *required*(C),*forbidden*$(C)\dots$ can be easily characterized as CTL properties:

- $\phi = reach(C)$: the whole set of reachable states.
- $\phi = AG(AF R_0)$: set of states returning unavoidably into the initial state.
- $\phi = \neg(r_1 \vee r_2)$: a given configuration of registers.

Quantification

1. Motivation
2. Preliminaries
3. Our robustness proposition
   - Fault Model
   - Repairing model
   - Quantification
4. Experiments
5. Conclusion and ongoing work

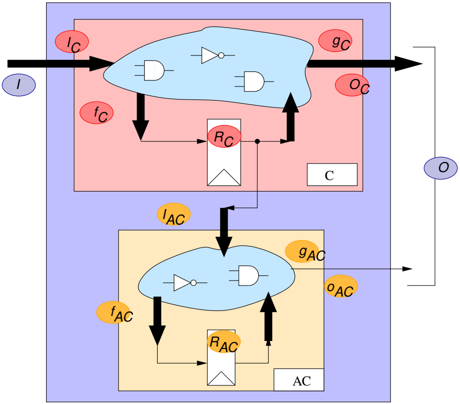
Robustness
State-based quantification



To quantify the circuit's robustness, we compute :

- The number of Error states.
- Potentiality: The number of Error states from which at least one infinite fair sequence is a repairing sequence.
- Eventuality: The number of Error states from which all infinite fair sequences are repairing sequences.

Computing potentially and eventually reparable states



Computation

Set of repaired configuration :
$Repaired = \{(\mathbf{r_C}, \mathbf{r_{AC}}) \in 2^{R_C} \times 2^{R_{AC}} \mid g_{AC}(\mathbf{r_{AC}}) = 1\}$

$$\nu_{pot} = \frac{|\mathrm{EF}_{fair}\ Repaired \cap \mathbf{R}_0|}{|\mathbf{R}_0|}$$

$$\nu_{ev} = \frac{|\mathrm{AF}_{fair}\ Repaired \cap \mathbf{R}_0|}{|\mathbf{R}_0|}$$

Robustness

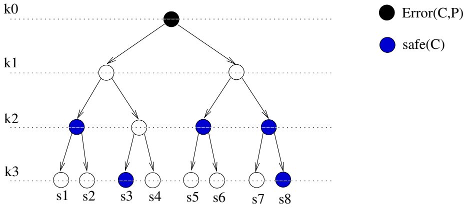
Sequence-based quantification

The velocity of the circuits is characterized by:

- Minimal and maximal length of repairing sequences
- The number of repairing sequences for each length between the bounds

Hypothesis

- We focus on the first repairing state along a repairing sequence.
- The environment reacts as soon as possible.



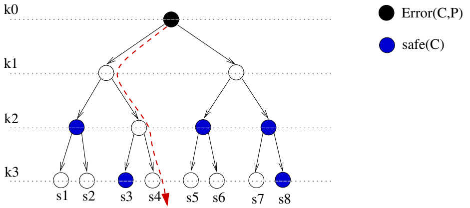
Robustness
Sequence-based quantification

The velocity of the circuits is characterized by:

- Minimal and maximal length of repairing sequences
- The number of repairing sequences for each length between the bounds

Hypothesis

- We focus on the first repairing state along a repairing sequence.
- The environment reacts as soon as possible.



Computing length

```
Input C: an instrumented circuit;
Output t: array of Integer;
  k=0;
 While SAT(WithoutLoop(C, k)){
   t[k] = #SAT(ElementaryRep(C, k));
   k=k+1;
  }
  Return(t);
```

Computation

We compute the elementary repairing sequences:

$$[\mathit{WithoutLoop}(C,k)] \wedge [\mathbf{r}_k \in \mathit{Repaired}] \wedge \left[\bigwedge_{0\leq j<k} \mathbf{r}_j \notin \mathit{Repaired}\right]$$

- Bounds are computed by applying SAT solver iteratively.
- Number of sequences is translated in a #SAT problem.

Experiments

Tool: extension of VIS

- What we have, the VIS model checker:
  - RTL inputs: Verilog
  - Symbolic structure: BDD
  - Temporal logics: CTL, LTL
  - Sat techniques.
- What we need:
  - Counting Error states,
  - Counting Reparable states (Error states satisfying CTL formulae)
  - Counting Elementary repairing sequences (sequences satisfying LTL formulae) $\Rightarrow$ $\#$Sat problem.

Case study : different versions of a *gcd* circuit

- State-based quantification:

C	$\lvert reach(C)\rvert$	$\lvert Error(C,P)\rvert$	ν_{pot}	ν_{ev}	Time
`gcd`	137929	2097152	100%	21%	0.36
$\texttt{gcd}_{fair}$				100%	2
$\texttt{gcd-v1}_{fair}$			98%	98%	0.40
$\texttt{gcd-v2}_{fair}$	304528	$5.368709e^{08}$	100%	100%	18

- Sequence-based quantification:

C	Time	Cycles						
		0-2	3	4	5	6	7	8
`gcd`	211	$5e^{-10}$	$1{,}47e^{-7}$	$9{,}85e^{-5}$	0,05	0,94	-	-
`gcd-v2`	1595	$3{,}93e^{-15}$	$8{,}70e^{-13}$	$4{,}28e^{-10}$	$1{,}54e^{-7}$	$1{,}22e^{-5}$	0,002	0,99

Conclusion and ongoing work

Conclusion

A new Framework

- Multiple transient faults by symbolic management
- Early in a design flow
- First implementation within a classical model checker (VIS)

New metrics

- Self-healing capabilities criteria
- Metrics to help choosing more robust design
- Metrics to determine the minimal set of protected register

Ongoing work

More elaborate fault model

Spatio-temporal windows

- Limit the number of fault occurrences
- Bounded the time of fault occurrences

More elaborate reparation

- Environmental context
- Circuit execution
- Time constraints