ACL2 for the Verification of Fault-Tolerance Properties: First Results

Laurence PIERRE, Renaud CLAVEL, Régis LEVEUGLE

TIMA Laboratory
Grenoble, France
Introduction

- Designing dependable circuits → evaluation of robustness against faults
- Faults:
  - Natural phenomena
  - Attacks
Introduction

- Designing dependable circuits → evaluation of robustness against faults

- Faults:
  - Natural phenomena
  - Attacks
Introduction

- Designing dependable circuits → evaluation of robustness against faults
- Faults:
  - Natural phenomena
  - Attacks

α particle
Introduction (cont'd)

- Consequences of faults = errors
  - Signals stuck at 1 or at 0
  - Bit flips in memories

- Goal: to ensure a given level of robustness → analysis of the potential consequences of errors
  - Usually based on fault-injection techniques
Using formal methods

- Most results based on enumerative techniques
  - Krautz et al. DATE'06: symbolic simulation to characterize correction capabilities
  - Seshia et al. DATE'07: SMV to identify latches that must be protected
  - Fey et al. ISQED'08: SAT-solving to compute a measure of the robustness
Using formal methods

- Goal: to avoid enumeration → meta-model for faults in ACL2
- Transition and output functions
  \[ \delta : I \times S \rightarrow S \]
  \[ \lambda : I \times S \rightarrow O \]
- Fault-injection function \( f \)
Overall framework

VHDL RTL description

VSYML

XML format

Characterization of errors

Translated

ACL2 macros

ACL2 code: transition and output functions + fault injection + properties
Overall framework

\[ \delta : I \times S \rightarrow S \]
\[ \lambda : I \times S \rightarrow O \]

VHDL RTL description

VSYML

XML format

Translator

Characterization of errors

ACL2 macros

ACL2 code: transition and output functions + fault injection + properties
Overall framework

VHDL RTL description

VSYML

XML format

Translator

Characterization of errors

ACL2 macros

ACL2 code: transition and output functions + fault injection + properties
Overall framework

VHDL RTL description

VSYML

XML format

Characterization of errors

ACL2 macros

Translator

$\delta : I \times S \rightarrow S$

$\lambda : I \times S \rightarrow O$

ACL2 code: transition and output functions + fault injection + properties

ACL2'2009
Single fault model

- Fault model: presence of an error in a single register
  - $f : S \rightarrow S$
  - $f(s) \neq s$
  - only one register differs from $s$ to $f(s)$
  - $\forall f(s) = \text{inject}_k(s)$
Single fault model

- Example: TMR counter

\[ s_n = (X,X,X) \Rightarrow \delta(i, f(s_n)) \iff \delta(i, s_n) \]

\[ s_0 = (X,X,X) \Rightarrow \delta(i, f(\delta^n(i,s_0))) \iff \delta(i, \delta^n(i,s_0)) \]
Hierarchical model

- TMR example:
  - CPU time for property 1: 0.01 s
  - CPU time for property 2: 6.35 s

- Unacceptable times for larger circuits → hierarchical model that takes advantage of the compositional circuit structure
Hierarchical model

- Exporting characteristic properties

Component $C_2$

$\delta_2: I_2 \times S_2 \rightarrow S_2$
$\lambda_2: I_2 \times S_2 \rightarrow O_2$
$Sp_2: S_2 \rightarrow B$
$Sreach_2: S_2 \rightarrow B$
Error function $f_2$

Component $C_1$

$\delta_1: I_1 \times S_1 \rightarrow S_1$
$\lambda_1: I_1 \times S_1 \rightarrow O_1$
$Sp_1: S_1 \rightarrow B$
$Sreach_1: S_1 \rightarrow B$
Error function $f_1$
Hierarchical model

- Exporting characteristic properties

Component $C_2$

- $\delta_2: I_2 \times S_2 \rightarrow S_2$
- $\lambda_2: I_2 \times S_2 \rightarrow O_2$
- $Sp_2: S_2 \rightarrow B$
- $Sreach_2: S_2 \rightarrow B$
- Error function $f_2$

Component $C_1$

- $\delta_1: I_1 \times S_1 \rightarrow S_1$
- $\lambda_1: I_1 \times S_1 \rightarrow O_1$
- $Sp_1: S_1 \rightarrow B$
- $Sreach_1: S_1 \rightarrow B$
- Error function $f_1$
Hierarchical model

- Exporting characteristic properties

Component $C_2$

- $\delta_2: I_2 \times S_2 \rightarrow S_2$
- $\lambda_2: I_2 \times S_2 \rightarrow O_2$
- $\text{Sp}_2: S_2 \rightarrow B$
- $\text{Sreach}_2: S_2 \rightarrow B$
- Error function $f_2$
Hierarchical model

- Example: component TMR register

```
(defspec TMR
 ; - Signatures
 (((TMR-Sp * *) => *) ; state recognizer
 (((TMR-next * *) => *) ; δ
 (((TMR-out_value * *) => *) ; λ
 (((TMR-e_detect * *) => *)
 (((TMR-reach_state *) => *) ; fault-free states
 (((TMR-error *) => *))) ; error
 (local (defun TMR-Sp (x)
    ...))
 (local (defun TMR-next (i s)
    ...))
 ...
```
Hierarchical model

- Example: component TMR register

```
(local (encapsulate
  (((TMR-error *) => *)
  (local (defun TMR-error (x) (TMR-inject1 x)))
  (defthm TMR-error-type1
    (equal (TMR-Sp (TMR-error x)) (TMR-Sp x)))
  (defthm TMR-error-def1
    (implies (TMR-Sp x)
      (not (equal (TMR-error x) x))))
  (defthm TMR-error-def2
    (or (equal (TMR-error x) (TMR-inject1 x))
      (equal (TMR-error x) (TMR-inject2 x))
      (equal (TMR-error x) (TMR-inject3 x)))
  :hints ("Goal" :in-theory (disable TMR-Sp))
  :rule-classes nil)))
```

...
Hierarchical model

- Example: component TMR register

; Robustness-related properties:
(defthm TMR-thm-hardened-1
  (implies (and (TMR-Sp S)
     (TMR-reach_state S)
     (true-listp I)
     (natp (nth *TMR/in_value* I))
     (booleanp (nth *TMR/ld_flag* I))
     (equal (len I) 2))
  (equal (TMR-next I (TMR-error S))
    (TMR-next I S)))
:hints (("Goal" :use
        (:instance TMR-error-def2 (x S))))
:rule-classes :rewrite)
...
Hierarchical model

- Example: ATM
Hierarchical model

Example: ATM

(defspec ATM-TMR
  ; - Signatures
  ( ((ATM-Sp          *) => *)   ; state recognizer
    ((ATM-next      * *) => *)   ; δ
    ((ATM-start_op  * *) => *)   ; λ
    ((ATM-keep      * *) => *)
    ((ATM-outc     * *) => *)
    ((ATM-e_detect * *) => *)
    ((ATM-reach_state *) => *)   ; fault-free states
    ((ATM-error    * *) => *))  ; error
  ...

C₂
Hierarchical model

**Example: ATM**

```lisp
(local (encapsulate ; for register n
   (((ATM-n_reg-error *) => *))
(defun ATM-n_reg-Sp (S) (TMR-Sp S))
(defun ATM-n_reg-next (I S) (TMR-next I S))
(defun ATM-n_reg-out_value (I S)  
   (TMR-out_value I S))
(defun ATM-n_reg-e_detect (I S)  
   (TMR-e_detect I S))
(defun ATM-n_reg-reach_state (S)  
   (TMR-reach_state S))
(local (defun ATM-n_reg-error (S) (TMR-error S)))
(definstance TMR n_register
   :functional-substitution
   ((TMR-error ATM-n_reg-error))
   :rule-classes :rewrite))
```
Hierarchical model

- Example: ATM

(defthm ATM-thm-hardened-1
  (implies (and (ATM-Sp S) (ATM-reach_state S)
    (true-listp I) (equal (len I) 7)
    (booleanp (nth *ATM/reset* I))
    (booleanp (nth *ATM/inc* I))
    (natp (nth *ATM/cc* I))
    (natp (nth *ATM/codin* I))
    (booleanp (nth *ATM/val* I))
    (booleanp (nth *ATM/done_op* I))
    (booleanp (nth *ATM/take* I))
    (equal (ATM-next I (ATM-error S))
           (ATM-next I S)))
  :hints ("Goal" :use (:instance ATM-error-def2 (x S))
           :in-theory (disable booleanp))
  :rule-classes :rewrite)
<table>
<thead>
<tr>
<th>Proof (book)</th>
<th>CPU time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Register with error-detection</td>
<td>0.24 s</td>
</tr>
<tr>
<td>TMR register</td>
<td>0.88 s</td>
</tr>
<tr>
<td>ATM with error-detecting registers</td>
<td>2.34 s</td>
</tr>
<tr>
<td>ATM with TMR registers</td>
<td>10.58 s</td>
</tr>
<tr>
<td>ATM with TMR registers (flat)</td>
<td>2559.14 s</td>
</tr>
</tbody>
</table>

* Intel Core 2 Duo
Conclusion

- First results → ACL2 can be useful for some kinds of faults/properties
- Improvements
  - Single fault:
    \[ f(s_i) \neq s_i \Rightarrow \forall k \neq i, f(s_k) = s_k \]
    instead of
    \[ \forall f(s) = \text{inject}_k(s) \]
Conclusion

- Improvements (cont'd)
  - Instantiation of components:
    - defcomp
      
      (defcomp TMR ...)
    - instcomp
      
      (defsrec ATM-TMR
       ...
       (local (instcomp TMR n))
       (local (instcomp TMR ok))
       (local (instcomp TMR code))
       ...
      )
THANKS!

ZingerBug.com