Lessons on Writing Papers

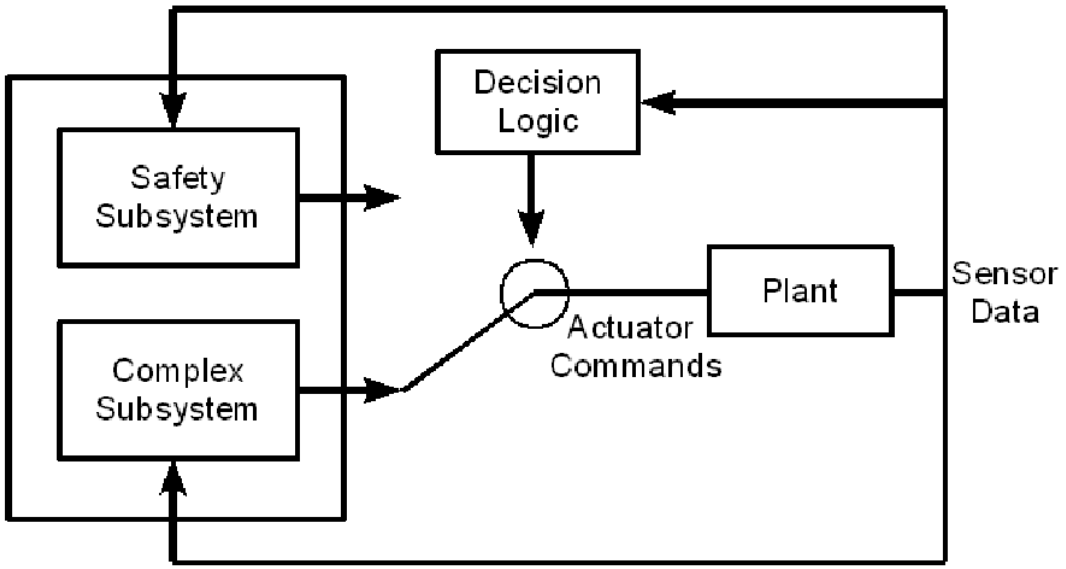
Stanley Bak
1/23/2009

History

- Paper submitted in 2008 for RTSS
  - Rejected
- Paper submitted in 2009 to RTAS
  - Accepted
- The difference?

Main Contribution

- Simplex Architecture on an FPGA
- We move the Safety Subsystem and Decision Logic to hardware for additional safety



Changes

- Motivation for changes
  - Reviewer suggestions
  - *"The Craft of Scientific Writing"* - Alley
- Combined with recommendations from professors, of course

Reviewer Suggestions

Reviewer Comments

- Several categories of comments
  - Misunderstandings (reword and clarify)
  - Small Changes (modify sentence or paragraph)
  - Large Changes (add/remove sections)

Misunderstandings

- “there is a serious problem when the decision logic does not receive signals from the complex processor... The paper just states that the last value is stored and used. I ask myself, how long is such a stored valid?”

- Unnecessary implementation details may confuse readers

  We store the most-recent output for each of the controllers and use that value when a decision must be made

Small Changes Comments

- “Why FPGA and not software?”
  - Agreed: You can use software, if you accept its associated complexity
- “Your example is bad because...”
  - Remove example
- “You said a formally verified OS is infeasible but what about the work by...”
  - Change to “Although great progress has been made towards producing a verifiable RTOS[3], most current RTOSes have been neither formally verified nor exhaustively tested”

Small Changes Comments (2)

- “This isn't new or interesting”
  - Emphasize what is new and non-obvious, and say why it isn't obvious

Two Simplex safety-critical components, the safety controller and decision module, are moved to a dedicated processing unit, **not for the typical HW/SW co-design reasons of power and performance**, but instead to provide isolation from software-related complexity.

Small Changes Comments (3)

- “There's no mention of limitations”

- Not mentioning limitations implies the solution is **always** easier, cheaper and better
  - which makes reviewers nervous

Simplex (both versions) should not be regarded as a one-size-fits-all robustness approach. To guarantee safety, we must [do X, Y, and Z, which are not always possible].

Small Changes Comments (4)

- Other mentions of limitations

An inverted pendulum, however, does not completely lend itself to our end-to-end design process.

One drawback of the System-Level version of Simplex is that additional resources are required to run the decision module and complex controller, such as the FPGA.

Large Changes

- Originally we had a section which formally model-checked the Simplex Architecture for safety given certain properties of the controllers

- The reviewers were not impressed
  - “Except for informal arguments, there is no proof that these 4 are necessary and sufficient to guarantee the correctness of the architecture.”

Large Changes (2)

- Instead, the reviewers hinted towards a more interesting direction
  - “The problem is that there don't seem to be any science in the sense of a systematic way of figuring out the correct implementation of the decision logic or the safety subsystem.”

- Solution: add section about modeling checking the decision logic and safety subsystem

Large Changes (3)

- We also added a section about AADL generation, which generated reviewer comments
  - “Authors should give a brief introduction or related references on the AADL model.”
  - “The AADL material should be improved and better motivated, as it is it seems to add little or nothing to the paper.”

“The Craft of Scientific Writing”
Michael Alley

General Changes from TCOSW

- Avoid pretentious words:
  - Utilize / Utilization -> Use
  - Facilitate -> Case / Bring About
  - Component -> Part

- Change title:
  - Demonstrate clear match to one of the conference topics
  - "*novel hardware or system architectures for real-time or embedded systems*"

Change Title

- From TCOSW, titles should do two things
  - Identify field of study
  - Separate from all other publications in that field

- Previous Title
  - *"The Hardware Simplex Architecture for Real-Time, Safety-Critical Systems"*

- New title
  - *"The System-Level Simplex Architecture for Improved Real-Time Embedded System Safety"*

List formating

- Lists should be of 2, 3, or 4 items
- Lists should be concise
- Lists should be precise
- Parallel structure

Main Contributions

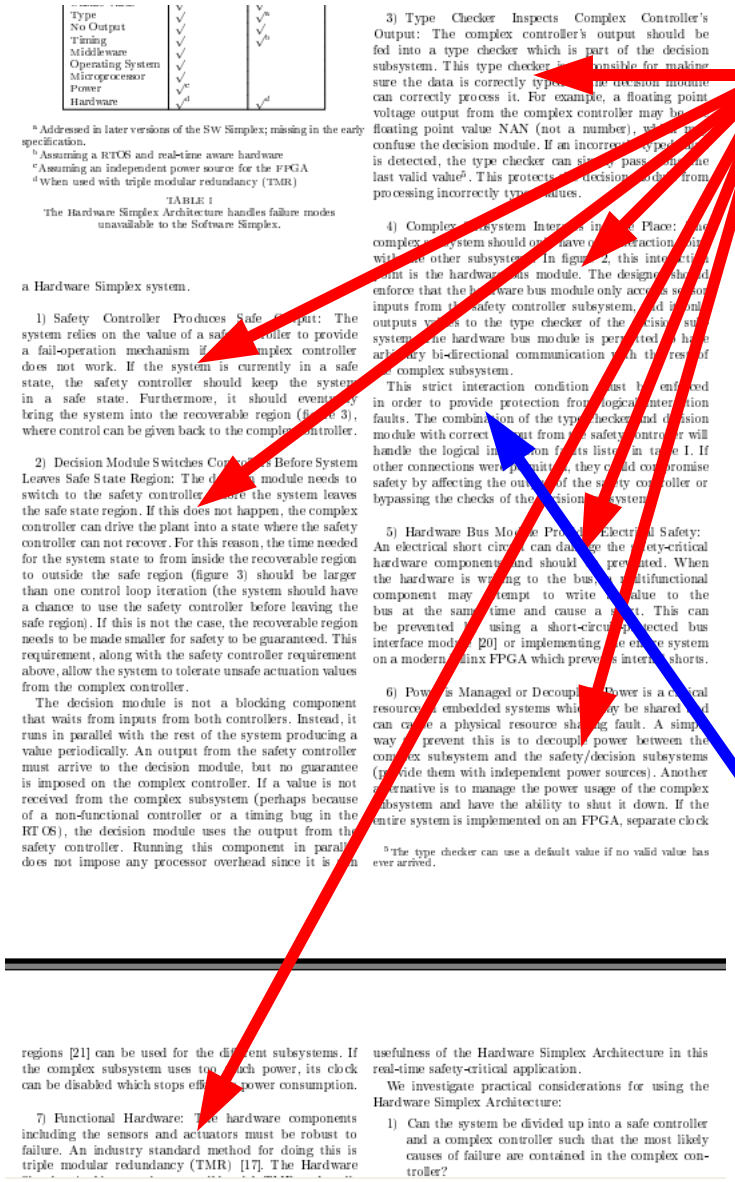
- original:

1) The Hardware Simplex Architecture can handle a superset of the failure modes of the SW Simplex Architecture. Additional faults that can be safely handled include RTOS timing faults, middleware bugs, and microprocessor errors.
2) The HW Simplex Architecture generates negligible processor overhead when compared to running a single controller with no fail-operational mechanism. This contrasts with the SW Simplex Architecture which requires the processor to run the decision module and safety controller.
3) The design model is formally specified in Maude [11] and two key properties are model checked:
   a) The system is safe regardless of the output of the complex controller, faults in the RTOS, or failures in the microprocessor
   b) If the complex controller, RTOS, and microprocessor perform correctly, the system always uses the high-performance output of the complex controller

- fixed:

- The design of the System-Level Simplex Architecture which can handle a superset of the failure modes of previous Simplex versions
- An end-to-end design process that both verifies a valid System-Level Simplex AADL architecture model and can generate hardware VHDL code from a finite-state machine description
- Empirical verification of both the practicality of end-to-end System-Level Simplex design, and the robustness guarantees through fault-injection testing in two case studies

Safety Rules (original)



- Long list of seven design rules
  - Spanning two pages
- Some with multiple paragraphs

Safety Rules (fixed)

- We changed the safety rules to three categories of necessary properties

We have identified several necessary conditions required by a System-Level Simplex design that are checked by our OSATE architecture checker. The properties can be classified into resource isolation properties, data consistency properties, and data flow properties. We briefly describe each of these, along with associated failures that may occur if they are not present.

Added Other Lists

- All items are questions
- All questions start with different word

- Can the system be divided up into a safe controller and a complex controller, such that the most likely causes of failure are contained in the complex controller?
- Is the System-Level Simplex end-to-end design process effective in the cardiac pacemaker context?
- How do the resultant safety guarantees compare to those of existing pacemakers?

Paper Structure

- Original:
  - Introduction
  - **Related Work**
  - HW Simplex Design
  - Case Studies
    - Pacemaker
    - Inverted Pendulum
  - **HW Simplex Model**
  - Conclusions
- Revised:
  - Introduction
  - SL Simplex Design
  - **Design Process**
  - Case Studies
    - Pacemaker
    - Inverted Pendulum
  - **Related Work**
  - Conclusions

Questions & Comments?

- I'll send out both versions of the paper and this presentation...