Computer Aided Verification of Safety Critical Real-Time Systems

Dr. Mark Lawford
Assistant Professor
Dept. of Computing And Software
Faculty of Engineering
McMaster University

lawford@mcmaster.ca

Outline

- Safety Critical Systems
- CAV: Computer Aided Verification
- Example: Functional Verification
- Example: Concurrent real-time systems verification

Safety Critical Systems

Failure results in:

- physical injury or loss of life
- unacceptable financial loss

Applications Areas:

- Medical equipment
- Aerospace
- Process control - e.g. Darlington Nuclear Generating Station Shutdown Systems (SDS)

NOTE: You have one chance to get it right!

Example Software System Failures:

- Medical equipment -

  THERAC-25 radiation therapy machine killed several patients

- Aerospace -

  Space shuttle - 1st flight delayed timing bug at initialization

  Ariane 5 launcher - 1st flight self-destructed after 45 seconds due to floating point overflow error in Inertial Guidance System

Other software related problems:

Process control - e.g. Nuclear Generating Station Shutdown Systems (SDS)

- Spurious trips cost $$$

- Difficult to make modifications & even more difficult to get regulatory approval for changes

How can such systems be handled properly?

Review, Review, Review . . .

Multiple independent reviewers do:

- Software Requirements Specification (SRS) review
- Software Design Description (SDD) review
- Code review

Then . . . Test, Test, Test:

Independent testers do one & only one of:

- Unit Testing (UT) - test each individual program separately
- Software Integration Testing (SWIT) - test components when they are combined
- Validation Testing - test system against original system requirements

Logic provides a precise, unambiguous method of specifying system details for reviewers and testers

That’s still not enough!

- I’ve discovered incorrect designs that have been reviewed by as many as 5 different people - there is just too much detail for a person to catch everything!

- Testing can’t cover all possible cases - e.g. 1st shuttle flight initialization CPU overload had 1 in 67 probability of occuring

- Minor changes result in another extensive (& expensive) round of testing & review

Logic provides a means of mechanizing verification details - Computer Aided Verification!

Computer Aided Verification

What is CAV? . . . Prove, prove, prove!

Use tools to mathematically "prove" a design implements a well defined specification. E.g.

- Automated theorem proving of functional equivalence (i.e. use PVS or IMPS to prove for all inputs x: Spec(x) = Design(x))
- Model-checking automatically verifies that a Design is a model of a Spec written as a logical formula

Why use CAV Tools?

- Independent check of system unaffected by verifier's expectations
- Domain coverage - Tools can often be used to check ALL input cases
- Tools let you automate verification and reverification
- Provide additional capabilities (e.g. generation of counter example for debugging, type checking, verifying whole classes of systems, etc.)

Example: Reactor Shutdown System (SDS)

What is an SDS?

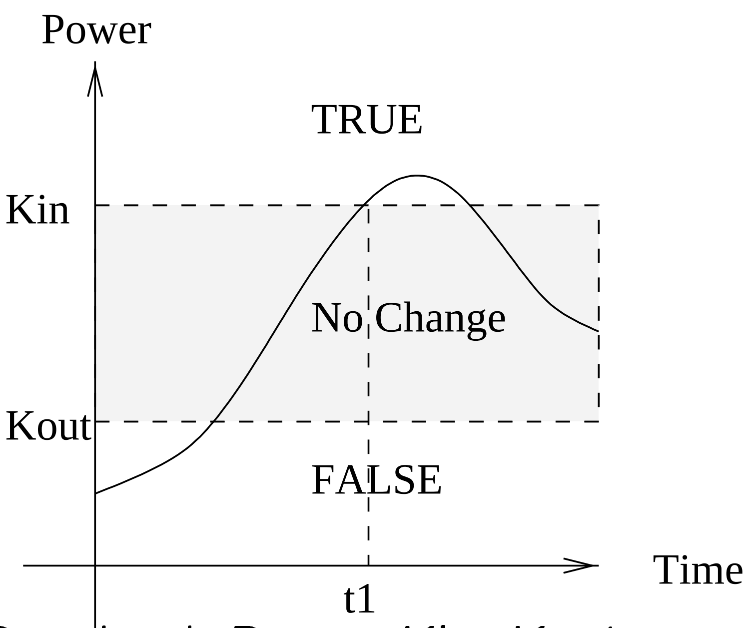
- watchdog system that monitors system parameters
- shuts down (trips) reactor if it observes "bad" behavior
- process control is performed a separate Digital Control computer (DCC) - not as critical

Consider simple subsystem: Power Conditioning

- Many sensors have a Power threshold below (or above) which readings are unreliable so it's "conditioned out" for certain Power levels.
- A deadband is used to eliminate sensor "chatter"

Idea: Use code reuse - write one general routine and pass in sensor parameters for different sensors

General Power Conditioning Function



PwrCond(Prev:bool, Power, Kin, Kout:posreal):bool =

$Power \leq Kout$	$Kout < Power < Kin$	$Power \geq Kin$
$FALSE$	$Prev$	$TRUE$

PVS (Prototype Verification System), a "proof assistant" can automatically check for completeness and determinism.

Problem: Determinism check fails when $Kin \leq Kout$.

Why? Implicit (undocumented) assumption from diagram that $Kin > Kout$

When Power:

- drops below $Kout$, sensor is unreliable so it's "conditioned out" ($PwrCond = FALSE$).
- exceeds Kin, the sensor is "conditioned in" and is used to evaluate the system.
- is between $Kout$ and Kin, the value of $PwrCond$ is left unchanged by setting it to its previous value, $Prev$.

E.g. For the graph of $Power$ above, $PwrCond$ would start out FALSE, then become TRUE at time $t1$ and remain TRUE.

PVS Specification of general *PwrCond* function

```
  PwrCond(Prev:bool, Power, Kin, Kout:posreal):bool = TABLE
    %----------------------------------------------------%
    |[Power<=Kout | Power>Kout & Power<Kin | Power>=Kin]|
    %----------------------------------------------------%
    |   FALSE      |         Prev            |   TRUE   ||
    %----------------------------------------------------%
  ENDTABLE

PwrCond_TCC1: OBLIGATION
 (FORALL (Kin: posreal, Kout: posreal, Power: posreal):
  NOT (Power <= Kout AND Power > Kout & Power < Kin)
    AND NOT (Power <= Kout AND Power >= Kin)
    AND NOT ((Power > Kout & Power < Kin) AND Power >= Kin));

PwrCond_TCC1 :

[-1]    Kin!1 > 0
[-2]    Kout!1 > 0
[-3]    Power!1 > 0
[-4]    Power!1 <= Kout!1
[-5]    (Kin!1 <= Power!1)
  |-------
[1]     FALSE

Rule?
```

SDS Safety/Performance Considerations:

- Check for short circuits/sensor failures
- Use dead-band to eliminate "chatter"
- Power dependent set points increase operating margin
- "Condition out" sensor in unreliable operating region
- Digital trip output uses "-ve logic" (fail-safe in power loss)

Additional SDS Considerations:

- Use multiple sensors to improve reliability
- There are many sensor trips, parameter trips, channel trips, warning lights, input buttons, etc. that all have to be given the same fail-safe treatment (i.e. 100s of functions)!

Electrical student's reaction:

"But I never had to worry about that stuff in Matlab?"

- Welcome to the real world.

Computer science student's reaction:

"Still way simpler than my 1st java text-editor applet."

- Did it ever crash?

Other Applications

- The Pentium^{Tm} floating point bug could have been detected by CAV.

- CAV was used after the bug was detected to prove the proposed fix corrected the problem.

- PVS has been used to verify similar circuits

Concurrent Real-time Example

Simple reactor trip system

- monitors plant parameters (Primary Heat Transport Pressure & Reactor Power) using sensors & A/D conversion

- if parameters exceed set-points in particular way, shutdown (trip) the reactor

- redundant systems run concurrently and perform majority vote to decide when to shutdown system

- old hardware implementation to be replaced by microprocessor based system with 0.1ms cycle time

Delayed Trip System

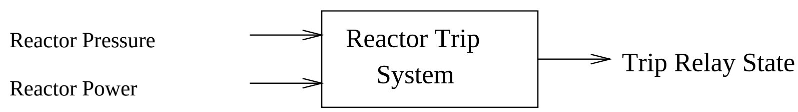


Figure 1: Block diagram for DTS

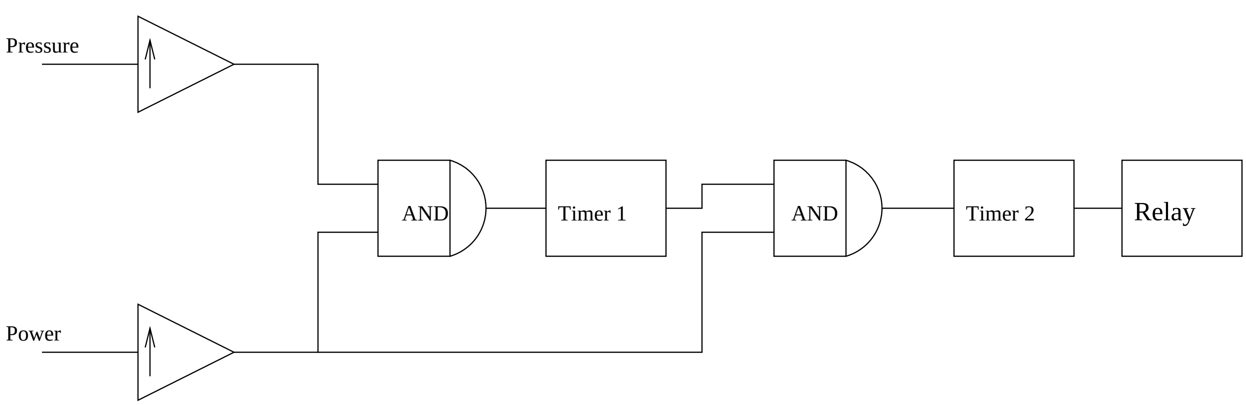


Figure 2: Analog implementation of DTS

Concurrent Real-Time Systems

Does this real-time control system do what we want?

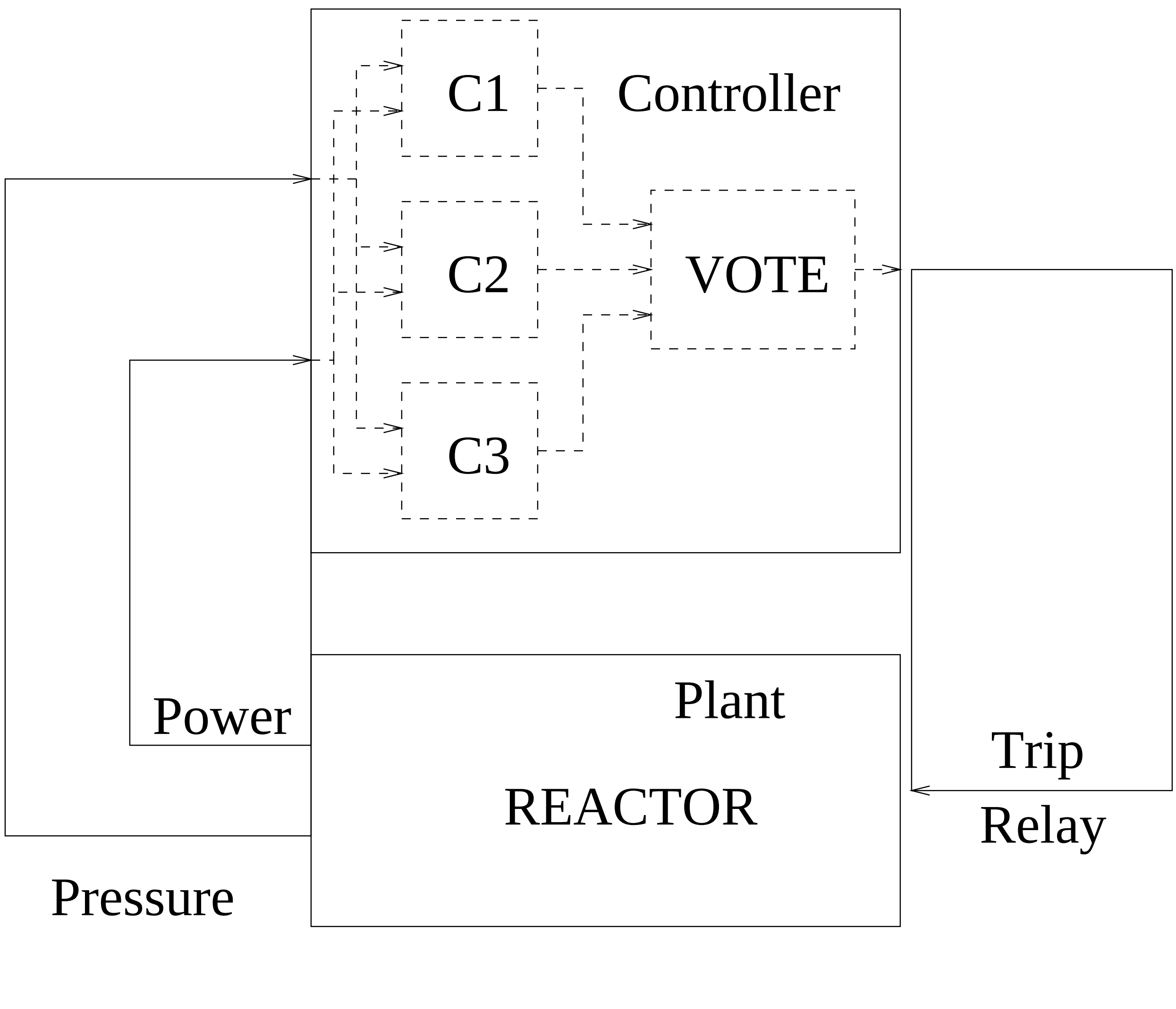


Figure 1: Block diagram for DTS

Research Interests

Computer Aided Verification:

- automating verification and re-verification tasks
- modeling & verification of concurrent real-time properties
- equivalence verification, model-checking & model reduction
- implementing provably correct safety critical systems in hardware using PLDs