

Verifying control  
systems using CSP,  
FDR, and  
*Handel-C*.

Alistair A. McEwan  
University of Surrey

## Introduction

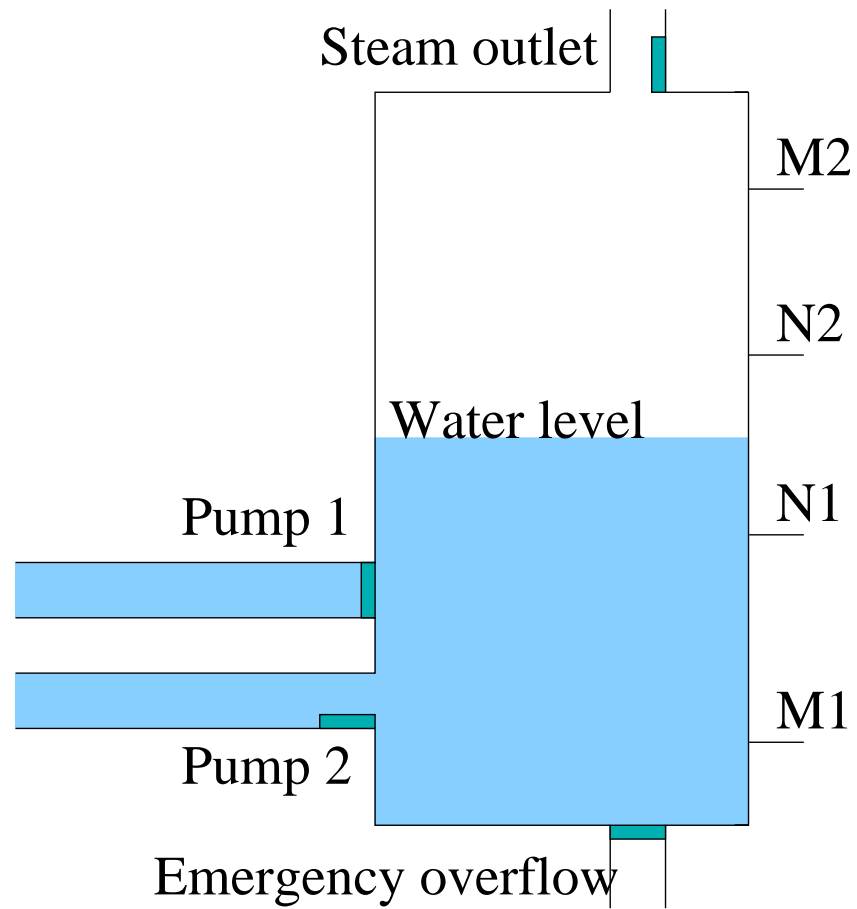
Investigate requirements →

automate verification →

automate implementation.

- Model a plant/system—existing or new
- Design control laws to operate this plant
- Control laws represent pre-learned intelligence
- Plant and control laws together give rise to autonomous system
- Autonomous system must be safe, and useful

## Case study: The Steam Boiler



## Formalisms and tools

- Communicating Sequential Processes (CSP): process algebra
  - concurrency, communication, refinement and abstraction
  - control system harness for abstract control laws
  - idiom uses concurrency and communication, not user state
  - further techniques for modelling component failure modes
- Failures-Divergences Refinement (FDR): model-checker for CSP
  - tests assertions, produces counter-examples
  - high level of expertise in exploiting bounded space
- Celoxica *Handel-C* implementation route to hardware
  - concurrency model maps to CSP: refinement calculus
  - CSP model produces networks of syncs: no variables

## The problem

- Separation of concerns: problem vs implementation
- Understand safety criteria, and domain requirements
- High level design: obscure verification and implementation details
- Automate verification (theorem provers are hard!)
  - fit models into bounded model-checker
  - state/process abstraction, compression etc
- Produce “embedded system”, on FPGA
  - Should be a natural consequence of design process

## Control Systems: pre-learned intelligence

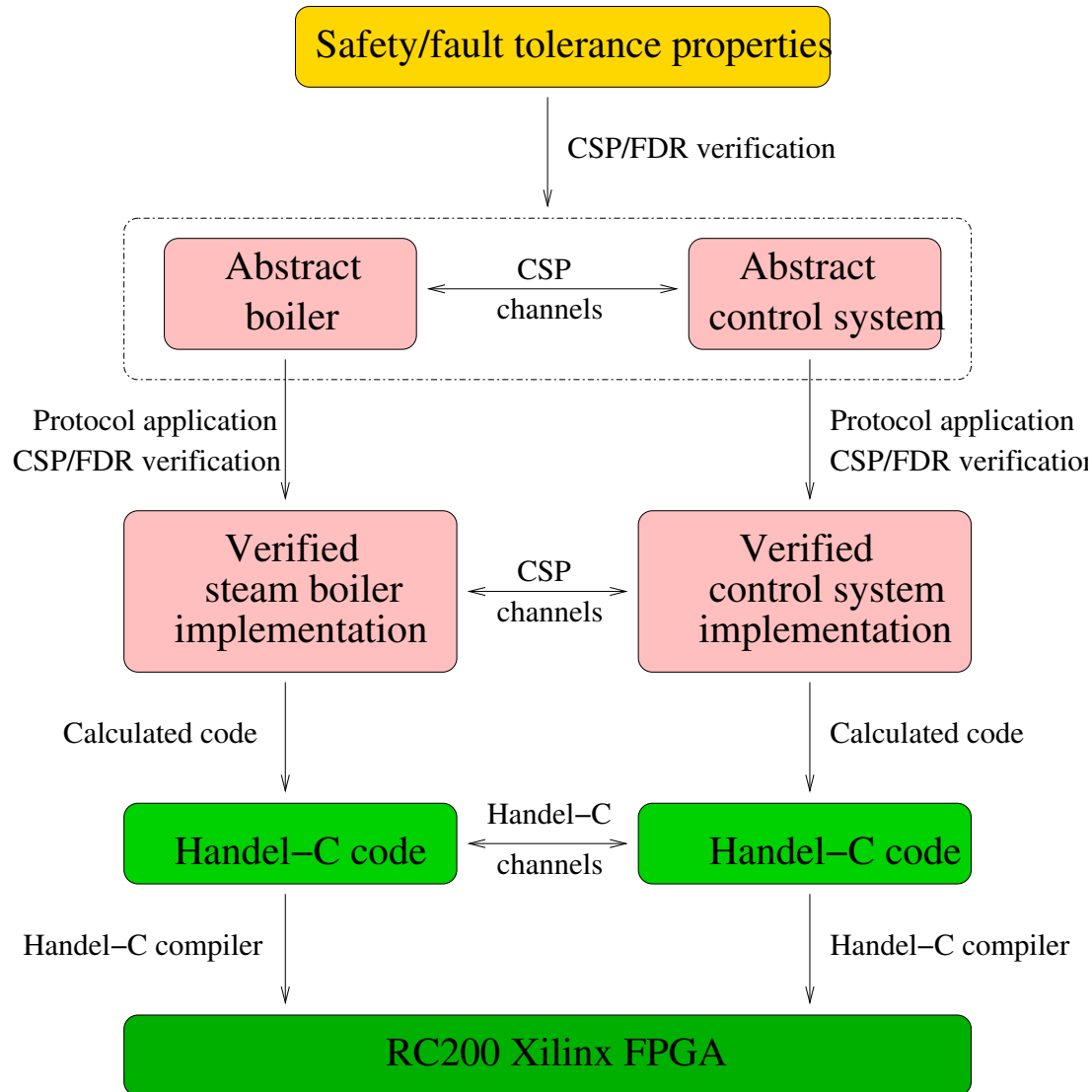
**Definition 1** *The structure of a control law*

$$\text{hypothesis} \hat{=} \text{fact}_1 \wedge \text{fact}_2 \wedge \dots \wedge \text{fact}_n$$

$$\text{Control Law} \hat{=} \text{hypothesis}(\text{sense}) \Rightarrow \text{conclusion}(\text{actuation})$$

$$\text{level}_1 \wedge \text{pumps\_open} \Rightarrow \text{close\_pumps}$$

- Control laws **abstract**: bear no relation to implementation
- Engineer concentrates on requirements
- Produce automatically verified specification in CSP
- Highly concurrent system: no explicit user state, no **local** view of global state
- Requirements calculation: test/refute using FDR



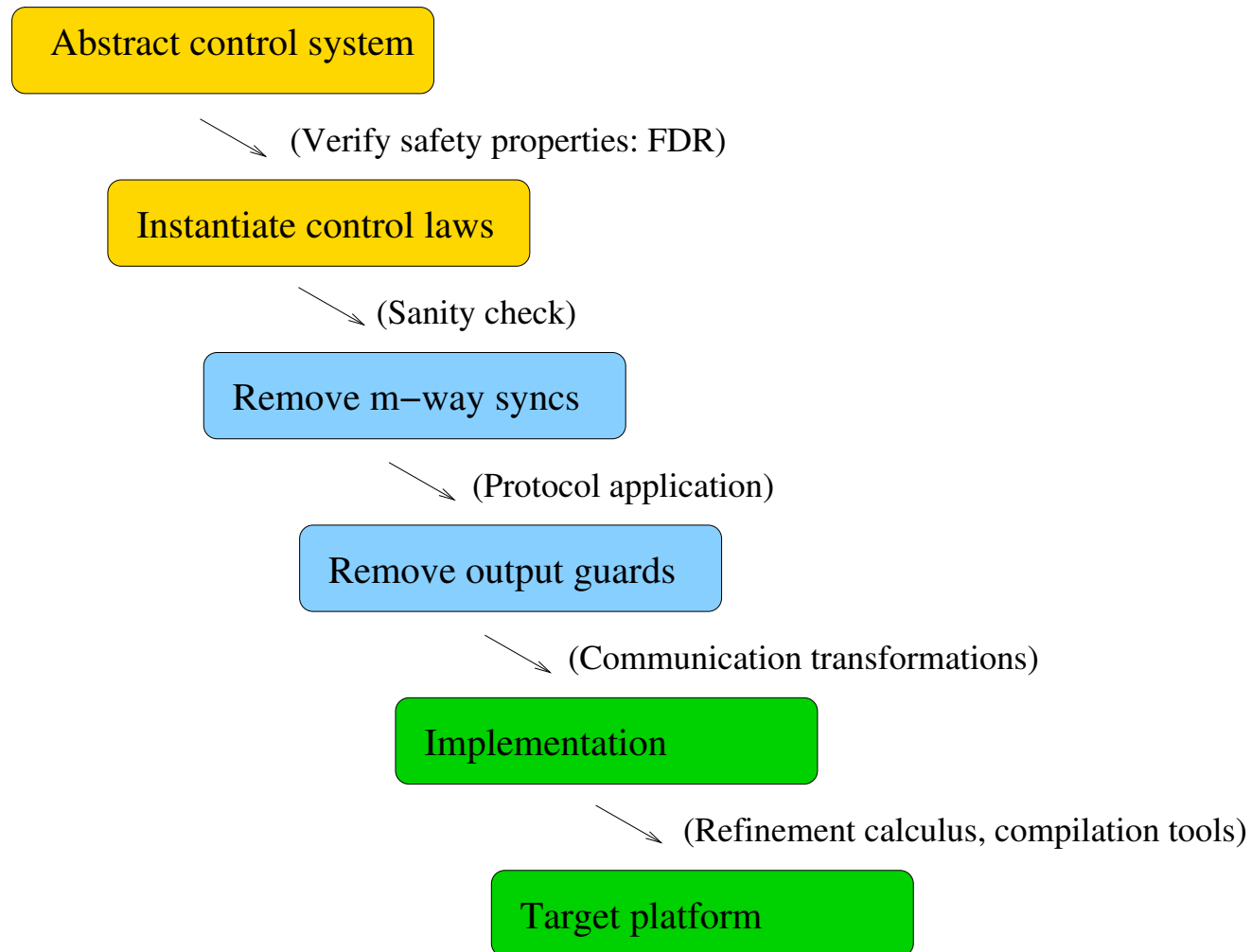
## Safety Specifications using CSP

- An example of a safety criteria
- The boiler should never be allowed to boom, or flood
- $\forall tr : \mathbb{P} \text{ seq } \mathcal{A}Boiler \bullet (boom \notin \text{ran } tr) \wedge (flood \notin \text{ran } tr)$ 
  - $\langle GtN2, tock, close\_pump1, tock, tock, tock, tock, tock \rangle \quad \checkmark$
  - $\langle GtN2, tock, open\_pump1, tock, tock, tock, tock, flood, \checkmark \rangle \quad \times$

assert

$Safety\ spec \sqsubseteq_T (Boiler \parallel Control\ system)$

## Calculating the Implementation



## Conclusions

- Verification of safety properties: requirements elicitation
- Model-checker gives confidence in safety arguments
- Refinement calculus gives confidence in *Handel-C* code
- Verified *Handel-C* implementation on FPGA
- Gate level verification?
- Model of existing plant/domain expertise?
- Number of rules, compositional properties?
- Unifying theories of programming, *Circus*?