APECS: An AADL and Polychrony based Embedded Computing System Design Environment with an Elevator Control Case Study

Matthew Anderson and Sandeep K. Shukla
Virginia Tech.

Partially funded by US Air Force Research Labs

Outline of the talk

Presentation Outline

Platform Design

Architecture Analysis and Design Language (AADL)

SAE International standard and framework for model based software systems

Platform Design

Architecture Analysis and Design Language (AADL)

SAE International standard and framework for model based software systems

AADL and its accompanying tool suites provide a modeling environment that allow for modeling:

- Target hardware platforms
- Complete software hierarchies
- Static and Dynamic system properties
  - Real-Time Schedulability
  - Resource Utilization

Software Hierarchy

Ocarina

An AADL tool suite developed in Ada to support building Distributed Real-Time Embedded Systems

Software Hierarchy

Ocarina

An AADL tool suite developed in Ada to support building Distributed Real-Time Embedded Systems

- Syntactic and Semantic Analysis
- Automated Code Generation
- Supports a variety of software languages
  - C/C++
  - Ada
  - Lustre and Esterel

Software Hierarchy

Ocarina

An AADL tool suite developed in Ada to support building Distributed Real-Time Embedded Systems

- Syntactic and Semantic Analysis
- Automated Code Generation
- Supports a variety of software languages
  - C/C++
  - Ada
  - Lustre and Esterel

So, what's missing?

Motivations for Polychrony

- A lack of formal semantics in C/Ada make verification difficult

Motivations for Polychrony

- A lack of formal semantics in C/Ada make verification difficult
- Lustre and Esterel have fully synchronous specifications.

Motivations for Polychrony

- A lack of formal semantics in C/Ada make verification difficult
- Lustre and Esterel have fully synchronous specifications.
- Software specifications still require a fine granularity, at the thread or function level.

Motivations for Polychrony

- A lack of formal semantics in C/Ada make verification difficult
- Lustre and Esterel have fully synchronous specifications.
- Software specifications still require a fine granularity, at the thread or function level.
- Thread synchronization
  - Oversynchronization
  - Deadlock

Motivations for Polychrony

- A lack of formal semantics in C/Ada make verification difficult
- Lustre and Esterel have fully synchronous specifications.
- Software specifications still require a fine granularity, at the thread or function level.
- Thread synchronization
  - Oversynchronization
  - Deadlock

What can be done?

→ Extend the backend of Ocarina with support for polychronous specifications from MRICDF sources.

Presentation Outline

Elevator Design Study

APECS Design Approach

Top-down system model design illustrated using an Elevator Case Study

Elevator Design Study

APECS Design Approach

Top-down system model design illustrated using an Elevator Case Study

- Physical Requirements
  - Centralized Control
  - Bank of four elevators
  - Service five floors

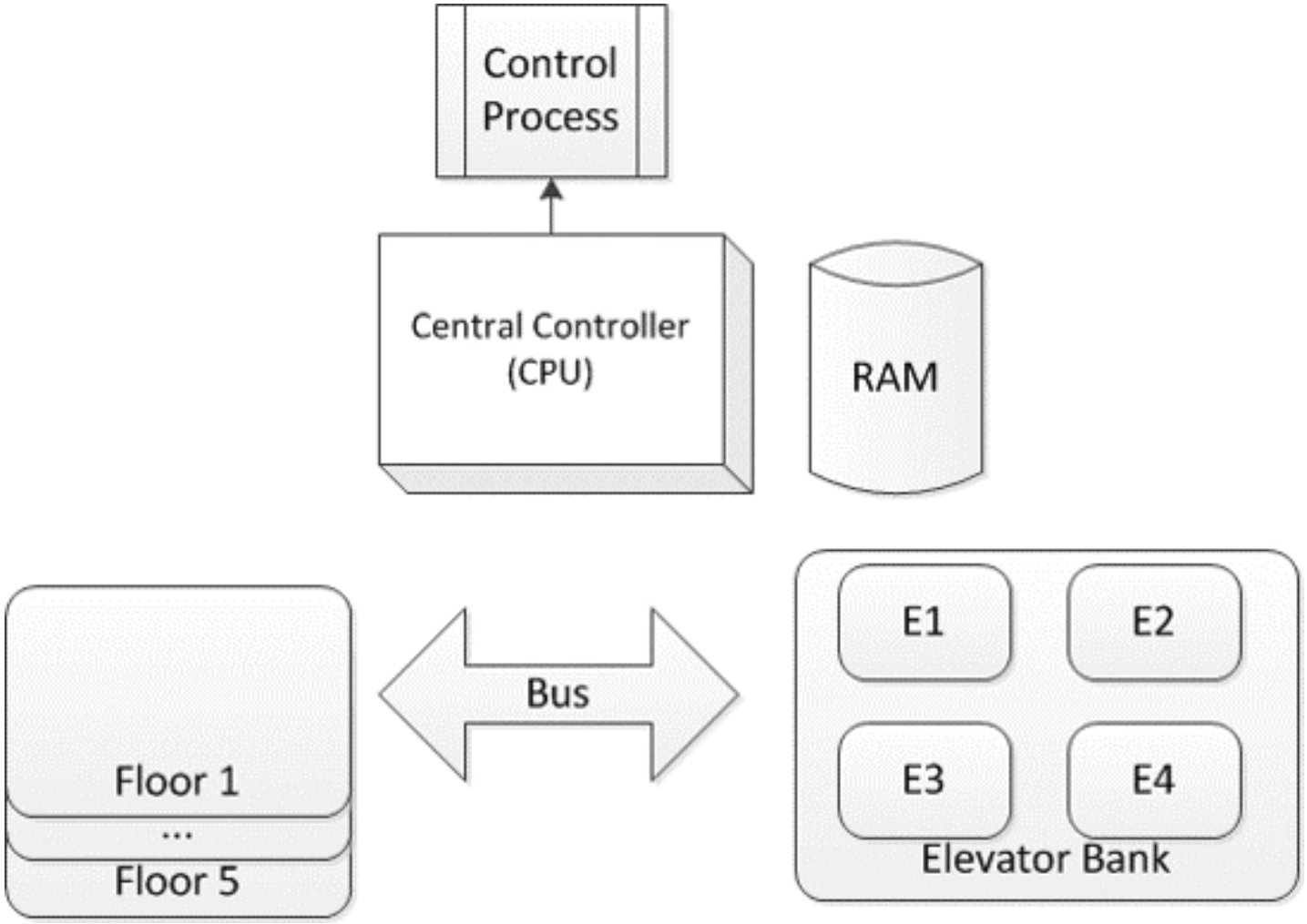
Elevator Design Study

APECS Design Approach

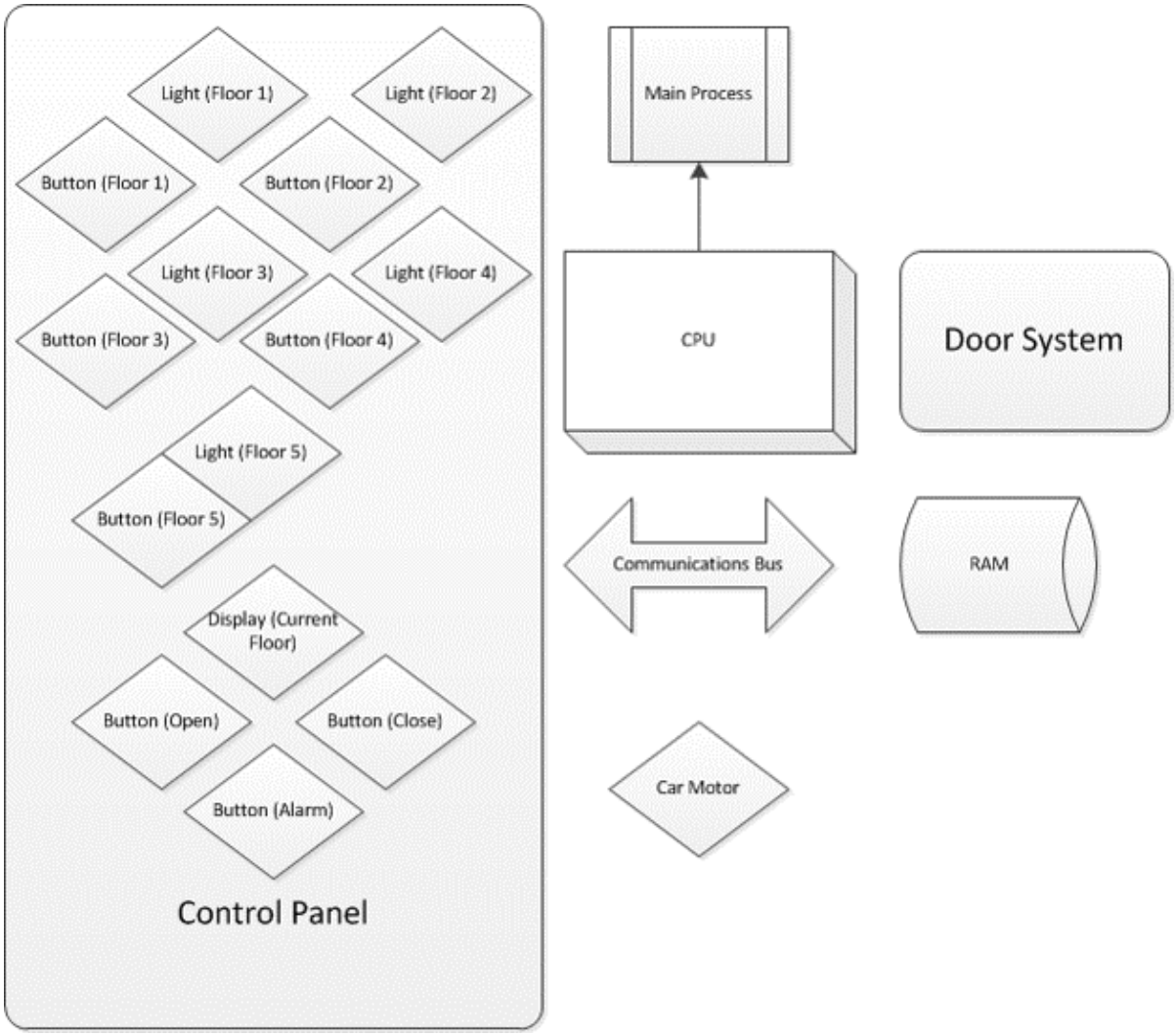
Top-down system model design illustrated using an Elevator Case Study

- Physical Requirements
  - Centralized Control
  - Bank of four elevators
  - Service five floors
- Safety and Behavior
  - Timely Operation
  - Safe door operation
  - Fire emergency response

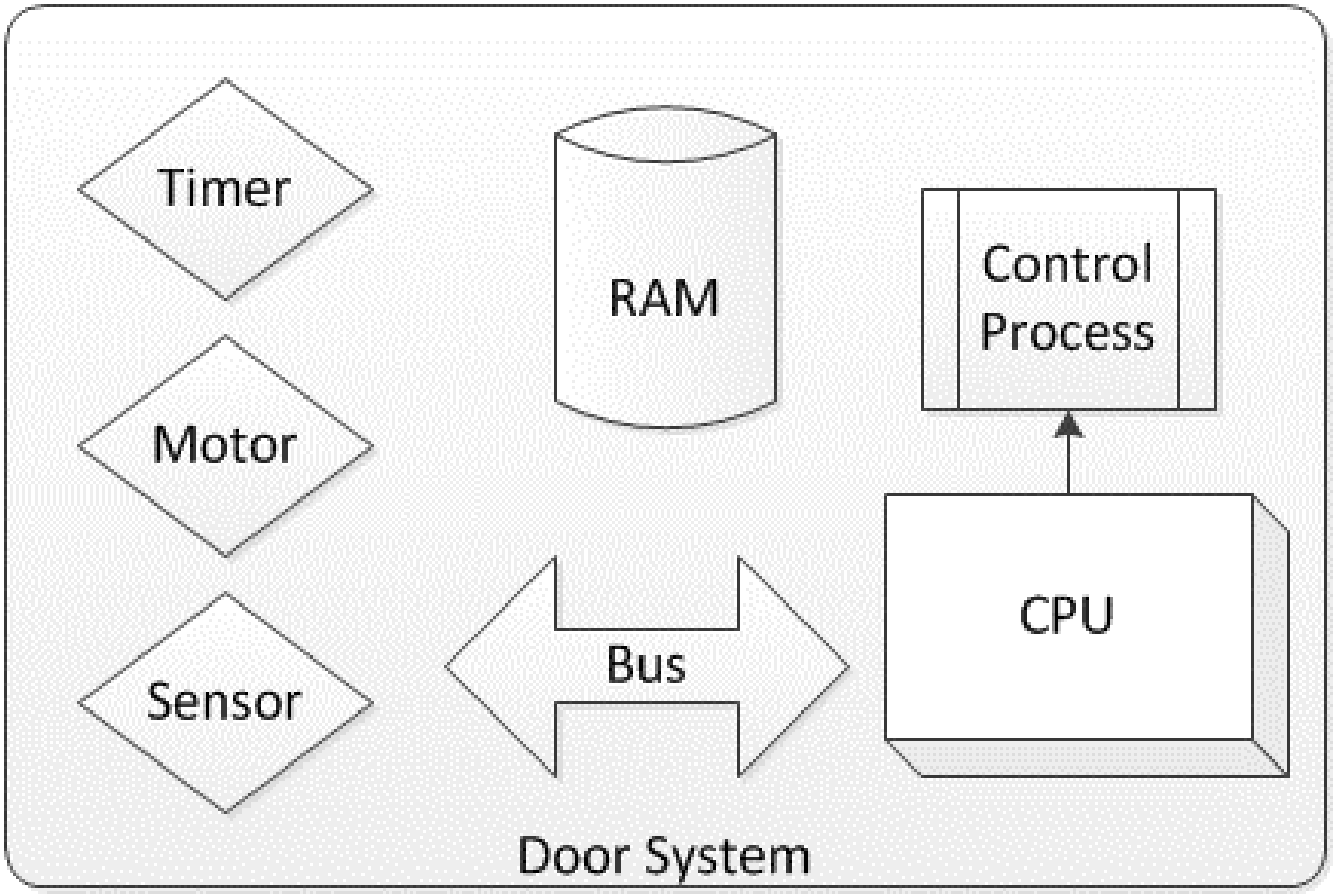
Top-Level Systems



Refining Systems



Refining Systems Cont'd

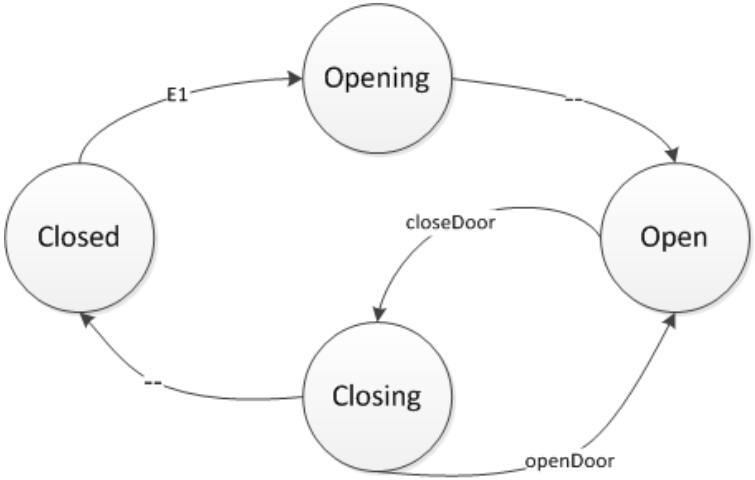


Door Behavior

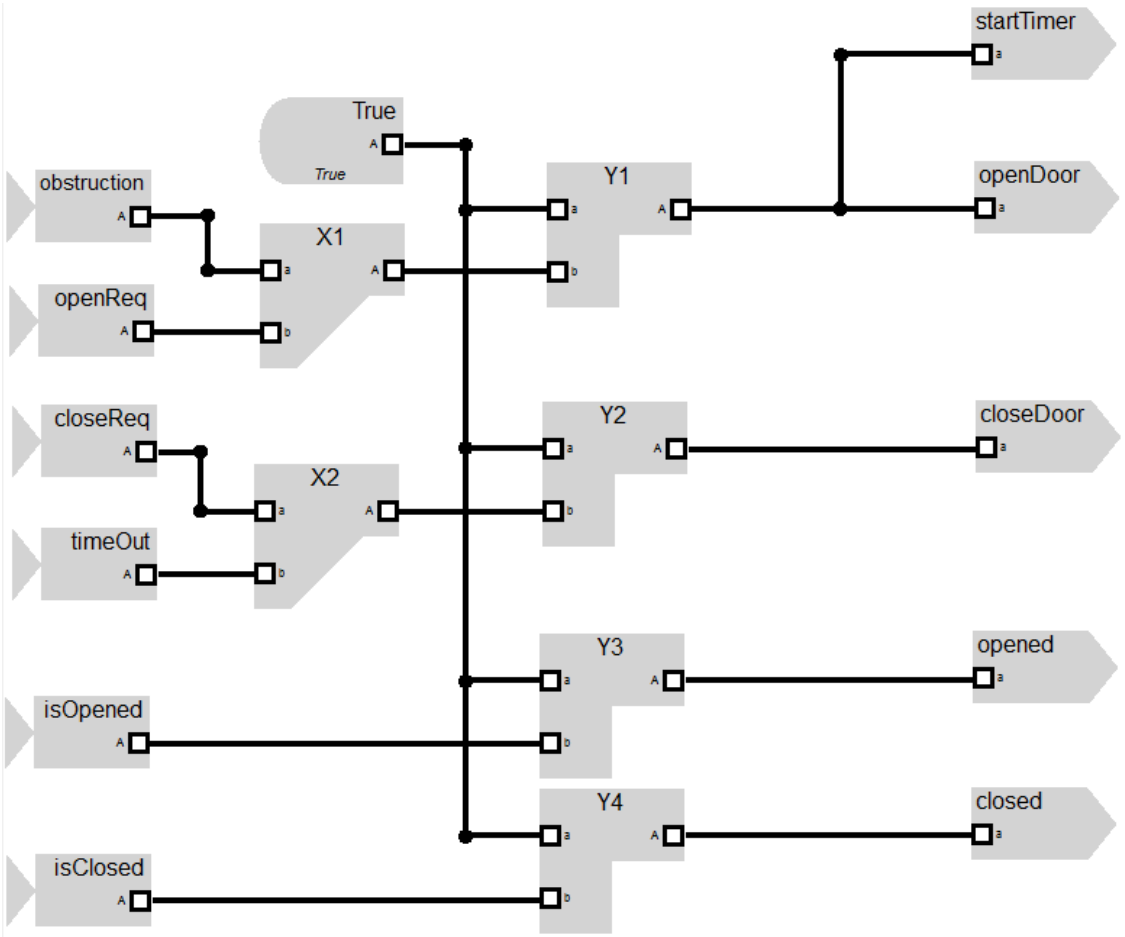
- State Table

Event	Conditions	Outputs
E1	OpenRequest	Motor <= open and Timer <= start
E2	Obstruction	Motor <= open and Timer <= start
E3	CloseRequest	Motor <= close
E4	Timeout	Motor <= close

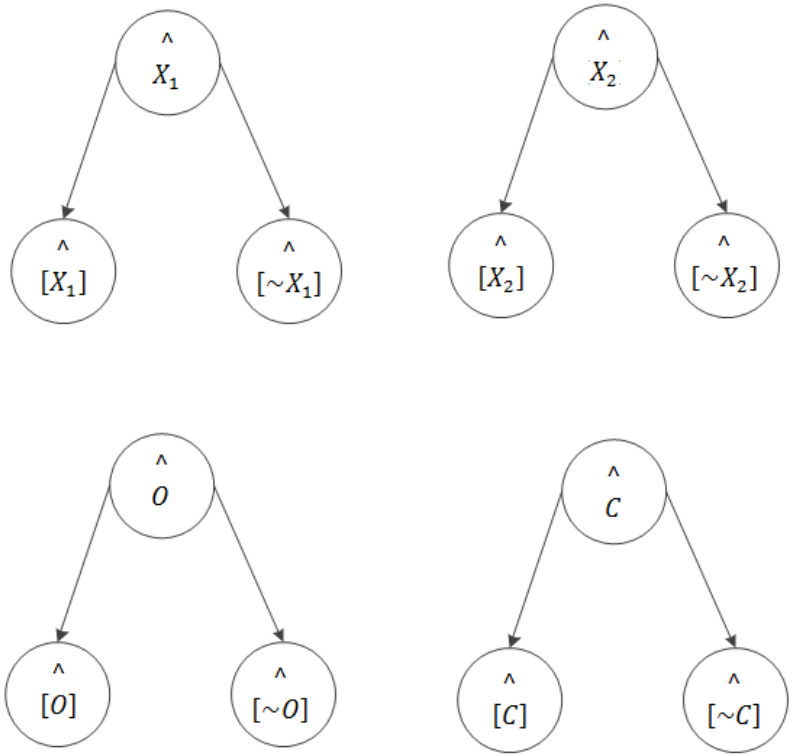
- State Machine



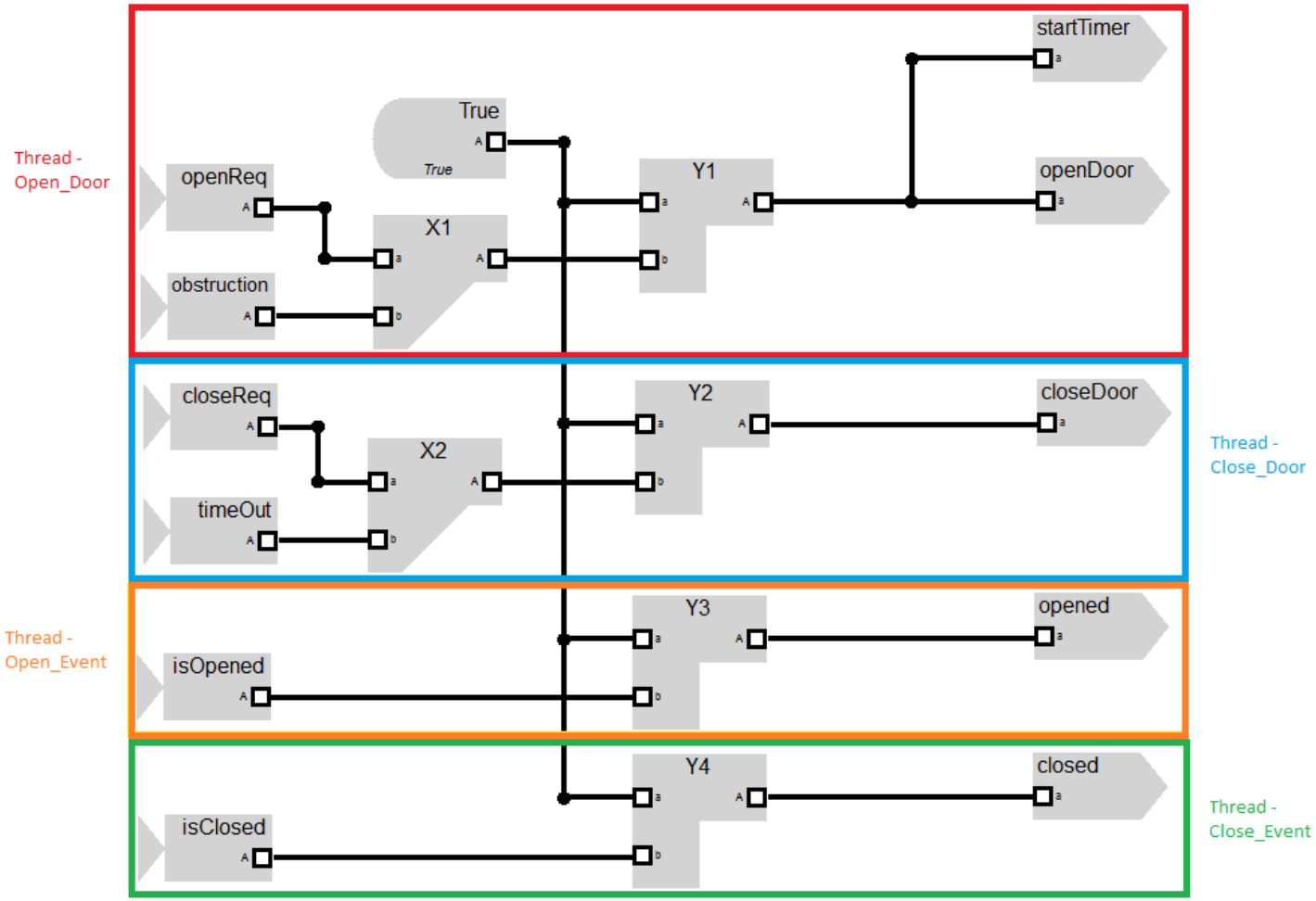
MRICDF Door Control Process



Clock Tree


$\hat{X}_1$
$[\hat{X}_1]$
$[\hat{\sim}X_1]$
$\hat{X}_2$
$[\hat{X}_2]$
$[\hat{\sim}X_2]$
$\hat{O}$
$[\hat{O}]$
$[\hat{\sim}O]$
$\hat{C}$
$[\hat{C}]$
$[\hat{\sim}C]$

MRICDF Door Control Process



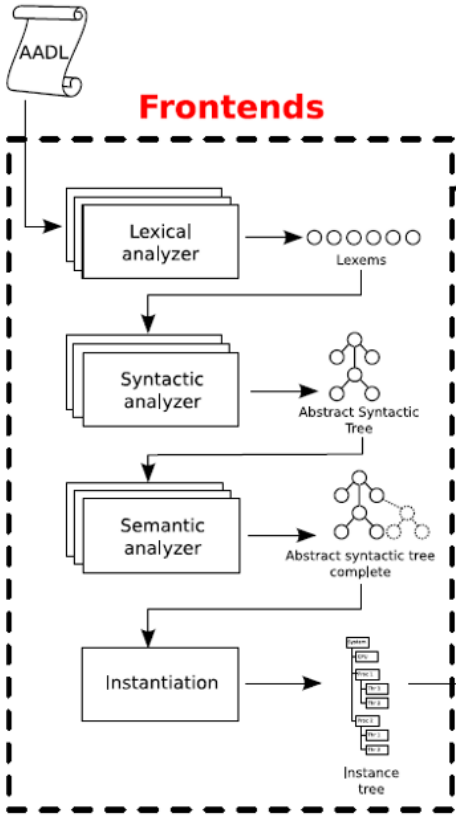
Associating Software Behavior with AADL

```
process controller
Features
        ...
    ...
end controller

process implementation controller.c
Properties
        ...
        source_name => "MRICDF_Controller";
        source_language => MRICDF;
        source_location => "../PATH";
end controller.c
```

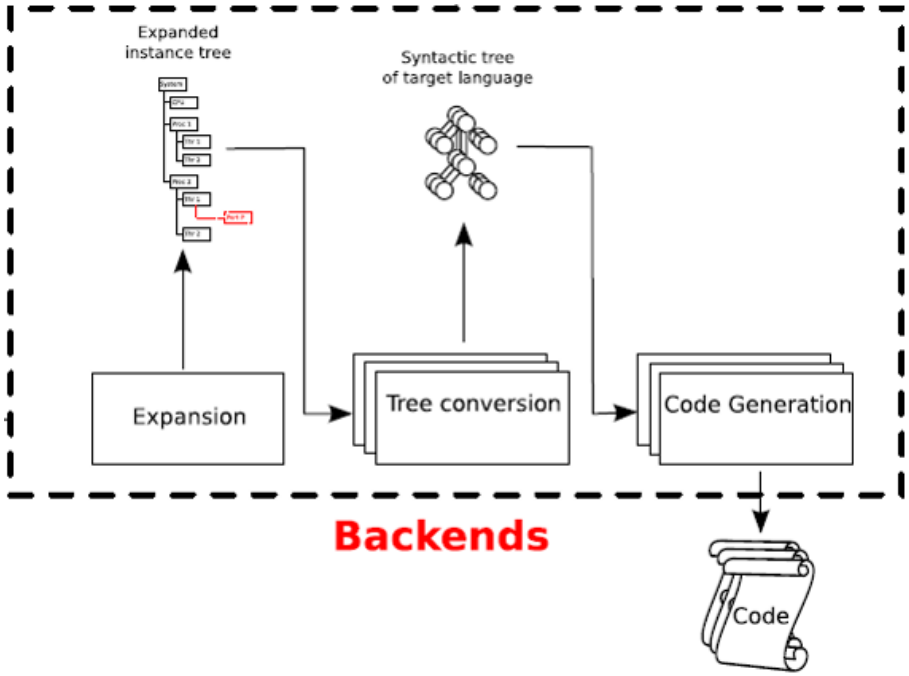
Ocarina Generation Frontend

- Frontend (AADL v 1.0 & v 2.0)
  - Parse AADL files
  - Syntactic Analysis
  - Semantic Analysis

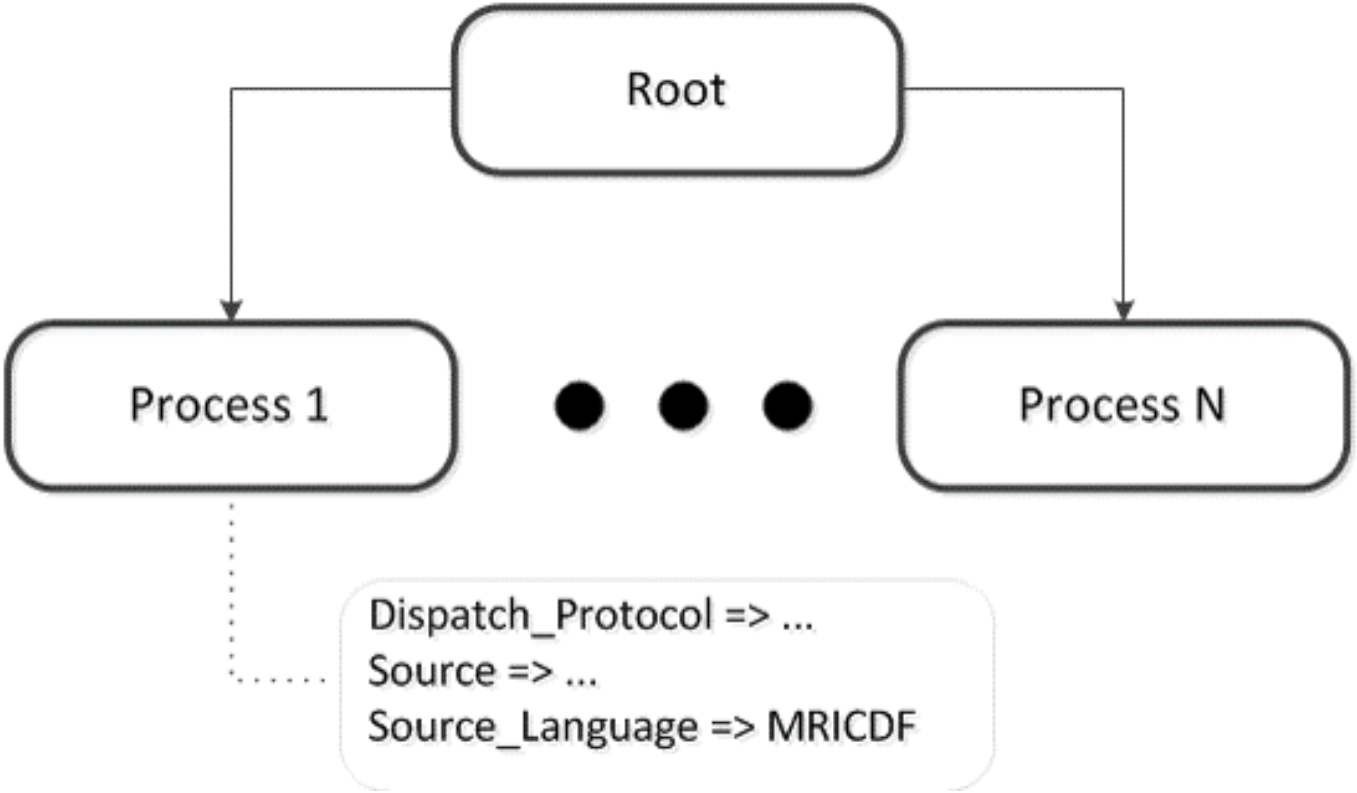


Ocarina Generation Backend

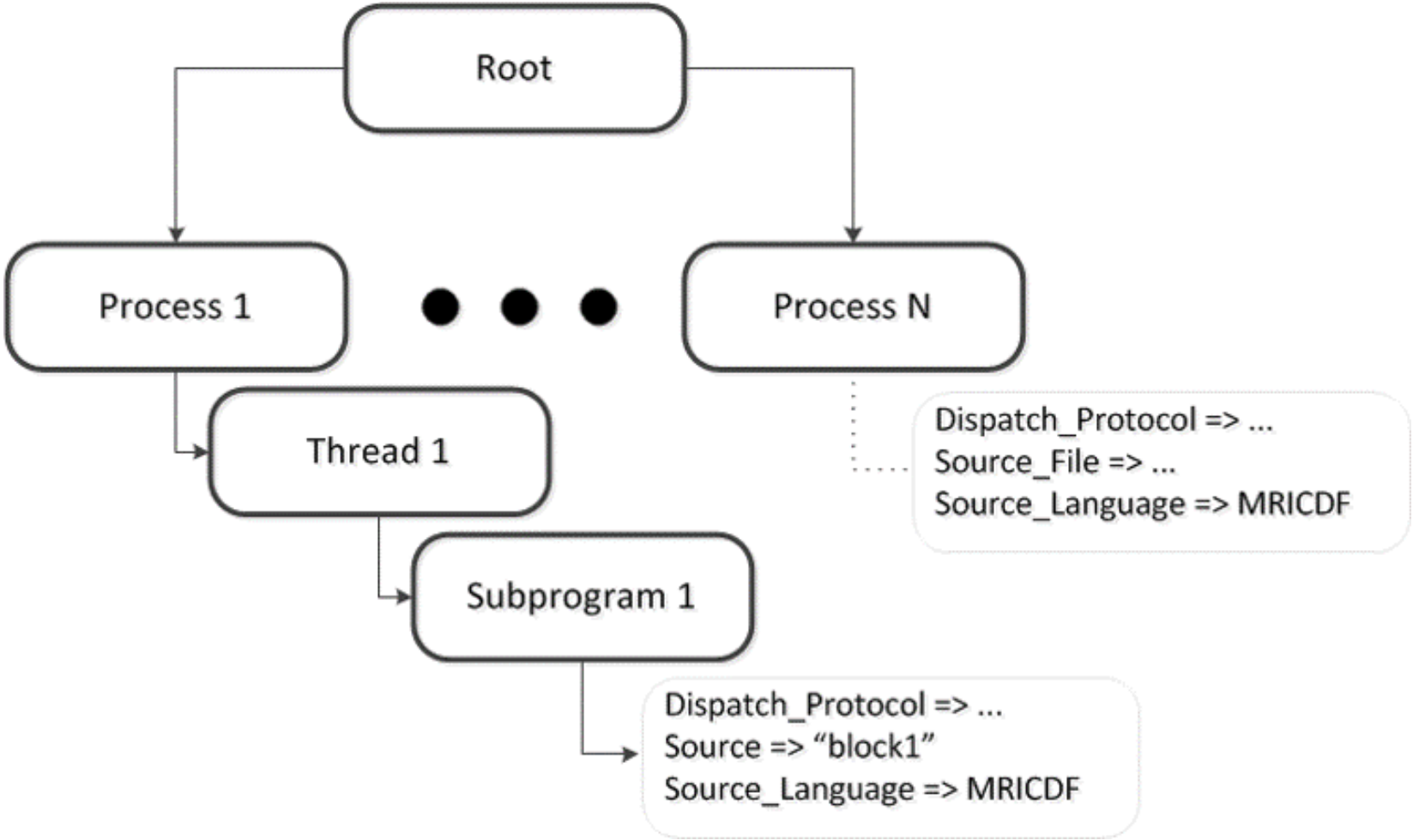
- Backend (Source Language)
  - Expansion
  - Intermediate Tree
  - Code Generation



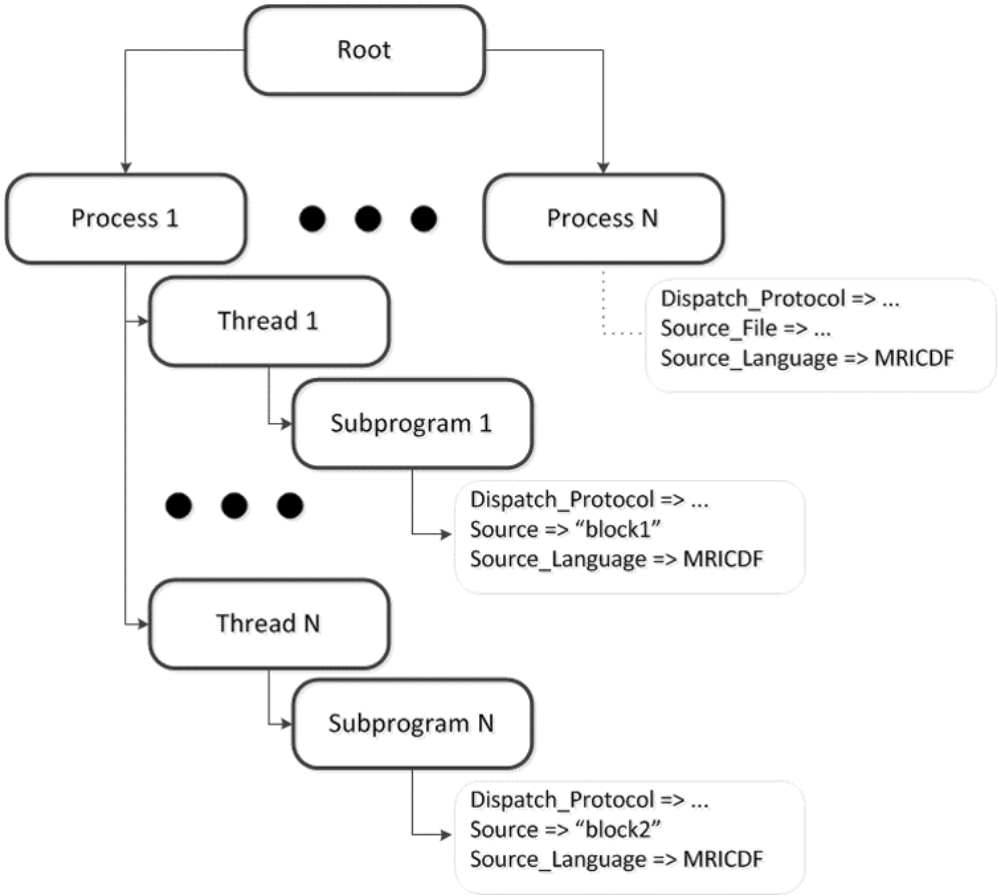
Dynamically Adding Threads



Dynamically Adding Threads Cont'd



Dynamically Adding Threads Cont'd



Presentation Outline

Status and Future Work

- An extension has been written for Ocarina to accept code generated by MRICDF
- A new code generation option has been added for dynamically parsing and adding threads.

Status and Future Work

- An extension has been written for Ocarina to accept code generated by MRICDF
- A new code generation option has been added for dynamically parsing and adding threads.
- Planned Additions:
  - Updates to MRICDF to streamline code generation
  - Inclusion of analysis for Real-time properties and code verification
  - Generate bus interface for distributed communication

Presentation Outline

Related AADL tools

A lot of work has been done on creating tools for AADL development:

- OSATE as an environment for textual and graphical developement
- Ocarina supports code generation from models
- Cheddar performs software schedulability analysis
- A number of model verification and analysis tools: COMPASS and BIP

AADL to Signal

"Toward polychronous analysis and validation for timed software architectures in AADL"

Y. Ma, H. Yu, T. Gautier, P. Le Guernic, J.P. Talpin, L. Besnard, and M. Heitz

- Translates an entire AADL model into Signal
- Formally analyze the translated model
- Simulate the behavior of the translated model

AADL to Signal

"Toward polychronous analysis and validation for timed software architectures in AADL"

Y. Ma, H. Yu, T. Gautier, P. Le Guernic, J.P. Talpin, L. Besnard, and M. Heitz

- Translates an entire AADL model into Signal
- Formally analyze the translated model
- Simulate the behavior of the translated model

Instead APECS:

- Acts as an extension of the AADL development environment
- Is aimed at generating executables for the targeted physical platform

Presentation Outline

Conclusion

- The APECS framework is based on the AADL development tools Osate / Ocarina and the polychronous tool EmCodeSyn
- APECS provides a environment for modeling end to end systems
- Software behavior is explicitly specified and formally verifiable
  - As opposed to informal specifications in C or Ada
  - Free from additional synchronization constrainsts of Esterel/Lustre
- We illustrate this with the Elevator System

Any Questions?