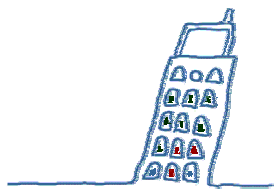
Automating mapping of functional blocks to real-time tasks

Bartolini Cesare

Scuola Superiore Sant'Anna



ERICSSON

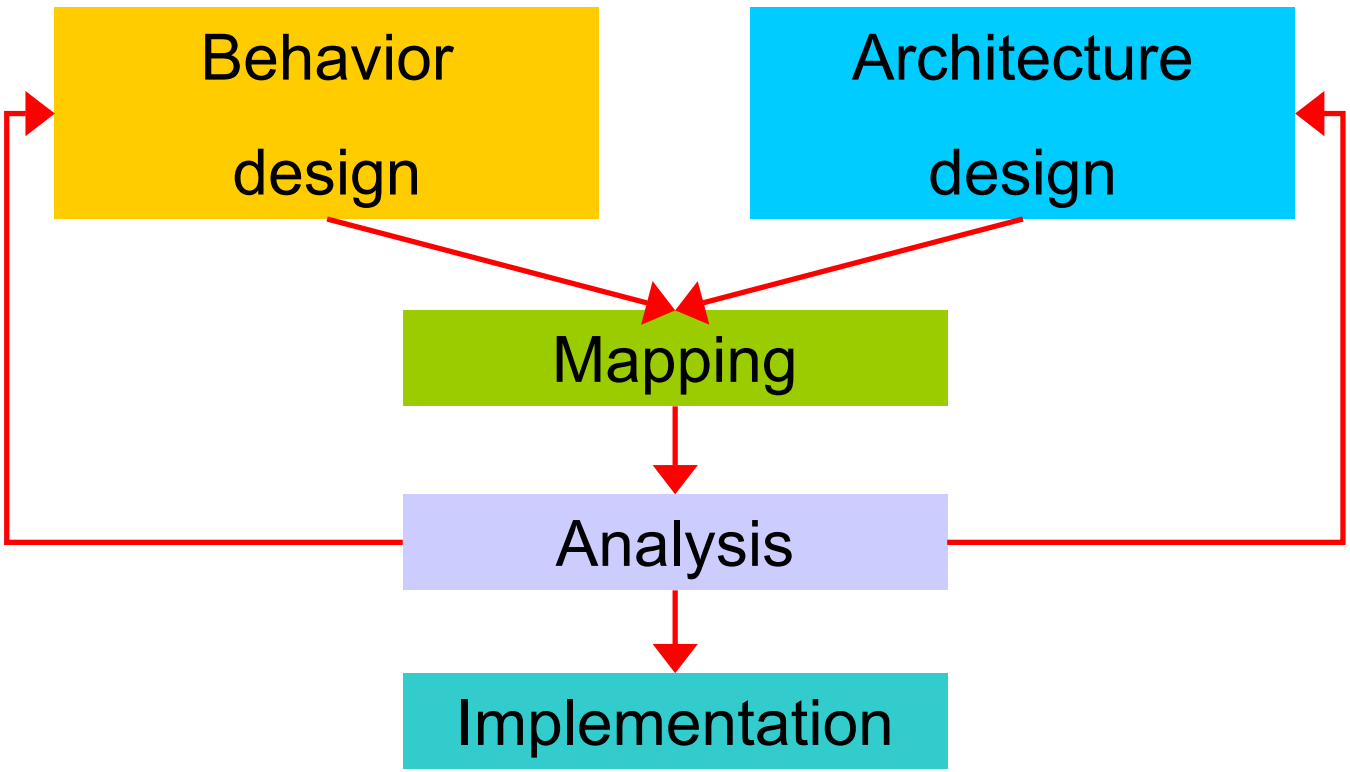
Embedded systems

- Low cost and time to market
  - Design should be fast and simple
  - Reutilization of components
- Efficiency
  - Real-time support
  - Resource management support

Real-time support

- Often, embedded systems are real-time
- Non-functional issues must be introduced
  - Scheduler
  - Tasks
  - Priorities
  - …
- “Non-functional” is referred to something the user need not know

Platform-based design



Design process

- Behavior
  - “What the system does”
  - Assumes infinite resources
  - Independent of underlying platform
- Architecture
  - “What the system can use top run the behavior”
  - Hardware: physical components
  - Software: scheduler, tasks
- Behavior and architecture are independent of each other

Design process (2)

- Mapping
  - The behavior is mapped over the architecture
  - Each behavior component is tied to an architecture component
    - This defines how that specific function is run
- Testing and simulation
  - Performance must be evaluated
    - The system scheduling must be feasible
    - Additional metrics: memory occupation, processor utilization

Addressed problem

- Mapping is done manually
  - Arbitrary choices
    - Task creation (mapping functionality to tasks)
    - Task properties
  - Scarce optimization
- An automated method might help
  - Finding optimal solution, if possible
  - Reducing the number of possible mappings

A single task

- The whole system might be a single task
  - Advantages
    - No context changes
    - No scheduling overhead
    - No task descriptors (low memory occupation)
  - Drawbacks
    - No concurrency
    - System deadlines cannot be addressed

One task per function

- Every function might be a task per se
  - Advantages
    - Total concurrency
    - Maximum flexibility and reactivity
  - Drawbacks
    - Too many context changes
    - Excessive scheduling overhead
    - Task descriptors become a burden

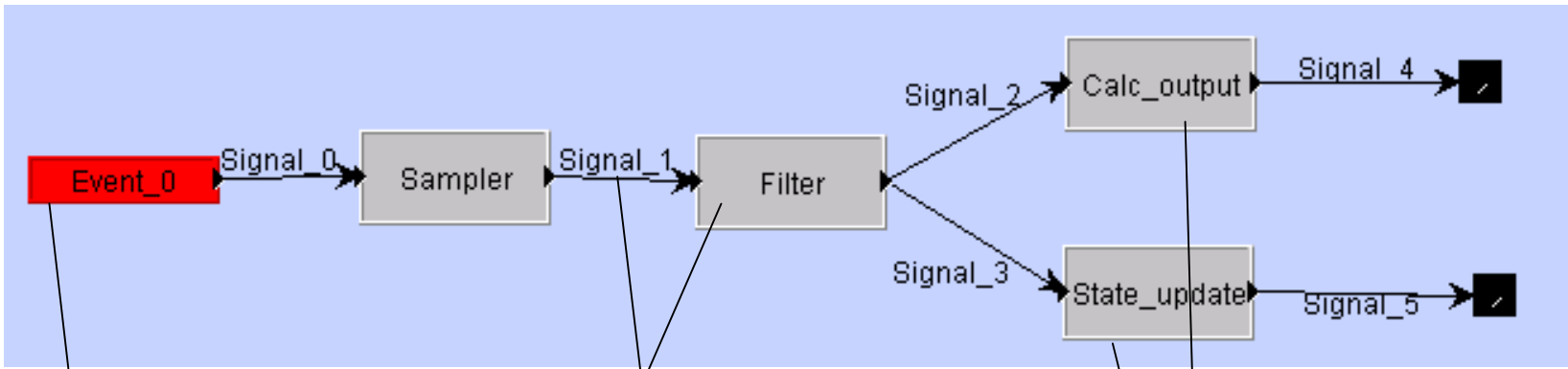
Goals of this research

- Define a method to automate at least part of the mapping process
  - Search for a suitable trade-off between the two extremes
  - The designer might review and modify it later
- Create a schedulable task set in a dynamic priority system
  - Fixed priorities might be used instead
- Allow the designer to focus on the behavior

Model of execution

- DAG (directed Acyclic Graph)
  - A vertex is a function (behavior component)
  - An edge is a communication between functions
    - Edges carry signals, data are not accounted for
    - The control path is referred to, not the data path
  - A *path* is an end-to-end sequence of edges and vertices
  - Asynchronous model
- The base element is the function, not the task

Behavior model



Function
When activated by a signal, it processes the input and then produces its output

Event
Signal generated by the external environment

Signal
Activates functions imposing precedence relations

Parallelism
These two functions might run concurrently

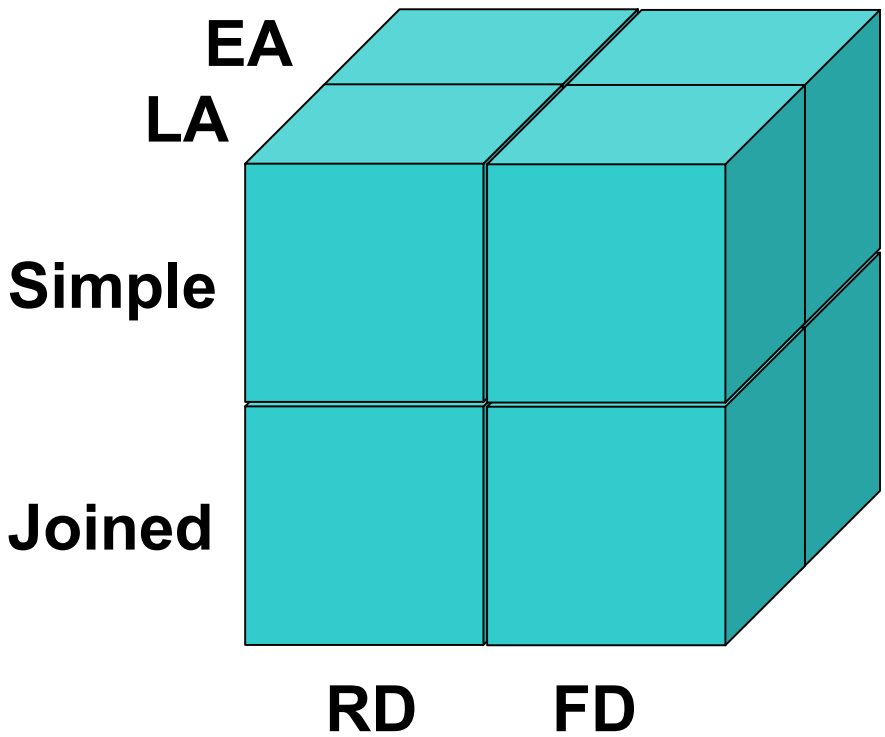
Automated mapping

- The objective is to generate a schedulable task set from the application structure
- End-to-end deadlines are required, whilst function WCETs are not
- A preemptive EDF scheduler is used

Three-step methodology

- Task partitioning: defines the set of tasks composing the system
  - Simple partitioning
  - Joined partitioning
- Deadline assignment: rules for assigning the task properties
  - Full deadlines
  - Rising deadlines
- Activation semantics: specifies when a task is activated
  - Early activation
  - Late activation

Possible combinations



Main problems

- Large task set
- The full methodology is heavy
  - Usage can be excessively burdening in an embedded system (large overhead)
- There are several degrees of simplification, with some performance loss
  - Small restrictions greatly reduce the overhead

Features

- Provides a basis for generating a task set
- Optimal solution
  - Feasibility
  - Preemptions
- Flexibility
  - Can be extended in any of the three dimensions
    - Actually, they are not orthogonal
- Feasibility tests are available

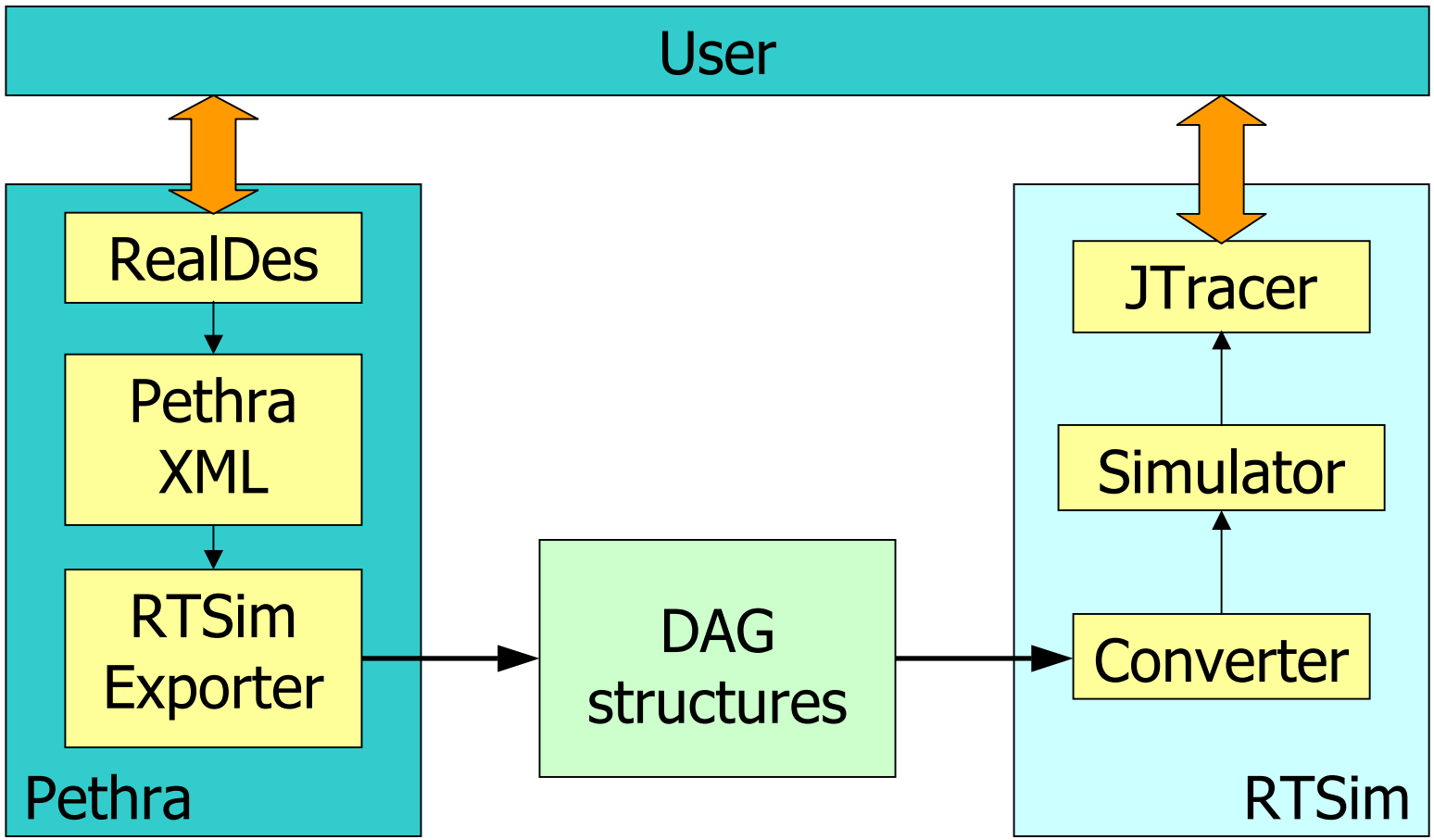
Tools

- Pethra
  - Development framework
- RealDes
  - Used to generate a directed acyclic graph (DAG)
- RTSim
  - Task set simulator

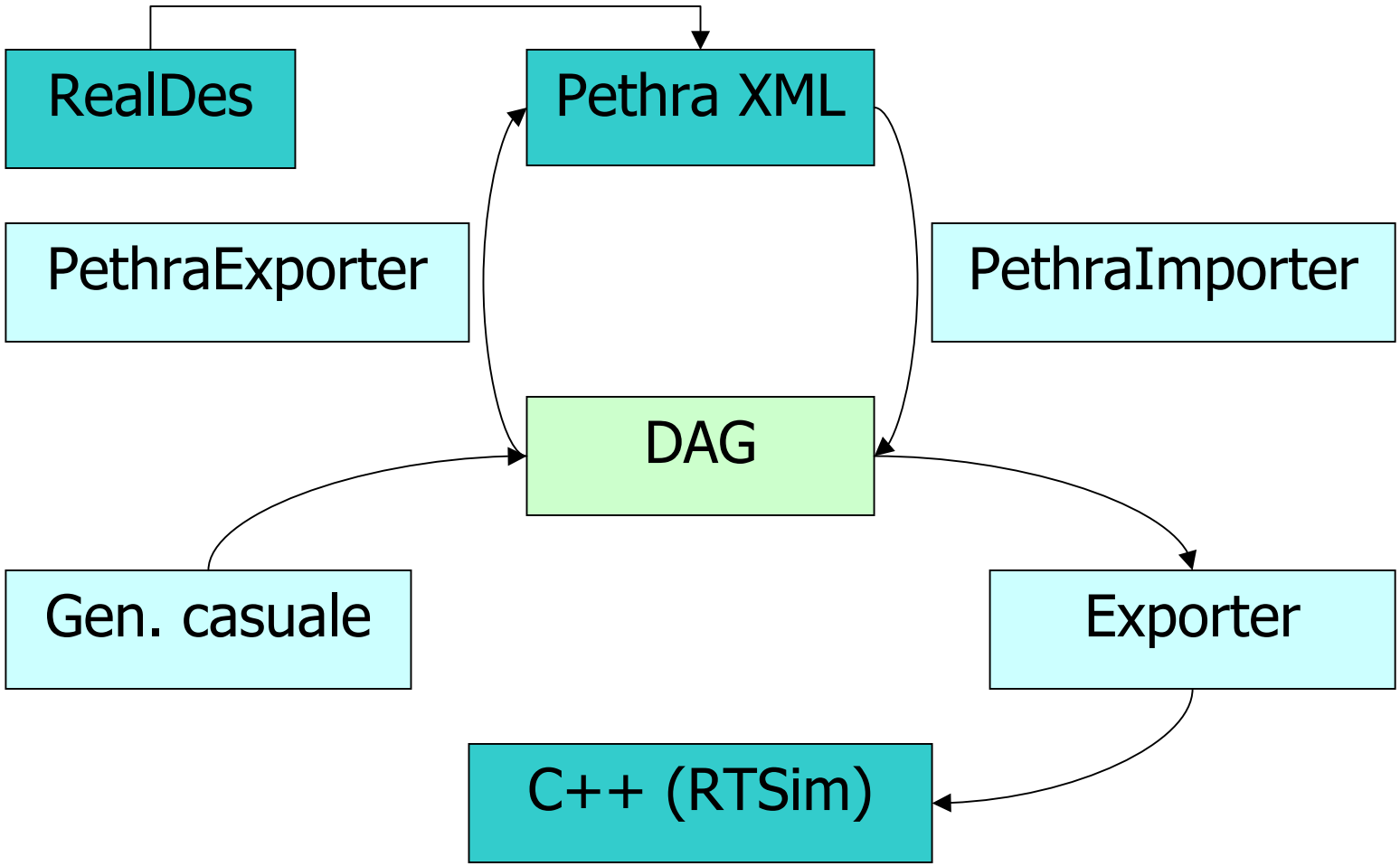
Current status

- Pethra module, used to export a DAG (created with RealDes)
- RTSim expansion, to convert the DAG into a task set
  - The task set properties must be selected
- Automatic random DAG generation
- Simulations, preliminary comparisons

Current status (2)



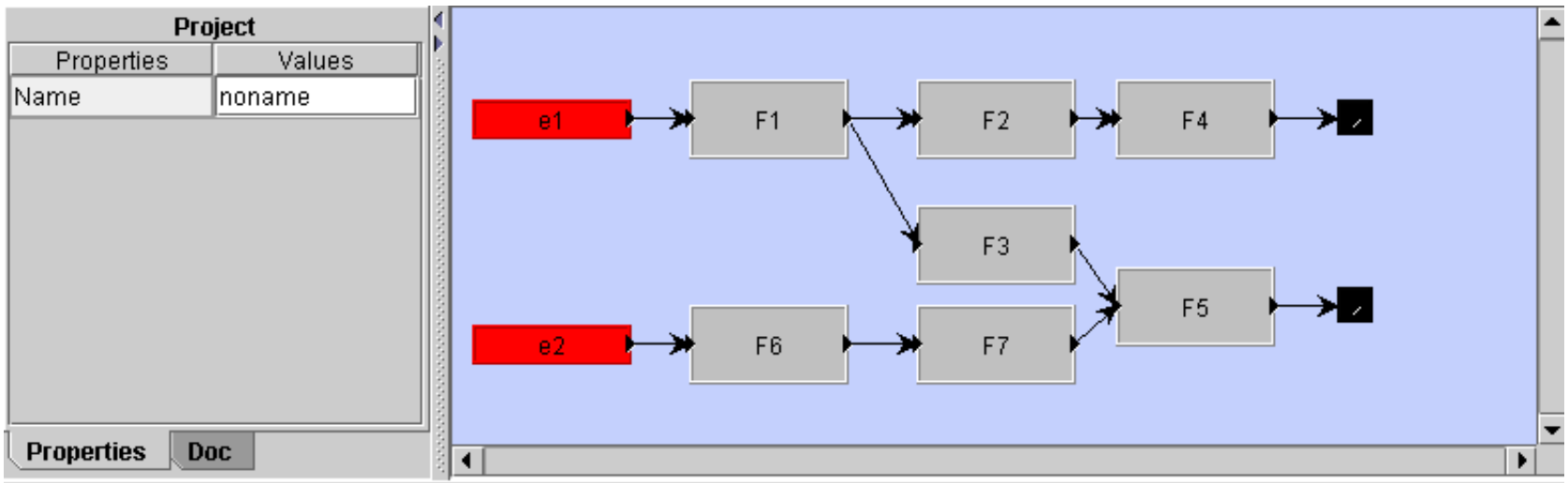
Work in Pethra



Work in RTSim

- Created task models suitable for the algorithms of the methodology
- Implementation of the three phases
  - Any combination can be used in simulation

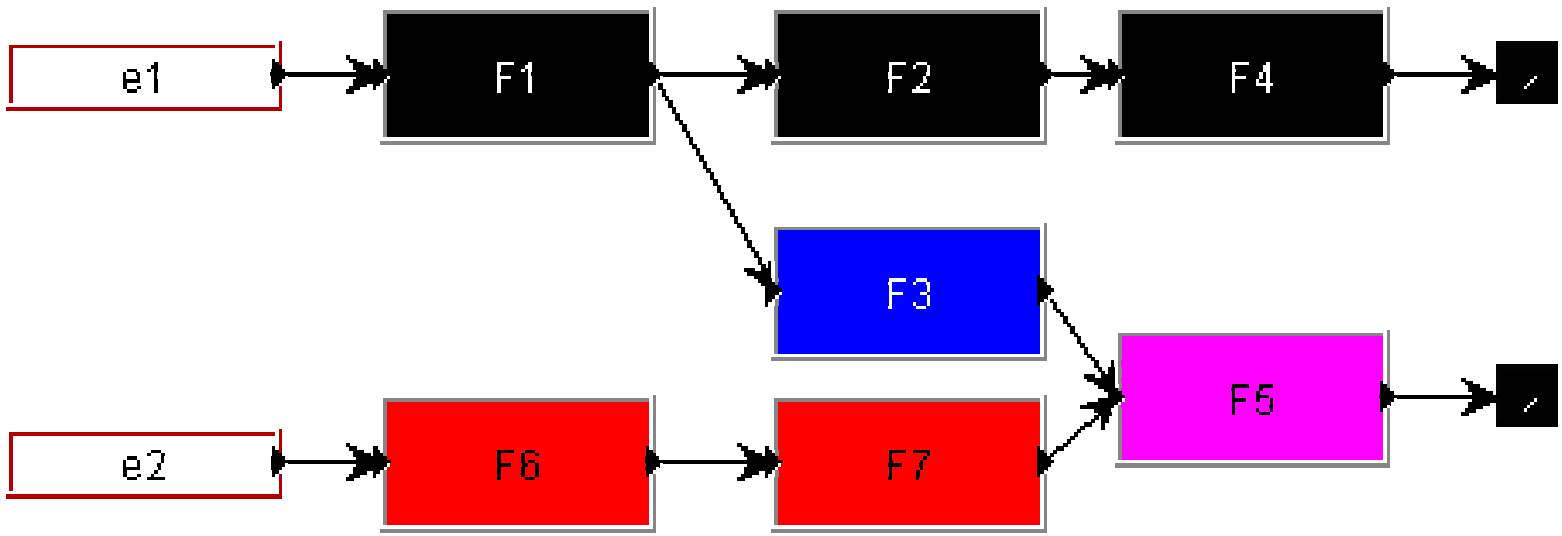
Example - Modeling



Paths | Tasks

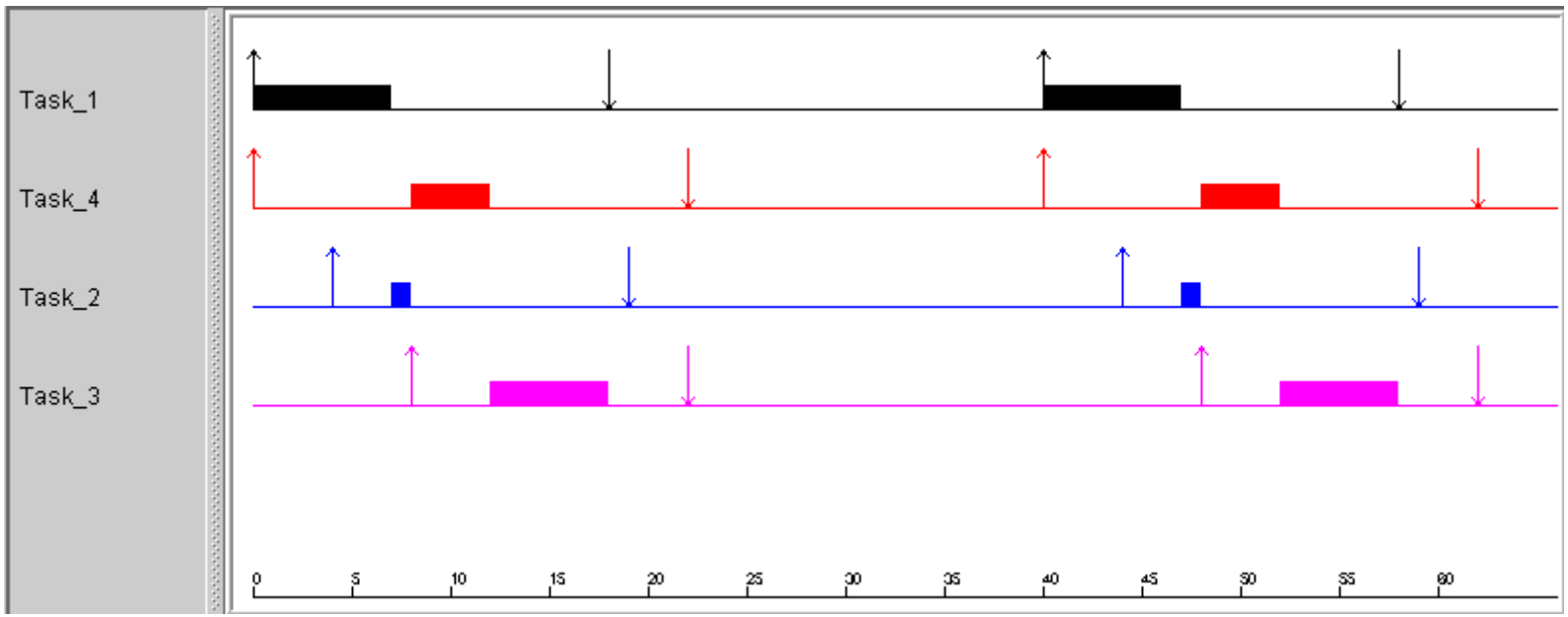
Name	Way	D.L.(ms)	Show
Path_0	[e1, , F1, , F2, , F4, , EndMarker_0]	18	
Path_1	[e1, , F1, , F3, , F5, , EndMarker_1]	22	
Path_2	[e2, , F6, , F7, , F5, , EndMarker_1]	25	

Example – Task set (Joined)

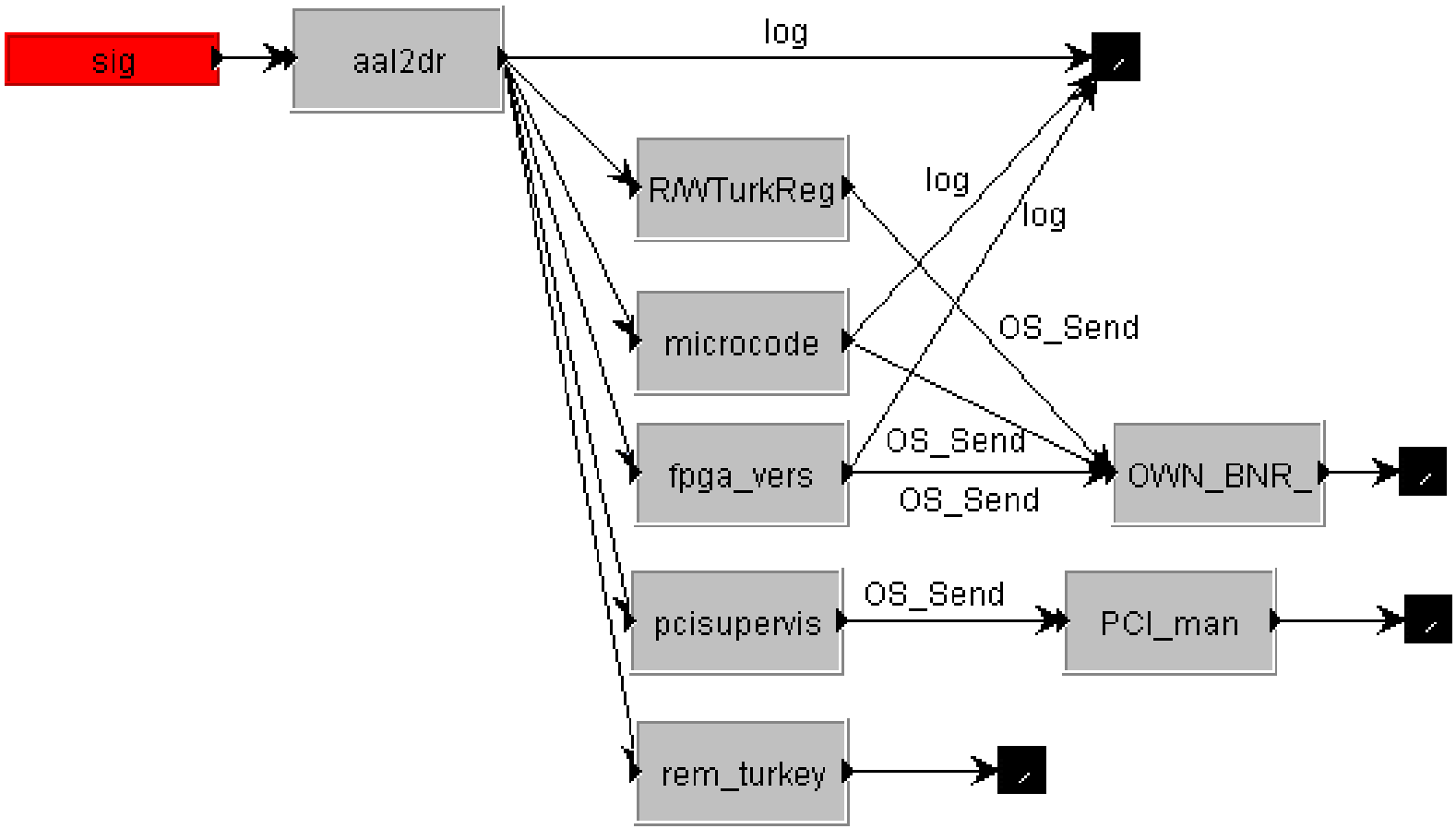


- Task 1
- Task 2
- Task 3
- Task 4

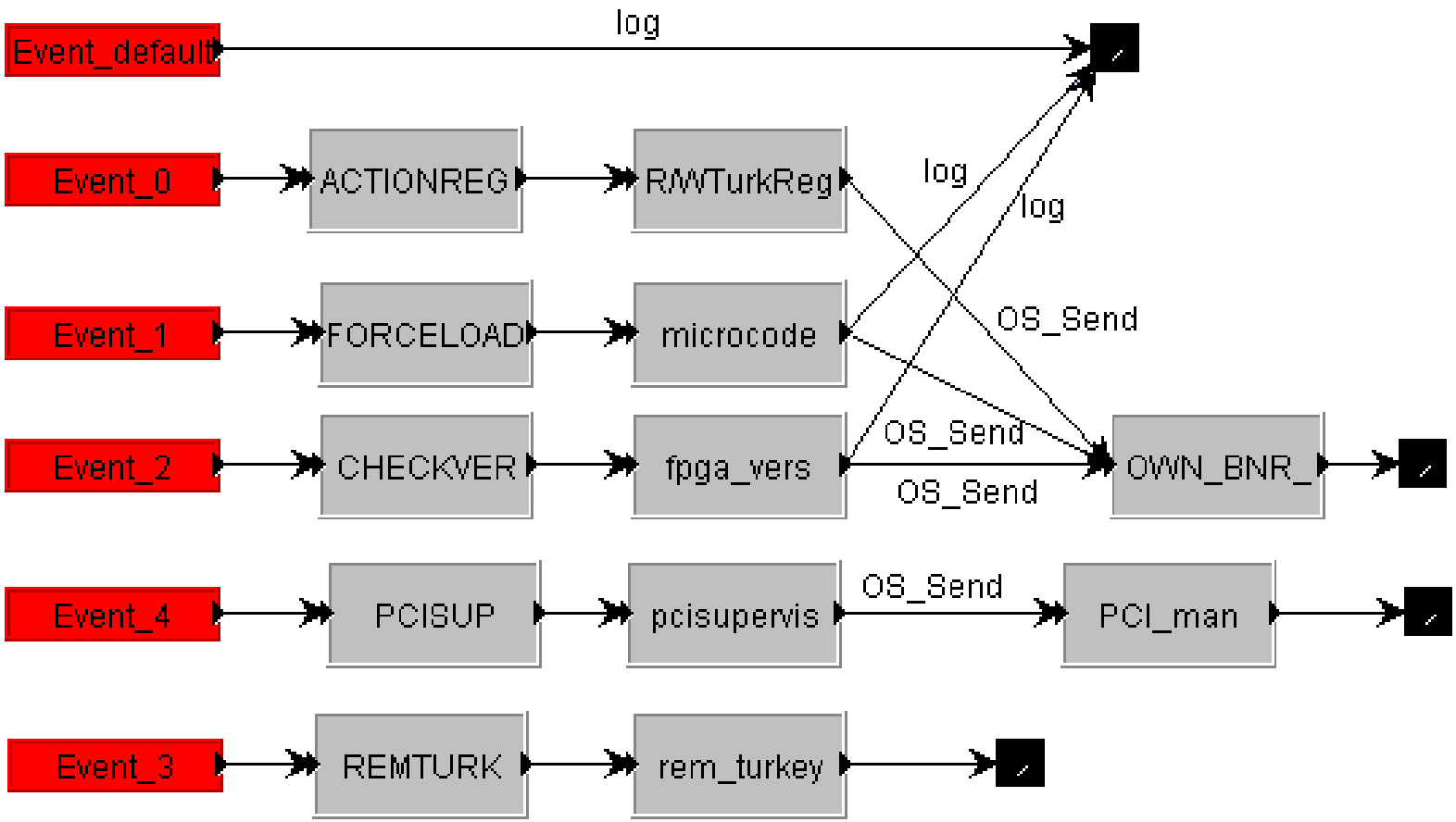
Example – Simulation (RD + LA)



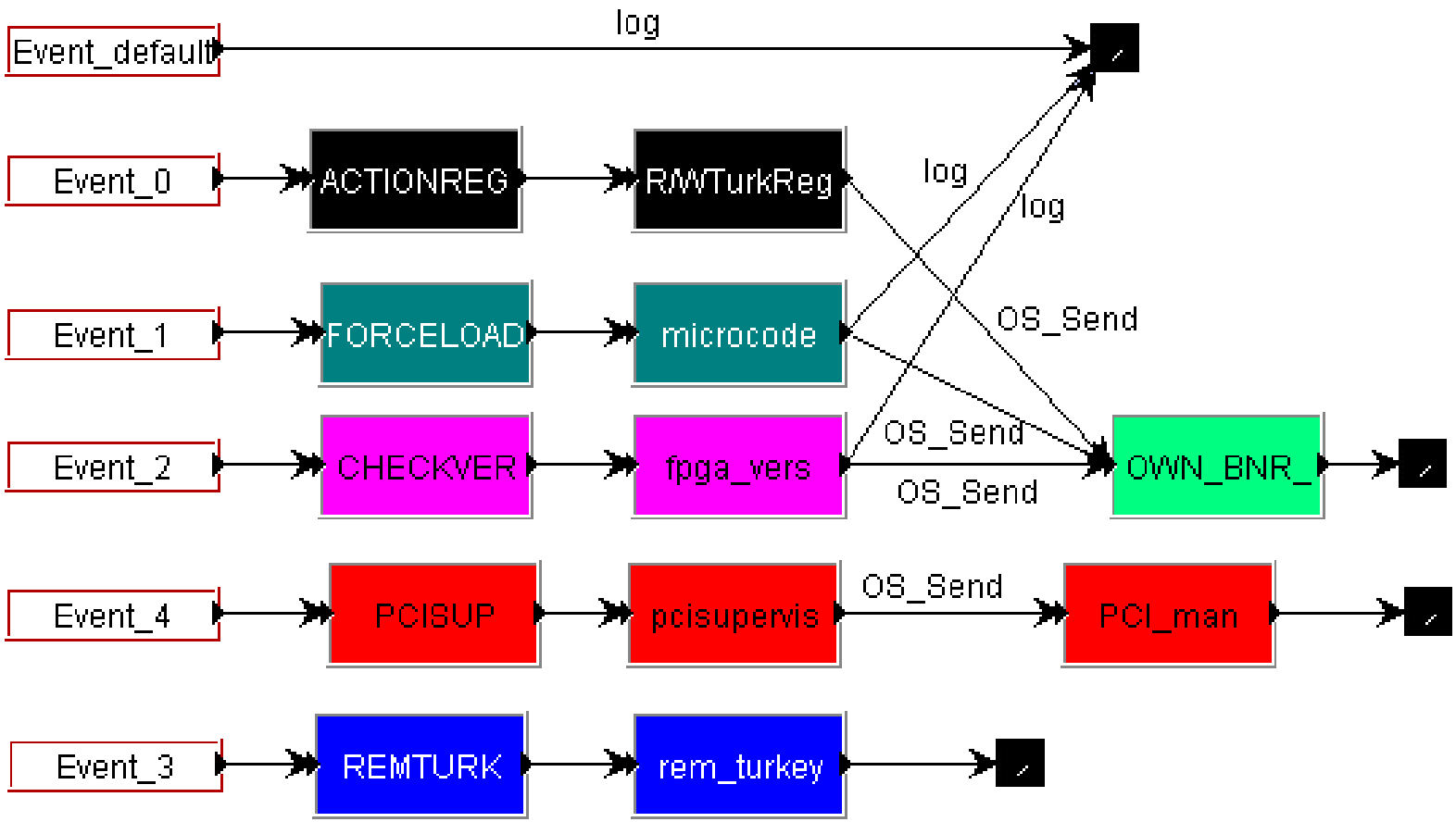
aal2dr


sig
aal2dr
log
RMWTurkReg
log
log
microcode
OS_Send
fpga_vers
OS_Send
OS_Send
OWN_BNR_
pcisupervis
OS_Send
PCI_man
rem_turkey

aal2dr (2)


Event_default
log
Event_0
ACTIONREG
RMWTurkReg
log
log
Event_1
FORCELOAD
microcode
OS_Send
Event_2
CHECKVER
fpga_vers
OS_Send
OS_Send
OWN_BNR_
Event_4
PCISUP
pcisupervis
OS_Send
PCI_man
Event_3
REMTURK
rem_turkey

aal2dr (3)


Event_default
log
Event_0
ACTIONREG
RMVTurkReg
log
log
Event_1
FORCELOAD
microcode
OS_Send
Event_2
CHECKVER
fpga_vers
OS_Send
OS_Send
OWN_BNR_
Event_4
PCISUP
pcisupervis
OS_Send
PCI_man
Event_3
REMTURK
rem_turkey

Conclusions and future work

- Reduce the task set
  - Different algorithms for task partitioning
  - Requires resource management policies (like PCP or SRP)
- Exploration of “and” semantics
- Extend the model of execution
  - RealDes must be extended, too
- XML
- Optimization (branch & bound, genetic algorithms)
- Advanced architectures
  - Multi-processor or distributed systems