Autonomic Management of Dynamically Partially Reconfigurable FPGA Architectures using Discrete Control

Xin An & Eric Rutten (INRIA),
Jean-Philippe Diguet & Nicolas Le Griguer (Lab-STICC),
Abdoulaye Gamatié (LIRMM)

June 26, 2013

Outline

Outline

Autonomic computing on reconfigurable hardware

Controlling FPGAs as autonomic computing

- Field Programmable Gate Arrays (FPGAs)
- dynamically partially reconfigurable (DPR) FPGAs

Control techniques to design the MAPE-K loops

- model possible behaviours, and control objectives, separately
- classically continuous time dynamics and differential equations
- discrete control, events and states, Petri nets or automata

Discrete control for autonomic FPGAs

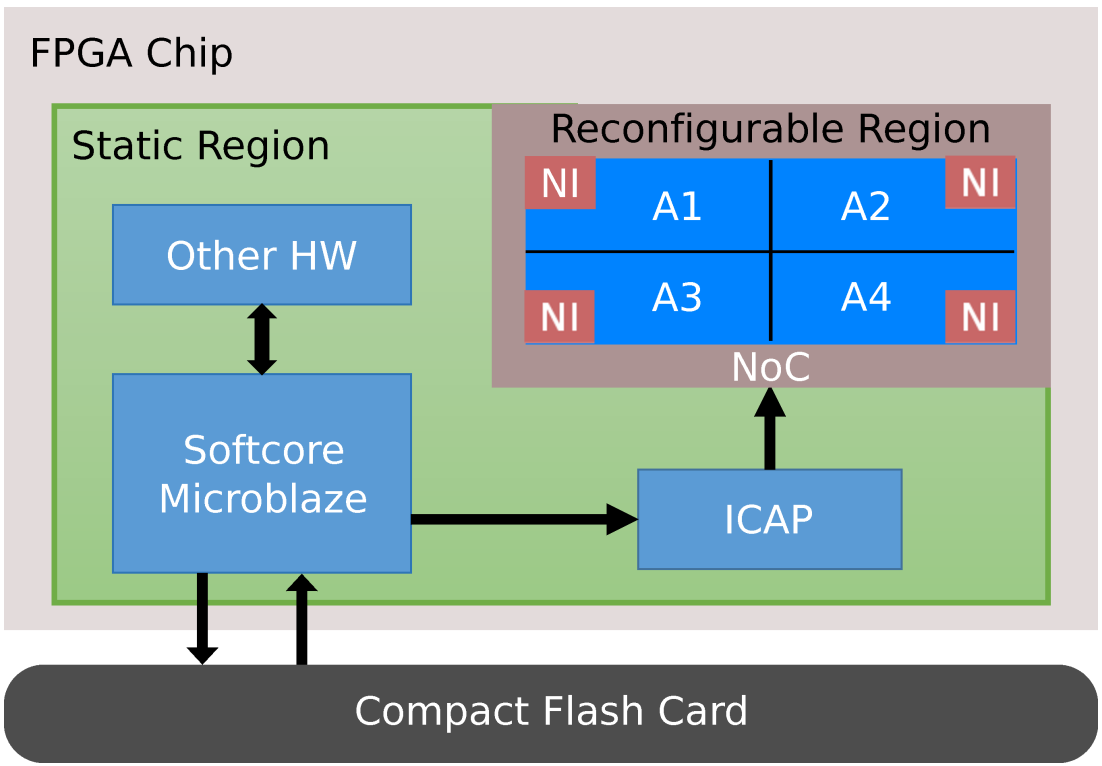
- systematic modelling framework
  application, tasks implementations, architecture
- Automata & Discrete Controller Synthesis (DCS)

Outline

FPGA-based architectures

Basic reconfigurable cell : programmable by bitstream

- Run-time partial reconfiguration : DPR FPGAs
- slower than ASICs, much faster than GP CPUs
- Management of reconfiguration : loading chosen bitstream



Reactive languages, synchronous programming

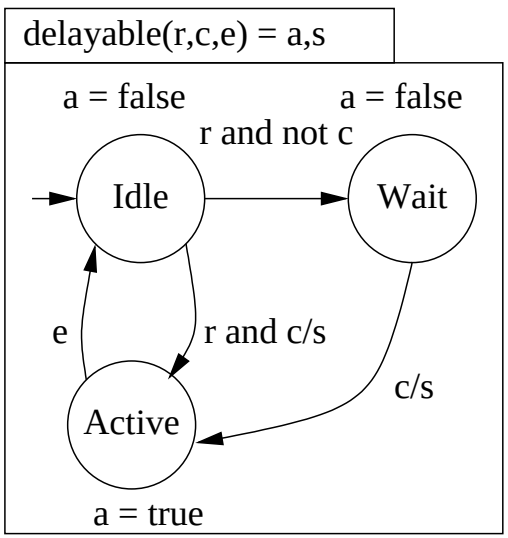
Modelling formalism **and** programming language

- reaction to input flows → output flows
- data-flow nodes and equations ; mode automata (FSM)
- parallel (synchronous) and hierarchical composition

synchronous languages, (25+ years)

tools: compilers (e.g., Heptagon), code generation, verification, ...

example: delayable task control (in Heptagon)




```
node delayable(r,c,e:bool) returns (a,s:bool)
let automaton
state Idle do
   a = false; s = r and c
  until r and c then Active
      | r and not c then Wait
state Wait do a = false; s = c
  until c then Active
state Active do a = true; s=false
  until e then Idle
end tel
```

Discrete controller synthesis (DCS): principle

Goal

Enforcing a temporal property Φ on a system
on which Φ does not yet hold a priori

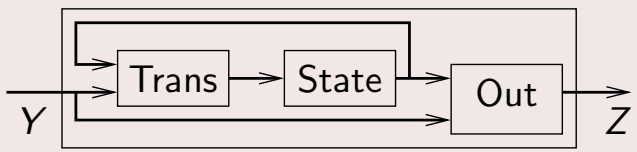
Discrete controller synthesis (DCS): principle

Goal

Enforcing a temporal property Φ on a system
on which Φ does not yet hold a priori

Principle (on implicit equational representation)

State	memory
Trans	transition function
Out	output function



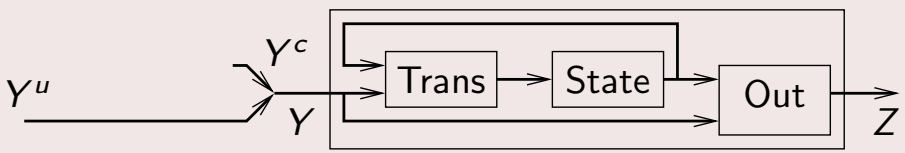
Discrete controller synthesis (DCS): principle

Goal

Enforcing a temporal property Φ on a system
on which Φ does not yet hold a priori

Principle (on implicit equational representation)

State	memory
Trans	transition function
Out	output function



- Partition of variables : controllable (Y^c), uncontrollable (Y^u)

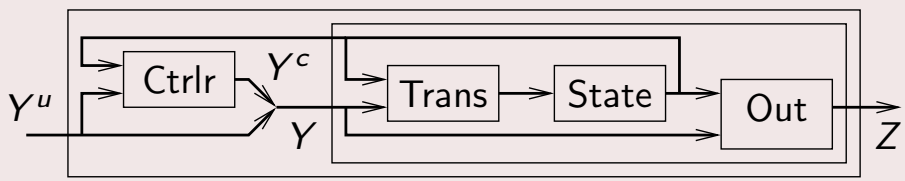
Discrete controller synthesis (DCS): principle

Goal

Enforcing a temporal property Φ on a system
on which Φ does not yet hold a priori

Principle (on implicit equational representation)

State	memory
Trans	transition function
Out	output function

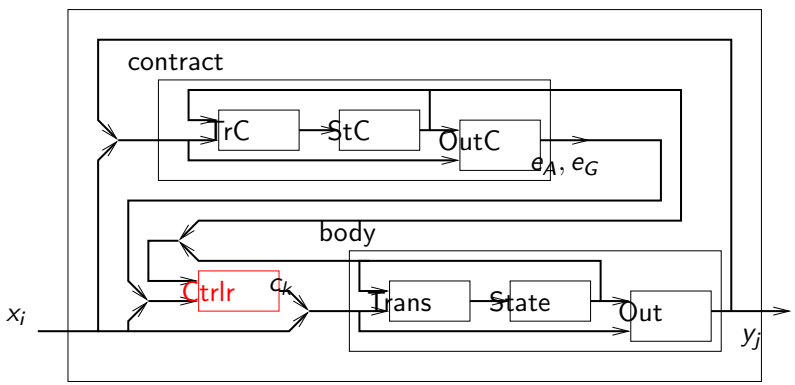


- Partition of variables : controllable (Y^c), uncontrollable (Y^u)
- Computation of a controller such that the controlled system satisfies Φ (invariance, reachability, attractivity, ...)

DCS tool: Sigali (H. Marchand e.a.)

BZR programming language [http://bzr.inria.fr]

- to each contract, associate controllable variables, local
- at compile-time (user-friendly DCS),
  compute a controller for each component
- *step* and *reset* functions ; executable code : C, Java, ...



$\texttt{twotasks}(r_1, e_1, r_2, e_2) = a_1, s_1, a_2, s_2$

enforce not $(a_1$ and $a_2)$

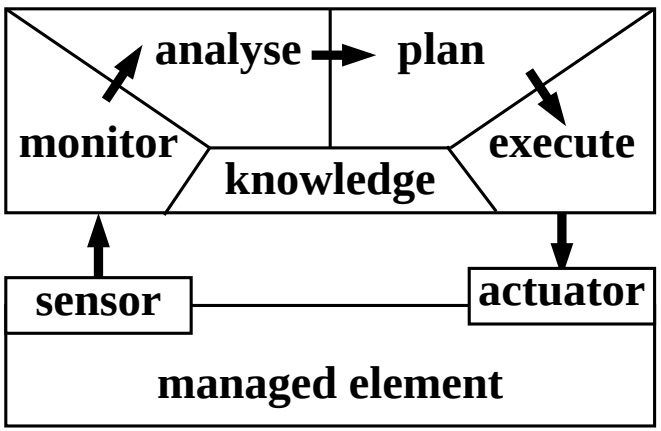
with c_1, c_2

$(a_1, s_1) = \texttt{delayable}(r_1, c_1, e_1);$

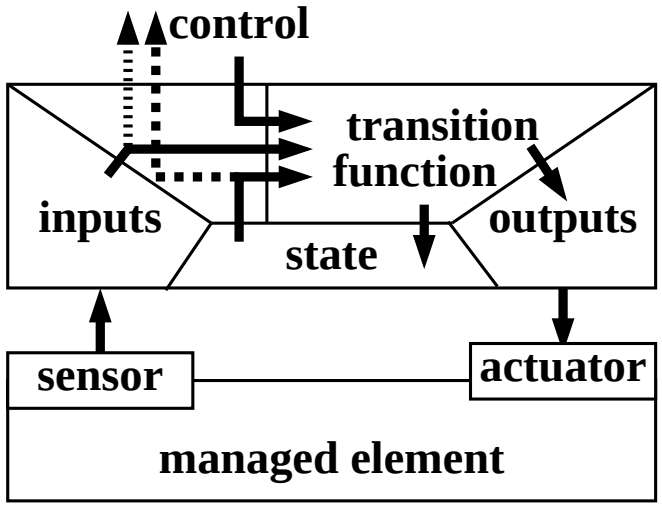
$(a_2, s_2) = \texttt{delayable}(r_2, c_2, e_2)$

& G. Delaval & H. Marchand [ACM LCTES'10] [jDEDS13]

Discrete control as MAPE-K

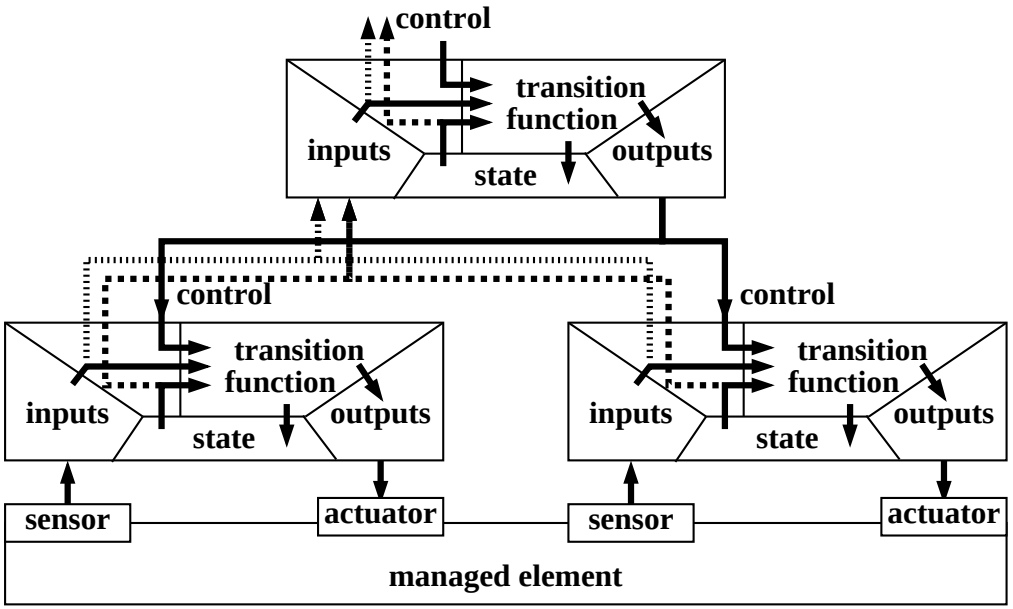


- autonomic MAPE-K
- flows : sensor observations to reconfiguration actions
- reactive language BZR used as DSL for decision



- FSM instanciation of MAPE-K
- exhibit state (observability)
- accept events or conditions (controllability)

Hierarchical architecture



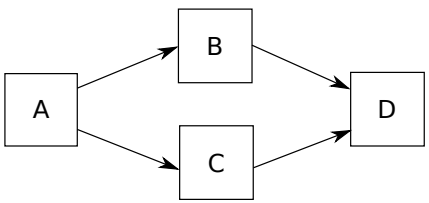
- hierarchical MAPE-K, through additional interfaces
- in components : composites using life-cycles of subcomponents
- implementation : *step*
  - synthesized and generated off-line
  - called at run-time in composite controller

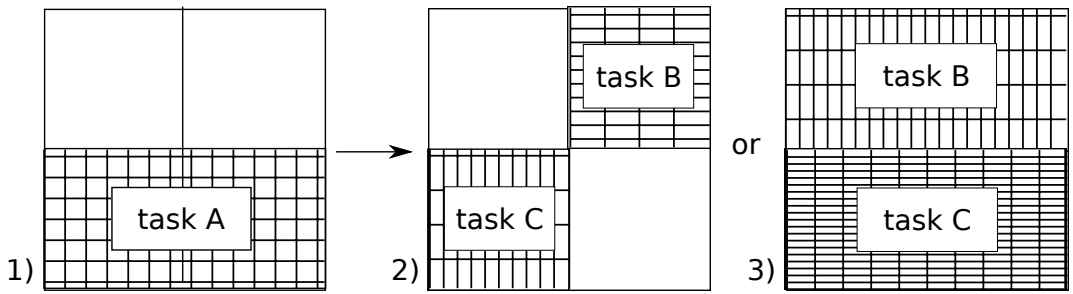
Outline

Considered DPR FPGAs

System architecture

- hardware : reconfigurable area divided into four tiles: A1 – A4
- battery (load levels)
- task implementations : used tiles, WCET, power peak
- application software : directed, acyclic graph (DAG) of tasks





Reconfiguration choices

- unused tile Ai can be put into sleep mode
- between task implementations : from 1) to **either** 2) or 3)

Reconfiguration policy

Separation of concerns, favoring re-use and variations

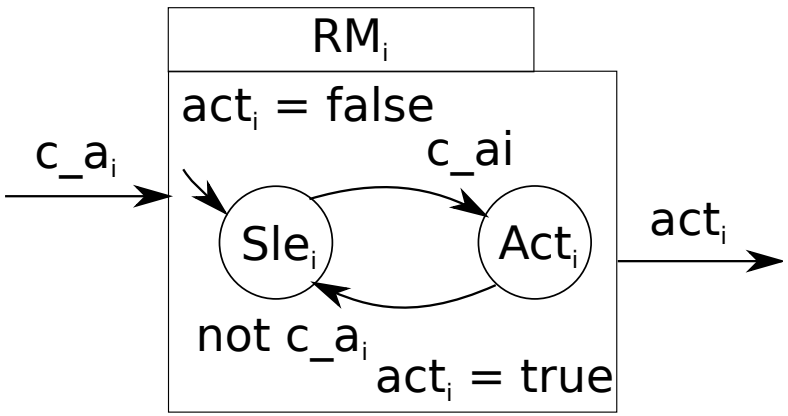
- description of possible behaviors
- specification of coordination policy

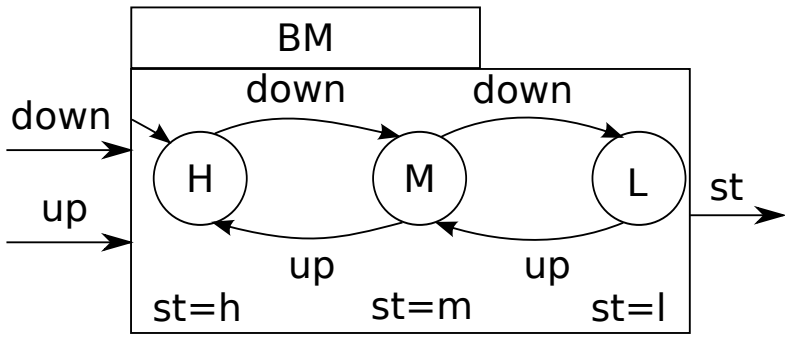
System objectives

1. resource usage constraint:
   exclusive uses of reconfigurable areas $A1$-$A4$;
2. energy reduction constraint:
   switch areas to sleep mode when executing no task;
3. power peak constraint : stay below bound
   constrained w.r.t battery levels;
4. minimise power peak of hardware platform

Outline

Architecture behaviour





Architecture

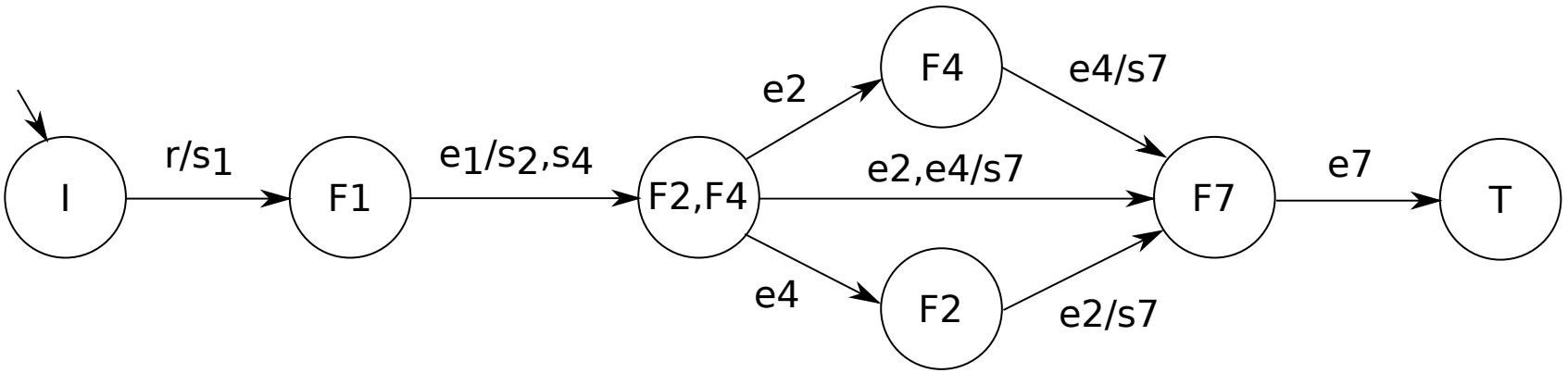
- four reconfigurable tiles A1, A2, A3, A4
  - modes : active, sleep
  - controllable switches

Battery

observer

- states H (high), M (medium) and L (low)
- input from battery sensor

Application behaviour

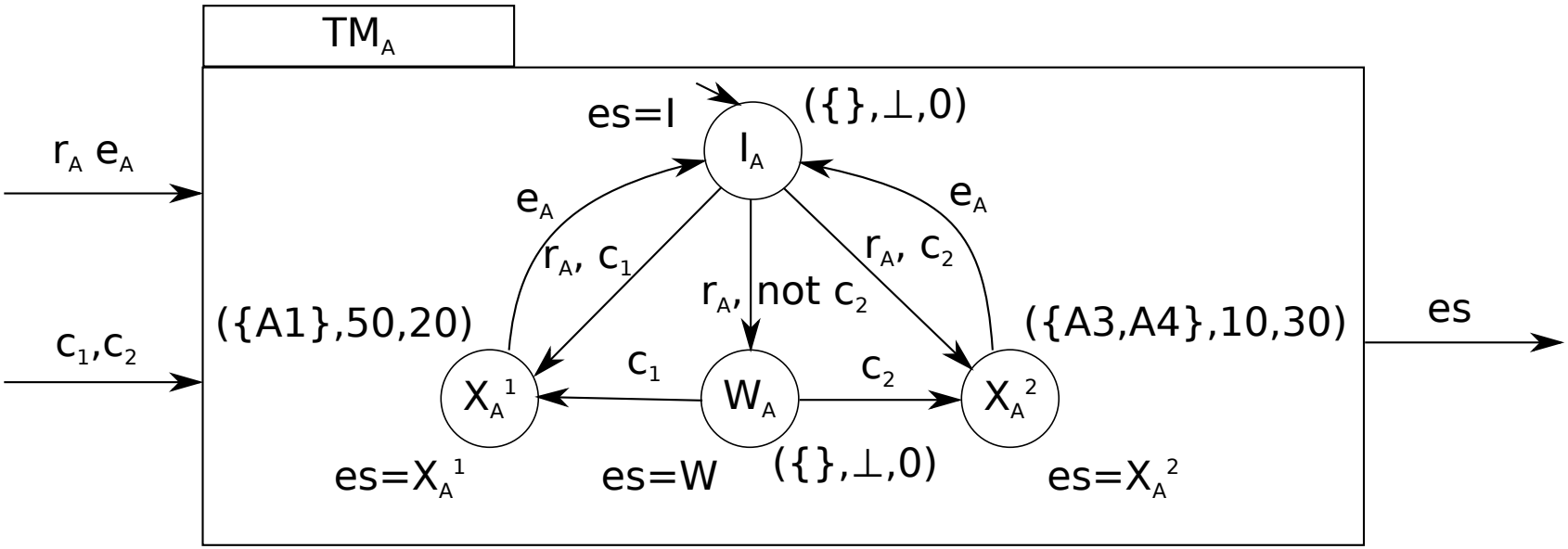


Follows the DAG of tasks

systematic construction of model from DAG

- inputs : request, task finish notifications
- outputs : start of tasks
- states : currently active tasks

Task execution behaviour



Example for task T_A

- 2 implementations
- waiting state (delayed)
- control on delay, and on implementation choice

Local costs on states

- $rs \in 2^{RA}$: used tiles (architecture resources)
- wt : WCET
- pp : power peak

Global system behaviour model

Parallel composition of control models

$$\mathcal{S} = RM_1|...|RM_4|BM|TM_A|...|TM_D|Sdl$$

- reconfigurable tiles RM_1-RM_4
- battery BM
- scheduler Sdl (from DAG)
- tasks TM_A-TM_D

Global costs, on global state $q = (q_1, ..., q_n)$

defined from the local ones

- used resources: union of used resources associated with the local states, i.e., $rs(q) = \bigcup rs(q_i), 1 \leq i \leq n$;
- power peak: the sum of values associated with the local states, i.e., $pp(q) = \sum(pp(q_i), 1 \leq i \leq n)$;

System objectives

Enforce invariance w.r.t. subset of states where :

(1) exclusive uses of reconfigurable tiles by tasks:
$\forall q_i, q_j \in q, i \neq j$, that $rs(q_i) \bigcap rs(q_j) = \emptyset$;

(2') switch tile A_i to sleep mode, when executing no task:
$\nexists q_i \in q, Ai \in rs(q_i) \Rightarrow act_i = false$;

(2'') switch tile A_i to active mode when executing task(s):
$\exists q_i \in q, Ai \in rs(q_i) \Rightarrow act_i = true$;

(3) battery-level constrained power peak (thresholds P_0, P_1, P_2):
$pp(q) < P_0$ (resp. P_1 and P_2)

Experimental validation

- video processing system, ML605 board from Xilinx with FPGA
- BZR used to encode models and objectives,
  and generate C code, integrated in runtime manager

Outline

Conclusions & perspectives

DPR FPGAs as autonomic computing

- systematic modeling framework for DPR FPGA
- LTSs and automatic controller generation using DCS
- experimental validation

Perspectives

- Domain-Specific Language (DSL)
  automated generation of models, patterns
  integration in design flow, executable code
- richer model, e.g. reconfiguration costs
- DCS : optimal, modular, quantitative, distributed controllers
- more complete experiment ongoing on video FPGA