Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC-PapersOnLine 48-7 (2015) 132–138

Towards Applying Logico-numerical Control to Dynamically Partially Reconfigurable Architectures

Nicolas Berthier * **Xin An** ** **Hervé Marchand** *

* *INRIA Rennes - Bretagne Atlantique, Rennes, France*
** *Hefei University of Technology, Hefei, China*

Abstract: We investigate the opportunities given by recent developments in the context of Discrete Controller Synthesis algorithms for infinite, logico-numerical systems. To this end, we focus on models employed in previous work for the management of dynamically partially reconfigurable hardware architectures. We extend these models with logico-numerical features to illustrate new modeling possibilities, and carry out some benchmarks to evaluate the feasibility of the approach on such models.

Keywords: Discrete Controller Synthesis, Infinite Systems, Control of Computing, Synchronous Languages, Hardware Architectures, Dynamically Partially Reconfigurable FPGA

1. INTRODUCTION

Recent proposals by Berthier and Marchand (2014) in the domain of symbolic Discrete Controller Synthesis (DCS) techniques have led to the development of a tool capable of handling *logico-numerical systems and properties*, i.e., involving state variables defined on infinite domains. The handling of such infinite systems opens the way to new opportunities for modeling and control, that still need to be investigated.

We extend real-life models proposed by An et al. (2013a,b) for the management of Dynamically Partially Reconfigurable (DPR) hardware architectures to: (i) assess the feasibility of the proposal on bigger systems, (ii) perform some performance evaluations of the new tool ReaX on realistic models, including for control objectives newly implemented in this tool; and (iii) introduce logico-numerical features in the model to assess that the approach can still be applied using models involving quantitative aspects.

DPR Hardware Architectures DPR hardware architectures, typically Field Programmable Gate Arrays (FPGAs) (Lysaght et al., 2006), have been identified as a promising solution for the design of energy-efficient embedded systems (Hinkelmann et al., 2009). However, such solutions have not been extensively exploited in practice for two main reasons: i) the design effort is extremely high and strongly depends on the available chip and tool versions, and ii) the simulation process, which is already complex for non-reconfigurable systems, is prohibitively large for reconfigurable architectures. Therefore, new adequate methods to deal with their correct dynamical reconfiguration are required to fully exploit their potential.

Dynamical reconfiguration management requires choosing new configurations depending on the history of events occurring in the system and predictive knowledge about possible outcomes of reconfigurations. Such decision-making component is difficult to design because of the combinatorics of possible choices, the transversal constraints between them, and even more, the history aspects. The work we present advocates the application of DCS techniques to fulfill this control problem.

Related Works The reconfiguration management in DPR technologies is usually addressed by using manual encoding and analysis techniques that are tedious and error-prone according to Göhringer et al. (2008). Other existing approaches dedicated to self-management of adaptive or reconfigurable systems use heuristics and machine learning techniques (Sironi et al., 2010; Paulsson et al., 2006; Jovanović et al., 2008) for instance. Maggio et al. (2012) discuss some approaches applying standard control techniques such as Proportional Integral and Derivative (PID) controller or Petri nets-based control. The same kind of control has also been used for processor and bandwidth allocation in servers (Lu et al., 2002). Eustache and Diguet (2008) applied close-loop control to select hardware/software configurations on an FPGA with a configuration control based on a data-flow model and diffusion mechanisms. We note that such a solution relies on heuristics and empirical laws that prevent instability and select the suitable configurations.

Compared to the above reconfiguration control techniques, major advantages of the discrete control approach considered by An et al. (2013a,b) are the enabled formal correctness and guarantees on run-time performance, as well as the possibility to synthesize the controller automatically.

Outline We first present in Section 2 the modeling formalism we use for expressing the reconfiguration problem, as well as the tools involved in our work. Sections 3 detail the problem of reconfiguration control for FPGA-based DPR systems. We expose the modeling and formulation as a DCS problem, as well as an illustrative logico-numerical extension of the model in Section 4, and we report on our performance evaluation experiments in Section 5.

10.1016/j.ifacol.2015.06.484

2. MODELING FORMALISM AND TOOLS

2.1 Arithmetic Symbolic Transition Systems

The model of Arithmetic Symbolic Transition Systems (ASTSs) is a transition system with (internal or input) variables whose domain can be infinite, and composed of a finite set of symbolic transitions. Each transition is guarded on the system variables, and has an update function indicating the variable changes when a transition is fired. This model allows the representation of infinite systems whenever the variables take their values in an infinite domain, while it has a finite structure and offers a compact way to specify systems handling data.

Let $V = \langle v_1, \ldots, v_n \rangle$ be a tuple of variables and $\mathcal{D}_v$ the (infinite) domain of v. We note $\mathcal{D}_V = \prod_{i\in[1,n]} \mathcal{D}_{v_i}$ the (infinite) domain of V. $v_i(V)$ gives the value of variable v_i in vector V.

Definition 1. (Arithmetic Symbolic Transition System). *An ASTS is a tuple $S = \langle X, I, T, A, \Theta_0 \rangle$ where:*

- *$X = \langle x_1, \ldots, x_n \rangle$ is a vector of state variables ranging over $\mathcal{D}_X = \prod_{j\in[1,n]} \mathcal{D}_{x_j}$ and encoding the memory necessary for describing the system behavior;*
- *$I = \langle i_1, \ldots, i_m \rangle$ is a vector of variables that ranges over $\mathcal{D}_I = \prod_{j\in[1,m]} \mathcal{D}_{i_j}$, called input variables;*
- *T is of the form $(x_i' := T^{x_i})_{x_i\in X}$, such that, for each $x_i \in X$, the right-hand side T^{x_i} of the assignment $x_i' := T^{x_i}$ is an expression on $X \cup I$. T is called the transition function of S, and encodes the evolution of the state variable x_i. It characterizes the dynamic of the system between the current state and the next state when receiving an input vector.*
- *A is a predicate with variables in $X \cup I$ encoding an assertion on the possible values of the inputs depending on the current state;*
- *Θ_0 is a predicate with variables in X encoding the set of initial states.*

For technical reasons, we shall assume that A is expressed in a theory that is closed under quantifier elimination as for example the Presburger arithmetic.

ASTSs can conveniently be represented as parallel compositions of Mealy automata with numerical variables and explicit locations or in its symbolic form.

Let us consider the following example ASTS where $X = \langle \xi, x, o \rangle$, $I = \langle a, i \rangle$ with $\mathcal{D}_X = \{\mathsf{F}, \mathsf{G}\} \times \mathbb{Z} \times \mathbb{B}$, $\mathcal{D}_I = \mathbb{B} \times \mathbb{Z}$

$$T = \begin{cases} \xi' := & \mathsf{G} & \text{if } (\xi = \mathsf{F} \wedge a \wedge x \geq 0), \\ & \mathsf{F} & \text{if } (\xi = \mathsf{G} \wedge i > 42), \xi \text{ otherwise} \\ x' := & 2x+1 & \text{if } (\xi = \mathsf{F} \wedge a \wedge x \geq 0), \\ & i & \text{if } (\xi = \mathsf{G} \wedge i \leq 42), x \text{ otherwise} \\ o' := & (\xi = \mathsf{F} \wedge a \wedge x \geq 0) \vee (\xi = \mathsf{G} \wedge i > 42) \end{cases}$$

$A(\langle \xi, x, o, a, i \rangle) = (\xi = \mathsf{G} \wedge 3x + 2i \leq 41 \wedge a)$
$\Theta_0(\langle \xi, x, o \rangle) = (\xi = \mathsf{F} \wedge x = 0)$

The corresponding Mealy automaton with explicit locations (leaving A aside) can be represented as in Figure 1.

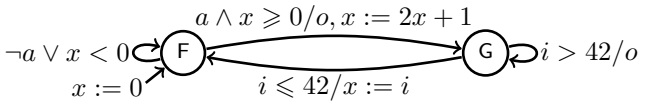

Fig. 1. Example ASTS as a Mealy automaton.

Remark 1. Observe that the variable o is actually an output *of the system, although it belongs to the vector of* state *variables. Indeed, we do not distinguish between those two kinds of variables to keep the ASTS models simple. We can characterize output variables as the ones that never appear in the right hand side of the assignments in T.*

Remark 2. We qualify as logico-numerical *an ASTS whose state (and non-output) and input variables are Boolean variables ($\mathbb{B}$) or numerical variables (typically, $\mathbb{R}$ or $\mathbb{Z}$), i.e., such that $X = \mathbb{B}^k \cup \mathbb{R}^{k'} \cup \mathbb{Z}^{k''}$ with $k + k' + k'' = n$ (and similarly for the input variables). ASTSs with only Boolean non-output state variables are called* finite.

To each ASTS, one can make correspond an Infinite Transition System (ITS) defined as follows:

Given an ASTS $S = \langle X, I, T, A, \Theta_0 \rangle$, we make correspond an ITS $[S] = \langle \mathcal{X}, \mathcal{I}, \mathcal{T}_S, \mathcal{A}_S, \mathcal{X}_0 \rangle$ where:

- $\mathcal{X} = \mathcal{D}_X$ is the state space of $[S]$;
- $\mathcal{I} = \mathcal{D}_I$ is the input space of $[S]$;
- $\mathcal{T}_S \subseteq \mathcal{X} \times \mathcal{I} \to \mathcal{X}$ is such that
  $\mathcal{T}_S(x, \nu) = (x_j')_{j\in[1,n]} \Leftrightarrow \forall j \in [1, n], x_j' := T^{x_j}(x, \nu)$;
- $\mathcal{A}_S \subseteq \mathcal{X} \times \mathcal{I}$ is such that
  $\mathcal{A}_S = \{(x, \nu) \in \mathcal{X} \times \mathcal{I} | A(x, \nu) = \mathit{true}\}$;
- $\mathcal{X}_0 \subseteq \mathcal{X}$ is the set of initial states, and is such that
  $\mathcal{X}_0 = \{x \in \mathcal{X} | \Theta_0(x) = \mathit{true}\}$.

The behavior of such a system is as follows. $[S]$ starts in a state $x_0 \in \mathcal{X}_0$. Assuming that $[S]$ is in a state $x \in \mathcal{X}$, then upon the reception of an input $\nu \in \mathcal{I}$ such that $(x, \nu) \in \mathcal{A}_S$, $[S]$ evolves in the state $x' = \mathcal{T}_S(x, \nu)$. We denote $\mathit{XTrace}([S])$ the set of states that can be reached in $[S]$. Given an ASTS S and a predicate Φ over X, we say that S satisfies Φ (noted $S \models \Phi$) whenever $\mathit{XTrace}([S]) \subseteq \{x \in \mathcal{X} | \Phi(x) = \mathit{true}\}$.

Control of an ASTS Assume given a system S and a predicate Φ on S. Our aim is to restrict the behavior of S by means of control in order to fulfill Φ. We distinguish between the uncontrollable input variables U which are defined by the environment, and the controllable input variables C which are defined/restricted by the controller of the system. For technical reason, we assume that the controllable variables are Boolean. Note that the partitioning of the input variables in S induces a "partitioning" of the input space in $[S]$, so we have $\mathcal{I} = \mathcal{D}_U \times \mathcal{D}_C$. A controller is then given by a predicate A_Φ over $X \cup U \cup C$ that constrains the set of admissible (Boolean) controllable inputs so that the traces of the controlled system always satisfy Φ.

Definition 2. (Discrete Controller Synthesis Problem). *Given an ASTS $S = \langle X, U \cup C, T, A, \Theta_0 \rangle$ and a predicate Φ over X, solving the* discrete controller synthesis problem *is to compute a predicate A_Φ such that*

$$S' = \langle X, U \cup C, T, A_\Phi, \Theta_0 \rangle \models \Phi$$

and $\forall v \in X \cup U \cup C, A_\Phi(v) \Rightarrow A(v)$.

The general control problem that we want to solve is undecidable. In (Berthier and Marchand, 2014), we then used abstract interpretation techniques to ensure, at the price of some over-approximations, that the computation