Syntax and semantics of the Alpha0 subset of Alpha (Version 2)

May 30, 2006

1 Introduction

Alpha0 was introduced by C. Dezan in her Phd thesis ([**?**]). It was conceived as the lowest level of Alpha or equivalently as a subset of Alpha with a structural interpretation. The main weakness of this subset came from the fact that Alpha programs were not structured. Since then, the structuring of Alpha was provided by F. Dupont [**?**] and the structured version of Alpha0 was studied by P. Le Moënner [**?**]. Alphard is now intended to be this subset of Alpha with structural interpretation, from which we can translate Alpha into VHDL or other RTL description languages. However, during the transformation path from Alpha to Alphard, the automatic structuring is a complicated process, hence it appear that the Alpha0 format is still necessary as an intermediate format with all the structural information but in an unstructured form[1]. The current document describes the second version of the Alpha0 format,

2 Definition of Alpha0

In this section we recall the format Alpha0 introduced in [**?**]. Then we introduce the additionnal notions of version 2.

2.1 Original Alpha0

An Alpha program is in Alpha0 form if it meets the three following conditions:

1. there exits for all declared domains (except for the domains of inputs and outputs variables) an interpretation function for the indices (see [**?**] for precise definition). Each index is either a temporal index or a spatial index.

[1] il faudra unitliser deux mots different pour structural et structuring

2. each variable is a *signal* that has, for each spatial interpretation, the temporal interpretation $\{t \mid t >= 0\}$

3. The equations of the system define the outputs and local variables. Each operator involved in the equations has a structural interpretation. These equations are of two types: data equations and control equations.

   - Data equations define the different signals of the program. They are composed of the following operators:
     - pointwise operators represent the corresponding combinatorial operators.
     - restriction are used to restrict the spatial interpretation of indices.
     - dependencies are temporal dependencies representing registers or spatial dependencies representing connections.
     - case expressions are spatial case allowing to gather several connections
   - control equations allow signals to be constructed from input variables and the behavior to be controled. Such equations use the following constructions:
     - input dependencies correspond to relations between formal input and variables of the programs indicating where and when input are sent.
     - output dependencies indicate where and when output results are valid.
     - case control (temporal case?) define control signals from boolean constant (in a *finite automate* fashion).

2.2 Alpha0 version 2

As a platform to Alphard, we need to remove the second point of the previous definition, namely the fact that all temporal domains are of the form $\{t \mid t >=\}$. The other basic principles remain the same with slight differences. Spatial cases contain no dependencies, connections are represented using a simple space dependency (no case). Temporal cases are always nested in a spatial case. Control initialization equations are called *control* equation, the *Mirror* equations are the equations affecting inputs, output and control signal to local variables (hence, control signal become data signals). All these descriptions are summarized by the following definition:

1. There exits for all declared domains (except for the domains of inputs and outputs variables) an interpretation function for the indices (see [**?**]

for precise definition). Each index is either a temporal index or a spatial index.

2. The equations of the system define the output and local variables. Each operator involved in the equations has a structural interpretation. These equations are of four types: data equations, connection equations, control equations and mirror equations.

   - Data equations define the different signals of the program which are necessarily local variables. They are composed of the following operators:
     - Pointwise operators represent the corresponding combinatorial operators.
     - Restriction are used to restrict the spatial interpretation of indices.
     - Dependencies are temporal dependencies representing registers or identity dependencies representing connections between two signal inside one cell.
     - Cases are spatial case allowing several signals to be gathered.
     - The `if` operator (with nested `case` in each branches) is interpreted as a multiplexer.
   - Connection equations are limited to the following form: a single spatial dependency between two signals (`A=B.(t,p->t,p-1)` for instance).
   - Control equations allow control signals to be initialized. They define signals which are only temporal (no spatial indices).
   - Mirror equations: equation between input variables of the system and local variable of the system (only for interface). equation limited to an affine function applied to a variable (`A[t,p]=a[f(t,p)]` for input mirror equation and `b[i,j]=B[f(i,j)]` for output mirror equations)..