Programmable Models of Computation for a Component–Based Approach to Heterogeneous Embedded Systems

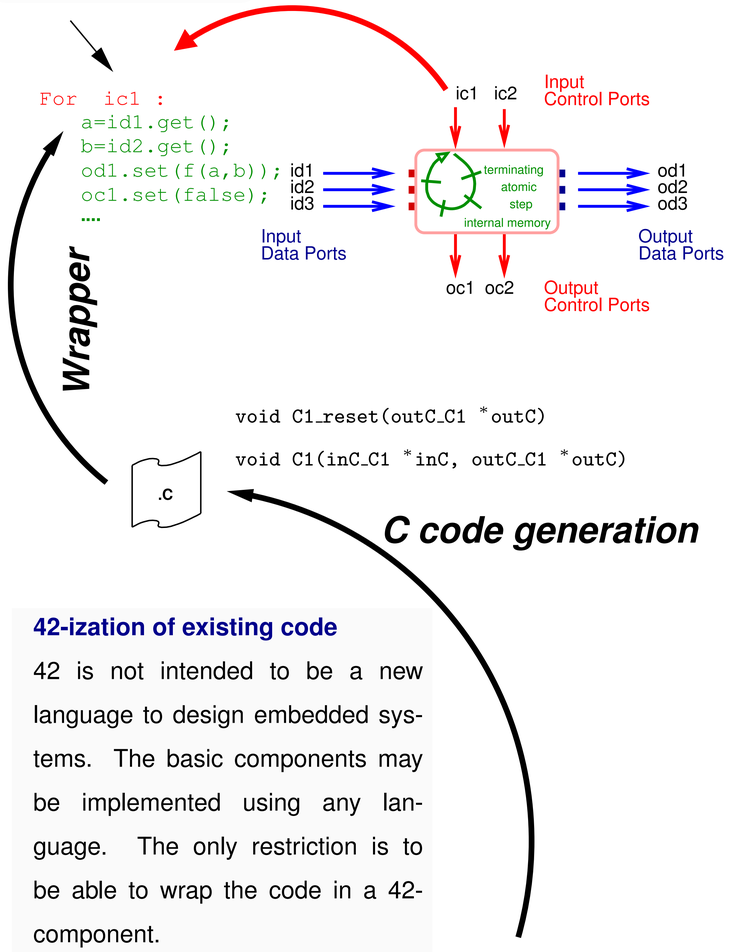
42

Florence Maraninchi, Tayeb Bouhadiba (Verimag/Grenoble INP) firstname.lastname@imag.fr

A basic component in the 42 formalism

Each time a component is activated with ic_i, it consumes inputs, performs an atomic computation, and produces outputs.

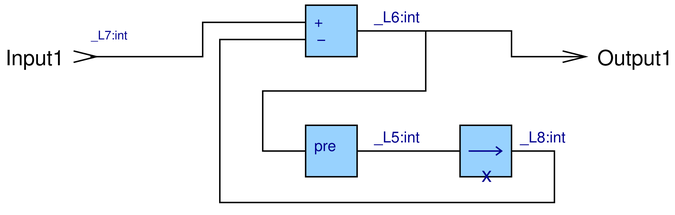
A piece of terminating and atomic code



42-ization of existing code

42 is not intended to be a new language to design embedded systems. The basic components may be implemented using any language. The only restriction is to be able to wrap the code in a 42-component.

Scade/Lustre diagram



42 Overview

42 [1] is a component-based approach to the virtual prototyping of embedded systems. A component is a black box with input and output data and control ports. Components can be programmed in all languages as long as they provide such an interface.

42 focuses on the “System-Level view”, i.e, the assemblage of components and the associated verifications.

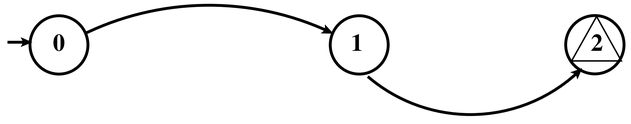
Protocol specification in 42

42 protocols

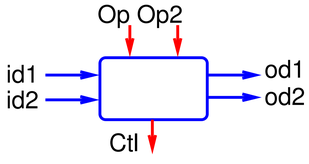
The possible uses of a component are specified by a control-contract, called a “protocol”. It may express:

- ⋆ Control sequencing
- ⋆ Data dependencies
- ⋆ Control information
- ⋆ Conditional data dependencies

(id1 and id2) op / α:=ctl (od1)



(id1 and IF α THEN id2) op2 (IF $\neg\alpha$ THEN od2)



Using 42 Protocols

42 protocols are used to verify assemblages of components and the compliance of the controller code w.r.t the components’ protocols. The verification may be done statically (a model- checking problem) or dynamically.

A system

```
Controller is :
var M : bool = true ;
for X do :{ /* defines X.
m_a, m_b, m_c: FIFO(1,int);
m_d, m_e, m_f: FIFO(4,int);
if (M) {
m_a.put ; /* reads i1.
m_a.get ; D.z;//*activates D.
m_f.put ; m_f.get ;
A.u; m_b.put; m_d.put;
m_b.get; B.v; M = M or p ;
m_c.put ; m_c.get ;/*assigns O1.
m_d.get ; C.w ; m_e.put ;
m_e.get; D.k ;
M = ! M ;
} else { ... }
Y = M;/*assigns Y.
}
```

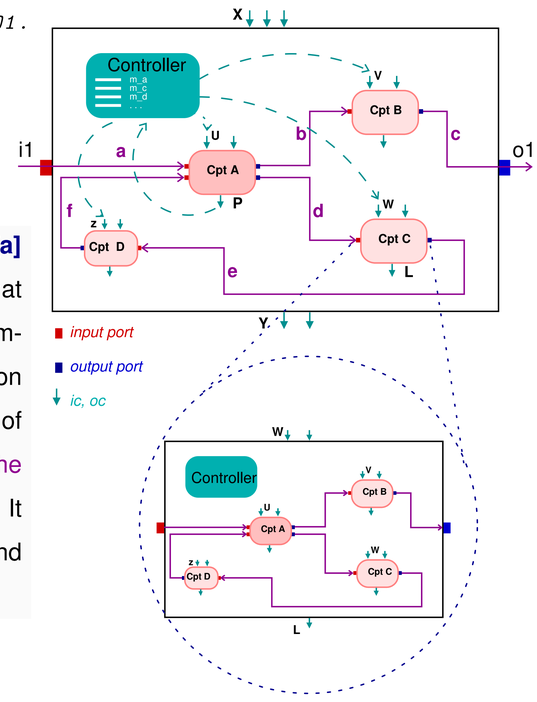
The controller defines the MoCC [a]

The controller is a small program that could be implemented in any programming style. For each global activation (e.g., X) the controller is in charge of activating components, managing the memory associated with each wire. It assigns values to the global data and control output ports (e.g., Y, O1).

A system

A system is made of a set of components whose data ports are connected by oriented wires. The wires do not express any synchronization.

The semantics of the assemblage is defined by the controller (as in Ptolemy [2]).



Modeling heterogeneity

A system that needs several MoCCs is modeled with several levels of hierarchy, each of them having a specific controller.

Reference and Definition

1 Florence Maraninchi and Tayeb Bouhadiba. 42: programmable models of computation for a component-based approach to heterogeneous embedded systems. In *GPCE ’07: Proceedings of the 6th international conference on Generative programming and component engineering*, pages 53–62, Salzburg, Austria, 2007. ACM Press.

2 http://ptolemy.eecs.berkeley.edu/ptolemyII/.

a MoCC : Model of Computation and Communication.