

Machine-Learning Framework for Automatic Netlist Creation

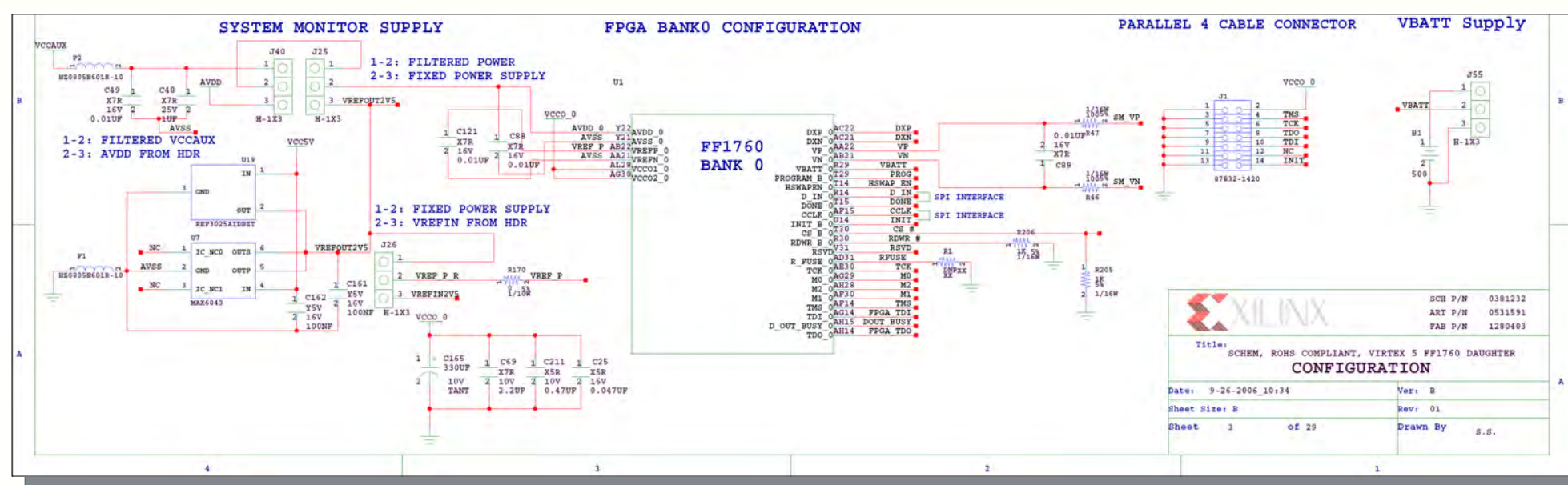
Mohamed Badreddine, Yves Blaquière, Mounir Boukadoum
Computer Science Department, microelectronic group, Université du Québec À Montréal

Abstract

This poster presents a framework for the automatic creation of netlists for arbitrary electronic circuits. The methodology relies on defining interfaces that allow a set of integrated circuits and other electronic components to be interconnected without user intervention. The framework, called “Intelligent Netlist Creator”, has been successfully tested on several circuits. The results show that the proposed flow for netlist creation assists the user by automating some connections whenever possible.

Introduction

An IC has pins with physical, electrical and behavioral constraints that can make schematic capture and the ensuing netlist creation very time consuming. Few reuses are done by the known tools on the market, where schematic entry restarts from zero for every new design, even if some ICs with up to hundreds of pins are reused under similar constraint restrictions.



Interface Approach

Interface definition :

- ◆ An interface is defined as the smallest set of pins that is necessary and sufficient to define a connection function for a component.
- ◆ Example: A port is a basic interface; a data bus is an interface...
- ◆ Interfaces can be reused by components of the same family.

Properties and uses of an interface :

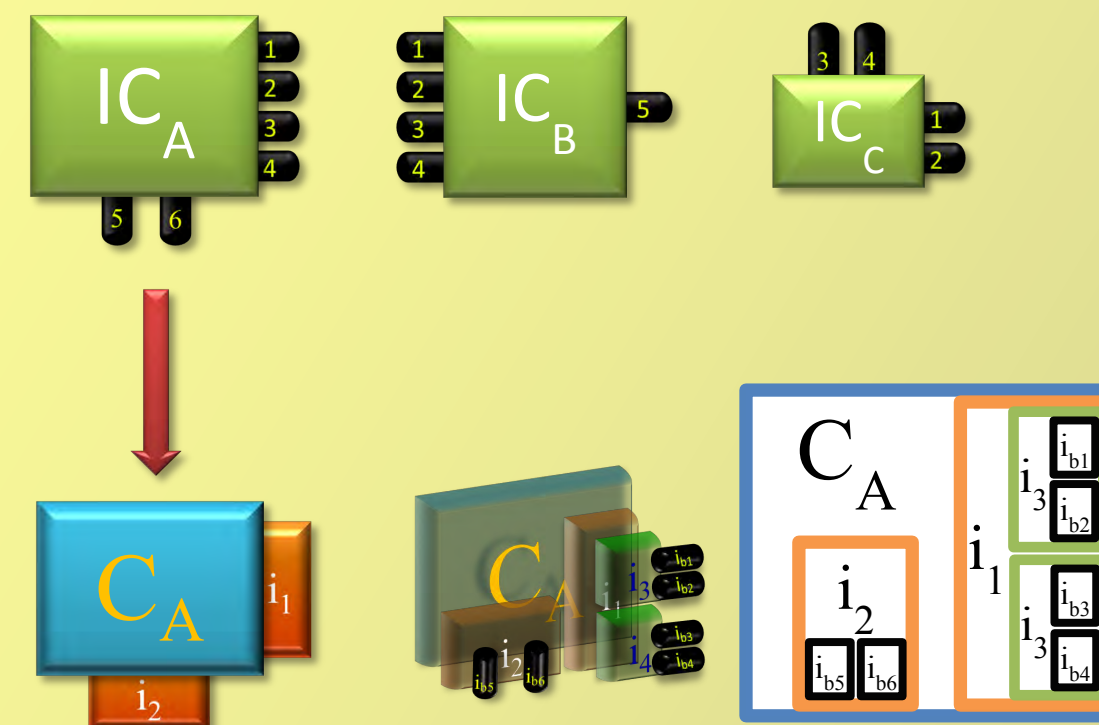
- **Atomicity:** Every pin is associated with at least one interface, called the basic interface.
- **Existence:** An interface is a set of basic and/or existing interfaces.
- **Grouping:** The configuration of a component is a set of interfaces that covers all its pins.
- **Sharing:** An interface can have one and only one parent interface in a configuration, unless it is sharable.
- **Connectivity:** Every interface has an attribute family (list) that designates the IC families to which the interface applies to.

Applying constraints :

- ❖ Physical (e.g. position) and electrical (e.g. V_{DD}) constraints are inherited from ports and components upon the creation or selection of an interface.
- ❖ Behavioral constraints (e.g. delay) are calculated by A.I algorithm.

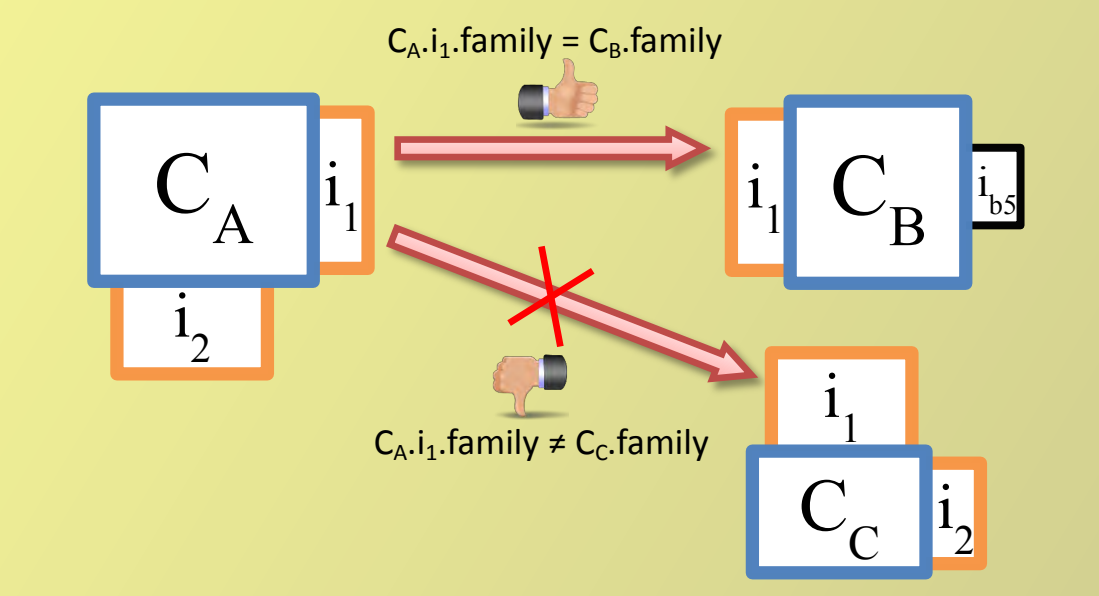
Automatic Connection Algorithm

1. Select a list of IC components.

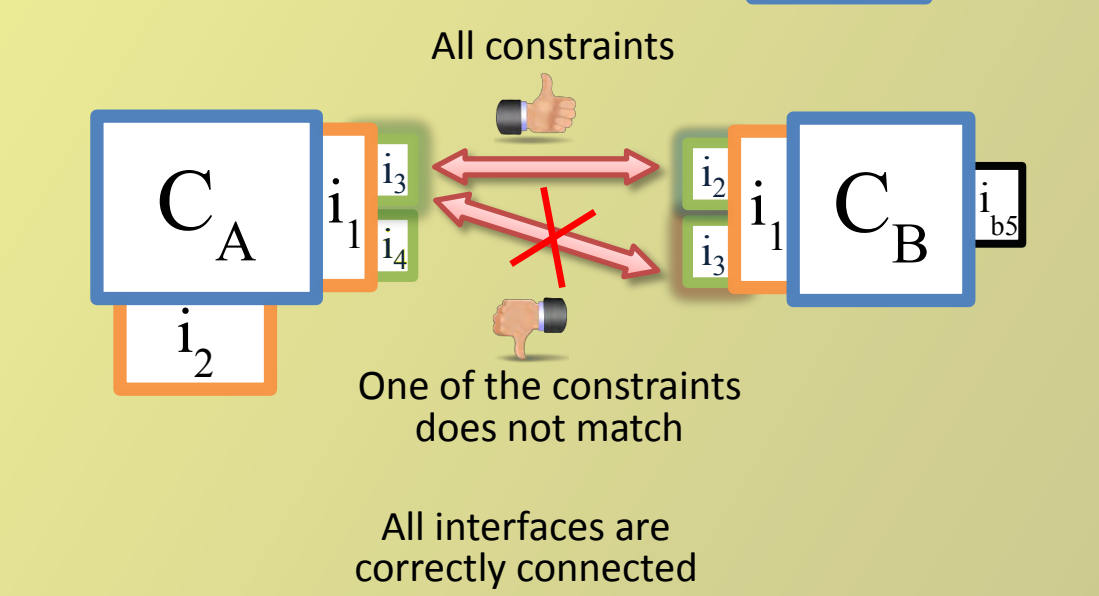


2. For each IC component, load or create its corresponding configuration based on interface properties and constraint inheritance.

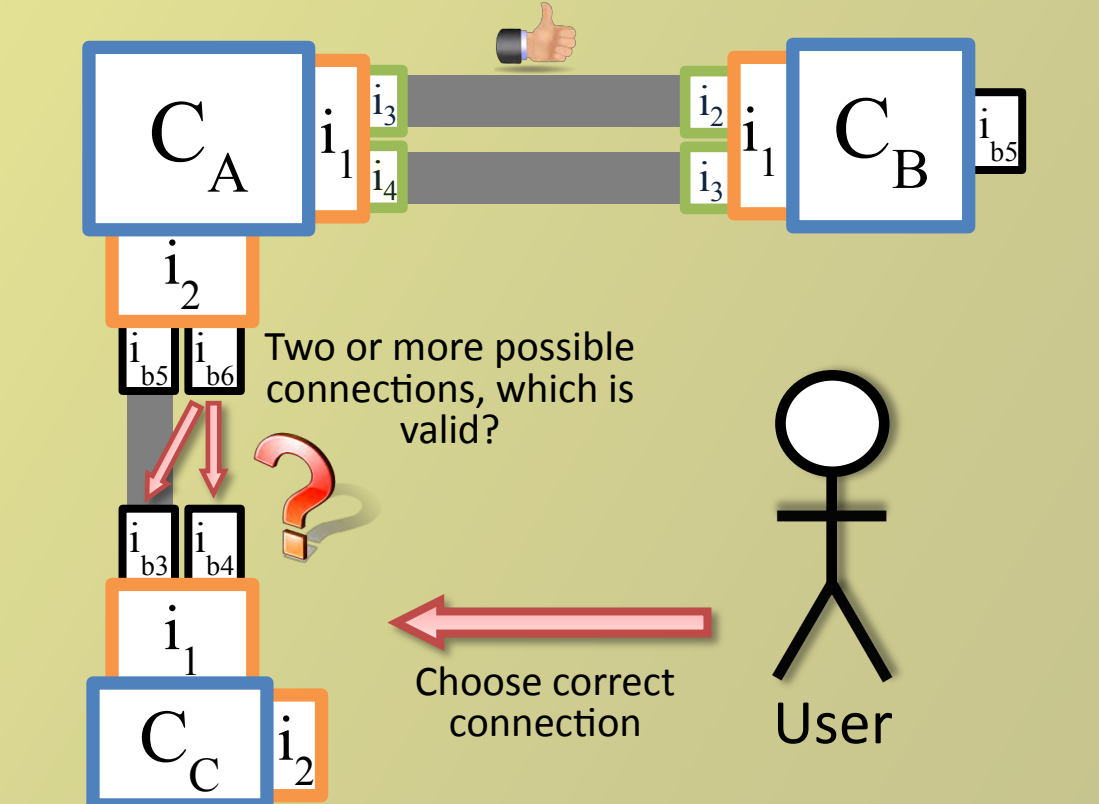
3. For each interface i_k , find the list L_{mi} of matching interfaces i_m from other configurations of the same family.



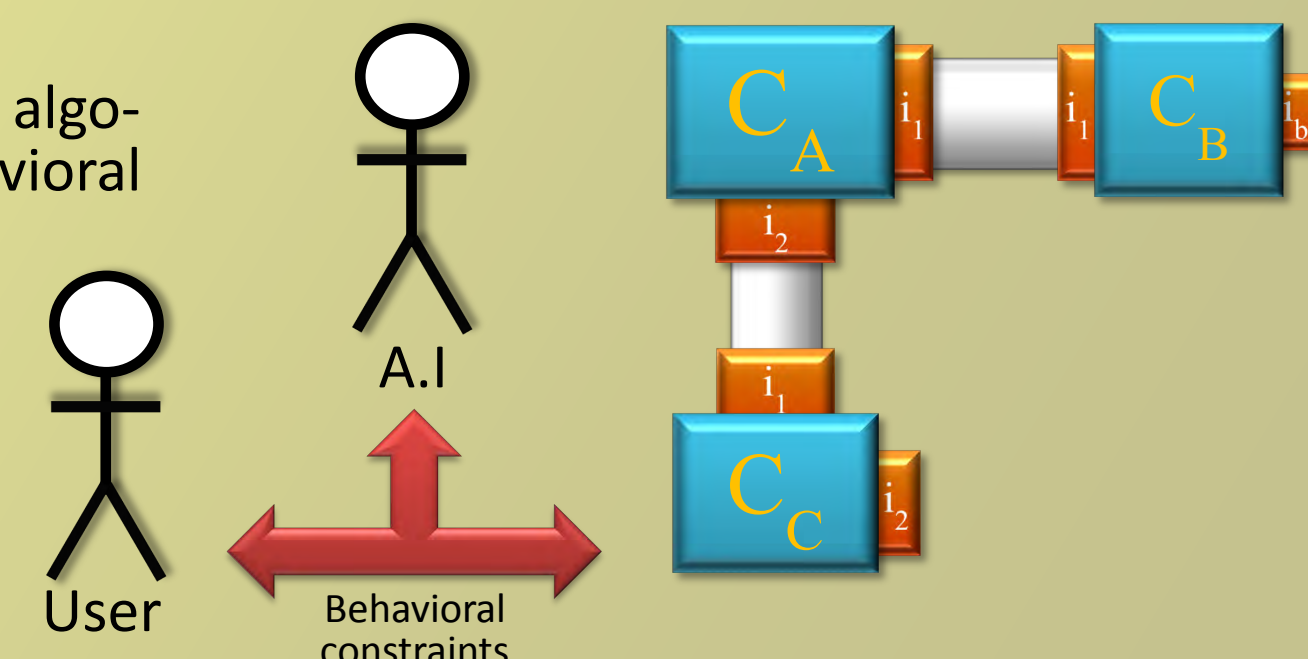
4. For each interface i_k , eliminate the wrong interface i_m in L_{mi} by applying matching electrical and physical constraint mismatch between the pair (i_k, i_m).



5. If only one interface i_m is remaining in L_{mi} then create a connection between i_k and i_m ; else, list the possible connections for user approval.



6. Call machine learning algorithm in order to fill behavioral constraints.



Interface Design Application

Table 1: Created interfaces of selected ICs in arbitrary circuits

IC	n	i_b	C_b	i	C
288Mb SIO RLDRAM	144	144	1	32	1
FPGA Spartan 3	208	262	1	153	3
64bits RAM	16	16	1	4	1
iP1202 power block	161	161	1	9	1

Where n: number of pins, i_b : nbr of basic interfaces, i: nbr of user-defined interfaces, C: nbr of user-defined configurations, C_b : nbr of basic configurations.

The total number of basic interfaces :

$$\sum i_b = n + \sum \text{alternated functions}$$

The total number of interfaces used in a circuit :

$$\sum i = \sum \text{user-defined interfaces} + \sum i_b$$

Since all interfaces have the same capacity C_i in memory, then, when compared to that of the original netlist in the database, the memory overhead M_o of the new netlist is approximately :

$$M_o = \sum i * C_i$$

The number of interface objects in a netlist mainly depends on the user design. The number of interface patterns stored in the database depends on the classification of IC components (in this case the family).

Key points of the interface approach :

- Meaningful interfaces are required to have efficient results.
- The user is assisted in the netlist design where the constraints are automatically verified.
- The abstract level of the interface design lowers the user effort to manage the circuit.
- Netlist design time decreases due of the interface reusability.
- Existing Netlists can be automatically parsed to extract interface models that will feed the A.I system.

Conclusion

A framework is proposed to assist the circuit designer in creating netlists based on the concept of interfaces. At this development stage, user intervention is still needed, especially regarding the behavioral constraints. The number of interface objects mainly depends on the user design and the number of interface patterns depends on the IC component classification. Meaningful interfaces are critical for the approach to yield efficient results, especially for the machine learning algorithm.

Acknowledgments

The authors thank the Natural Sciences and Engineering Research Council of Canada (NSERC), PROMPT Québec, MITACS and Gestion TechnoCap Inc. for their financial support and CMC Microsystems for providing design tools, support and access to technologies.