Politecnico di Milano

A data-path oriented, IP-Based Framework for Flexible Design Exploration

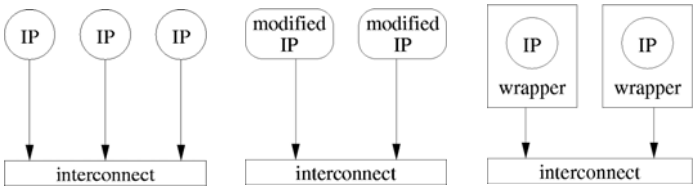
Speaker: Laura Frigerio
LFrigerio@Elet.polimi.it
C. Bolchini, C. Brandolese, W. Fornaciari, F. Salice

Goals

- The paper **aims** at proposing a new design flow, based on VHDL code generators, allowing both
  - a fast design of parametric specifications and
  - a fast redesign in order to respect tight constraints
- Key aspects:
  - Code readability
  - Technology independence
  - Parametric specification in term of both size and architectural aspects
- The activity **focuses** on portions of the design do not usually covered by standard (and complex) IP cores

Introduction

- Development of digital designs:
  - **HDL** has been the common practice for 15 years but *productivity gap* enforce the development of more efficient techniques
  - **High Level Synthesis** raises the abstraction level but is not completely satisfactory
  - **IP-Based Design** is a more promising solution but with some drawbacks
    - Implemented functionality are usually standards or complex blocks
      - Subset of user needs
    - HDL source is not even disclosed
      - hard modifications



3

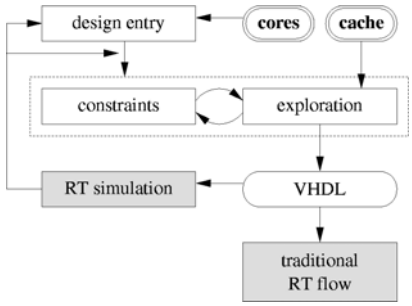
IP flows paradigms

- The current IP-based methodology partitions the design in two portions: standard blocks and sparse logic
- This work is a preliminary proposal for an intermediate-level block-based design flow aimed at filling the gap between the two portions
- Core-based design:
  - Pre-verified cores are integrated in the design
    - **+** Only the integration has to be performed
    - **-** Insufficient flexibility
- Design for reuse
  - Internal development of cores is performed
    - **+** Great flexibility
    - **-** More development effort

4

Proposed flow

- The proposed flow



- The framework is based on the *RoadRunner* toolset composed by:
  - RRCore: collection of parametric core generators
  - RRCache: database storing core characterization integrated with an estimation engine for uncharacterized modules.

5

Proposed flow

- System definition
  - **High Level Schematic Entry** exploits cores available in the *RRCore*; the schematic entry allows to identify the topology of the system
  - Modules exposes **parameters** that could be assigned or **left unassigned for the exploration phase**
  - Strategies for cores not available in the library
    - New generators are added to the library and to the design
    - New HDL modules are added only to the design
  - At the end of this phase a model of the system is available:
    - **Not all parameters are assigned**
    - **Functionality are all specified, but not architectures**

6

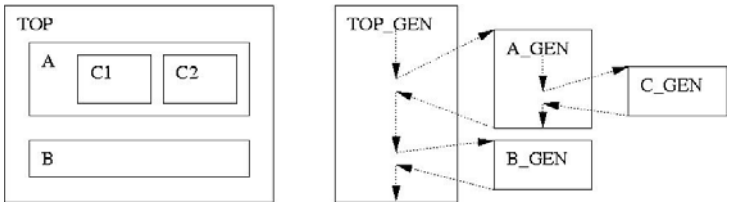
Proposed flow

- Constraints setting
  - The designer could specify some constraints to guide the exploration phase, according to the information stored in *RRCache*
    - Most common: area and delay
  - Constraints can be imposed on top level or on modules
- Space analysis
  - Different solutions are analyzed and compared, and the most suitable solutions are proposed
    - *RRCache* supports this phase
  - The designer could change the parameters to perform different explorations
- Simulation, synthesis and back-end
  - An external simulators is used; if the result is not satisfactory another iteration is performed
  - Finally, synthesis and back-end are performed with commercial tools.

7

Parametric cores

- Modules composing the library
  - ***Atomic IP*** or ***atom***: very simple module with no internal hierarchy
  - ***Molecular IP*** or ***molecule***: composite module constituted of Atoms of Molecules
- Hierarchy is obtained combining the generators and **not the modules themselves**



8

Parametric cores

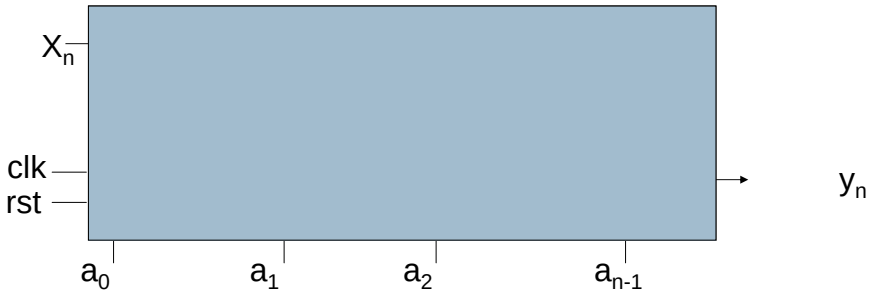
- Both atoms and molecules are characterized by parameters
- Molecules have **opacity** as an additional property
  - The degree of opacity changes the way its structure is visible from outside
    - **Transparent**: all the parameters are exported to the top level
      - The molecule exposes all its constituents
    - **Semi-transparent**: parameters are exported only at a given level of hierarchy
    - **Opaque**: only parameters related to top level are visible
      - No parameter related to molecule constituents is accessible for the exploration

9

Parametric cores

- Example: FIR FILTER

OPAQUE



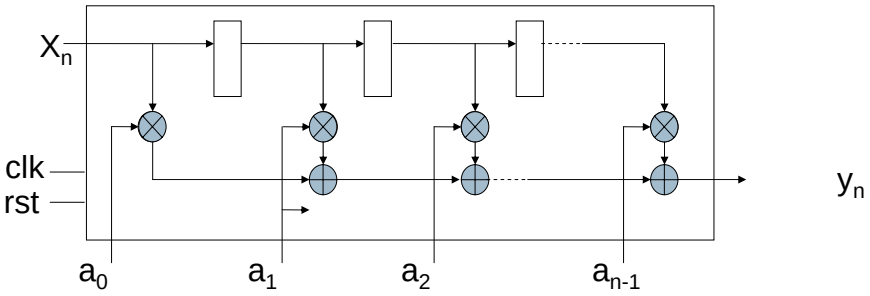
Number of TAPS
Weights
Active clock edge
Active reset level

10

Parametric cores

- Example: FIR FILTER

SEMI-TRANSPARENT



Number of TAPS
Weights
Active clock edge
Active reset level

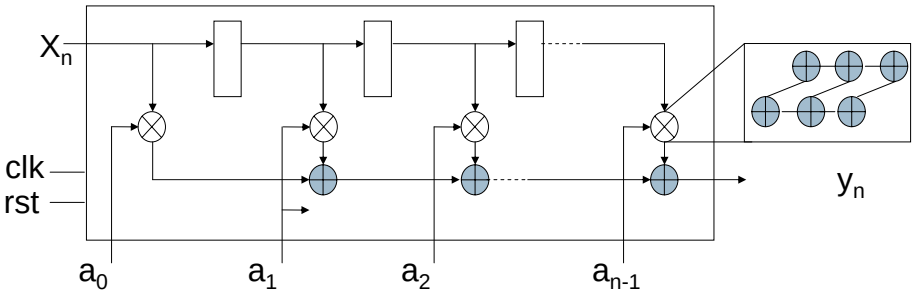
Multiplier architecture
Adder architecture

11

Parametric cores

- Example: FIR FILTER

TRANSPARENT



Number of TAPS
Weights
Active clock edge
Active reset level

Multiplier architecture
Adder architecture

Architecture of adders
to build multipliers

12

Parametric cores

- When structuring the system the designer must:
  - Instantiate atoms and molecules,
  - Connect them and
  - **Assign values to parameters**
- A parameter can be **explorable** or **non-explorable**
  - **Explorable** parameters can be left unassigned in the design phase and determined during exploration
- The value of a parameter can be **fixed** or **variable**
  - A **fixed** value is assigned by the designer, a variable one is left free for the exploration
- A value can be defined at top level or at module level
  - For example reset is defined at top level and inherited by all submodules

13

Parametric cores

- Parameters and values properties

			PARAMETERS	
			Explorable	Non-explorable
VALUE	Fixed	Module	X	X
		Top-level	X	X
	Variable	Module	X	
		Top-level	X	

14

Parametric cores

- Parameters and values properties

			PARAMETERS	
			Explorable	Non-explorable
VALUE	Fixed	Module	X	X
		Top-level	X	X
	Variable	Module	X	
		Top-level	X	

15

Parametric cores

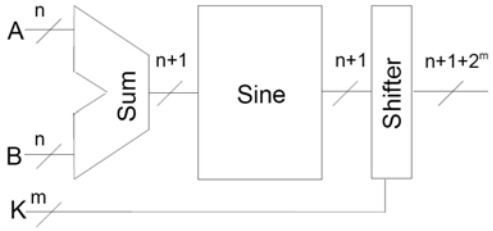
- A complete design is composed of modules and a number of module-level and top-level parameters, some of them explorable and variable
  - **These parameters define the dimension of the design space**
- The exploration is basically an efficient analysis of all possible assignments of values to parameters under a set of constraints
  - Minimum clock period, maximum combinatorial delay, minimum area
- The exploration relies on values stored in a database and on an interpolation mechanisms to extract values not stored

16

Experimental results

- Simple example: computation of

$$F(A, B, K) = \sin (A+B) \cdot 2^k$$



- Adder architectures: ripple-carry, carry look-ahead
- Sine function architectures: LUT, CORDIC pipelined, CORDIC iterative
- Shifter architectures: arithmetic or logarithmic

17

Experimental results

- Parameters: *n* is fixed at 8 and *m* at 3
- Area and time figures of each module are considered
- For sequential modules throughput and latency are also considered:
  - The CORDIC iterative architecture requires 9 clock cycle to compute the data
  - The CORDIC pipelined architecture provide a result every clock cycle after an initial latency of 9 cycles

	LSI_10k (mils) Optimization for		Virtex-II Pro (LUT) Optimization for	
	Area	**Time**	**Area**	**Time**
Adder-cla	81	273	9	9
Cordic-pl	6992	7164	179	184
Cordic-it	1565	1632	107	116
...				

Experimental results

- Three different types of constraints:
  - **Area optimization**

Module	FPGA Tech.	Asic Tech.
Adder	Adder-RC	Adder-RC
Sine Comp.	Cordic-it	LUT
Shifter	Shifter-log	Shifter-log

  - **Area/timing tradeoff**

Module	FPGA and Asic Tech.
Adder	Adder-RC
Sine Comp.	LUT
Shifter	Shifter-log

  - **Timing optimization**

Module	Max Throughput	Min delay
Adder	Adder-CLA	Adder-CLA
Sine Comp.	Cordic-pl	LUT
Shifter	Shifter-log	Shifter-log

Conclusions

- This paper presents a new methodology for the design of complex data-paths. Major novelties:
  - Rich library of module generators from very simple ones (adders, shifters, ...) to rather complex ones (CORDIC, FFT, ...)
  - Modules are highly parametric; only functionality needs to be specified allowing to exploit architectural details for space exploration
  - The VHDL code is clear and readable, important both for documentation purposes and for maintainability/customizability
  - The exploration algorithm relies on figures derived from pre-characterization of the modules or from estimates.
- The current design flow relies on a preliminary implementation but formal aspects have been already defined