FripGa: A Prototypical Design Tool for Embedded Multi-Core Systems-on-Chip

Alexander Biedermann and Sorin A. Huss
biedermann@iss.tu-darmstadt.de, huss@iss.tu-darmstadt.de

TECHNISCHE UNIVERSITÄT DARMSTADT
Department of Computer Science
Integrated Circuits and Systems Lab

Idea

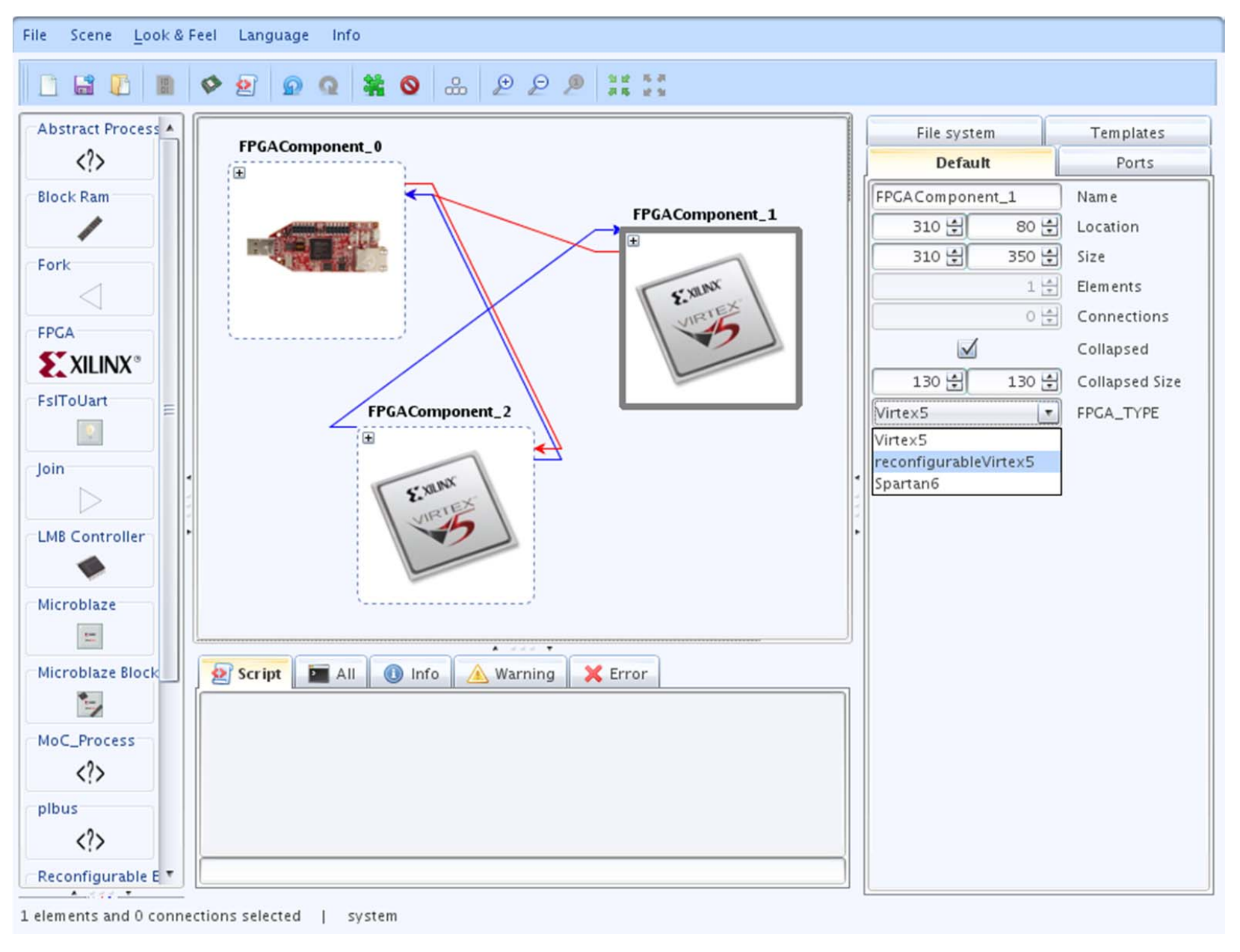
- Today's FPGAs offer plenty of logic resources
- Complex Systems-on-Chip may consist of several dozens of processors and HW IP Cores
- Design tools often still tailored to "single processor with (HW) co-processors" designs

➔ **Develop prototypical design tool with usability features which future commercial tool suites should provide soon**

Challenges

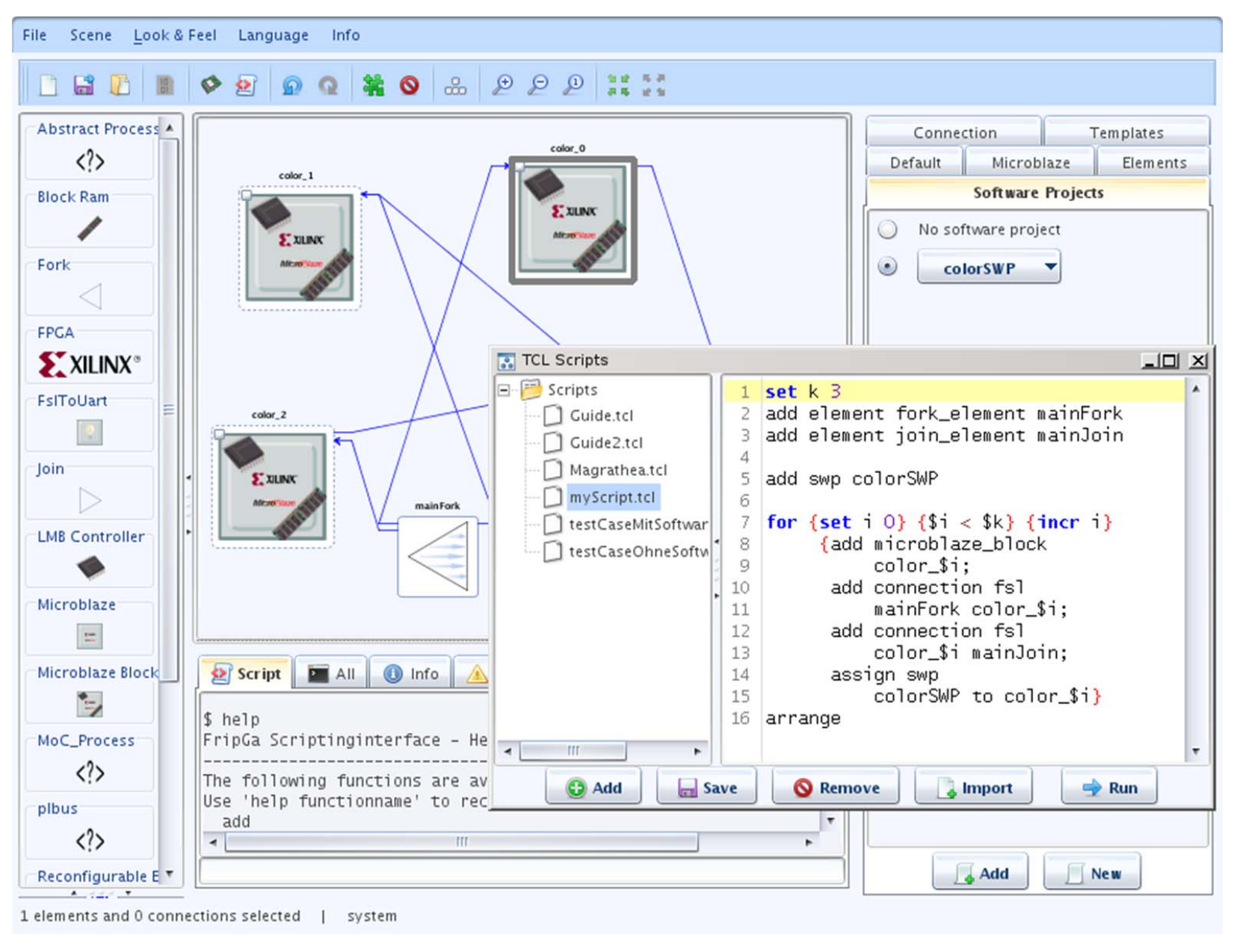
- How to enable smooth, fast, and safe creation of complex Systems-on-Chip?
- How to avoid repetitive, error-prone user inputs?
- How to visualize complex Systems-on-Chip?
- How to support easy design space exploration?
- How to include methodologies that abstract from underlying architectures?

FripGa: Key Features



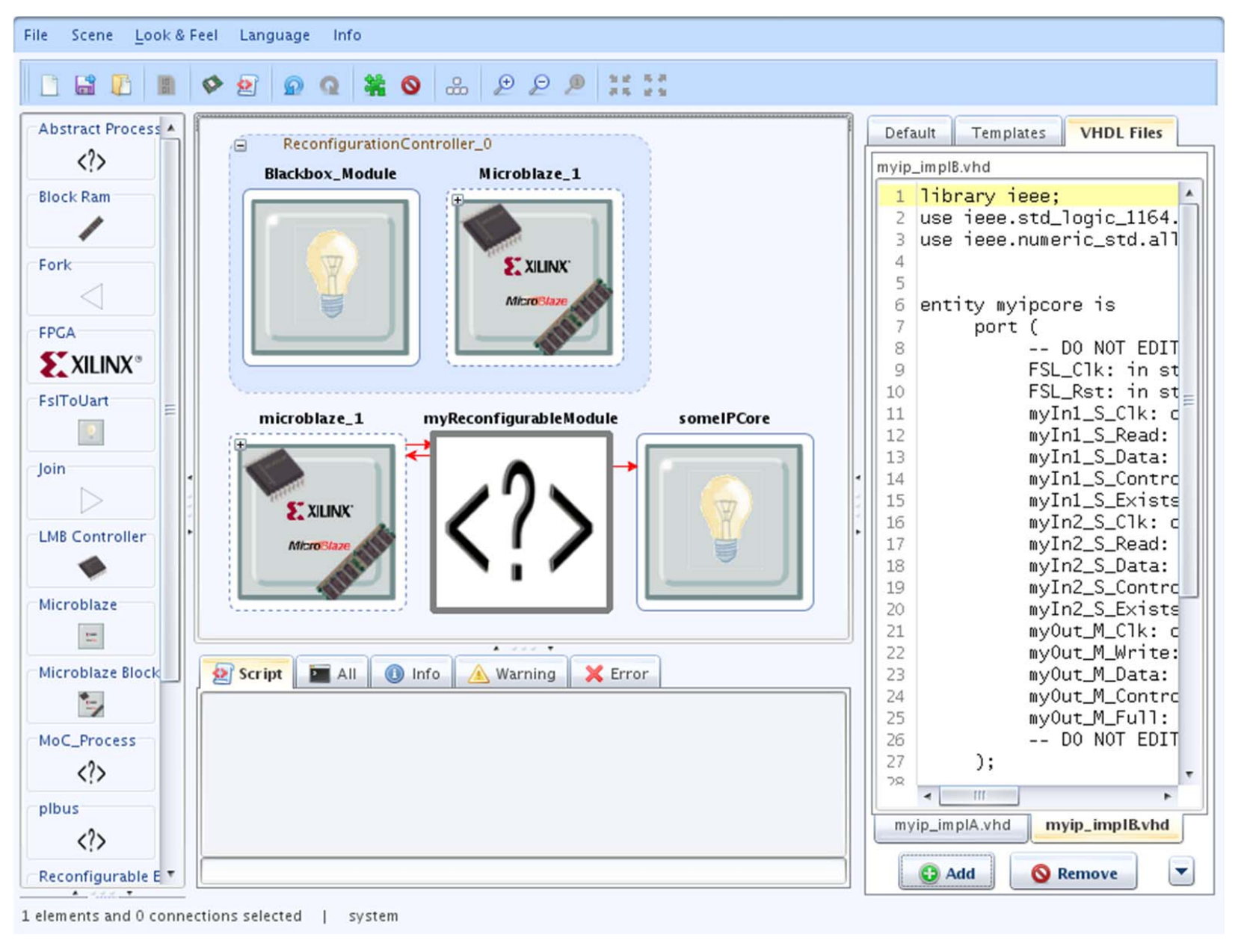
Multi Chip Designs

- Scatter designs over multiple FPGAs
- Automatic mapping of modules to FPGAs based on user constraints (max. number of FPGAs, max communication speed, or minimal resource consumption)



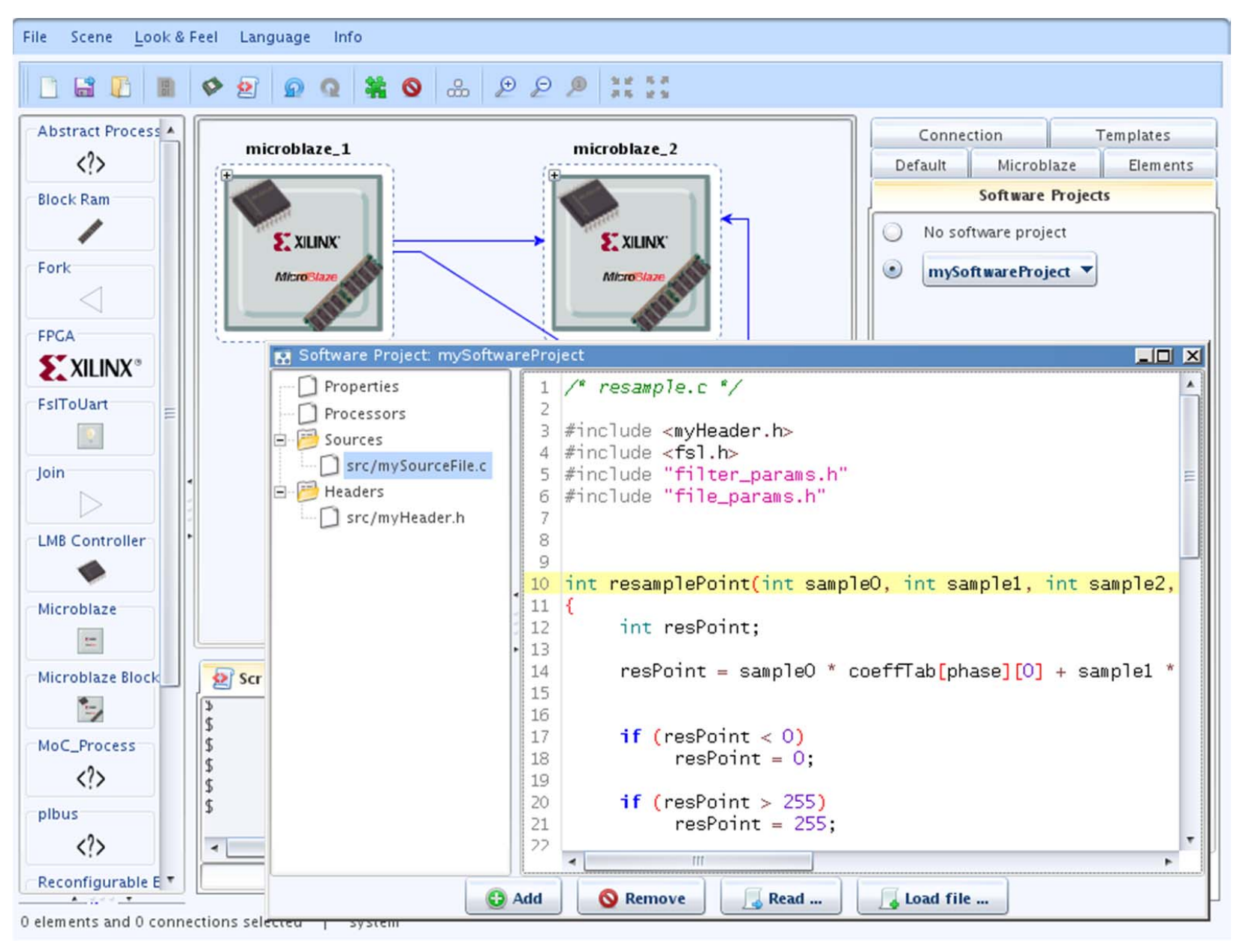
User Interface

- Build HW/SW designs from scratch
- Instantiate, duplicate, connect, and configure embedded processors and IP cores
- TCL scripting engine for all features



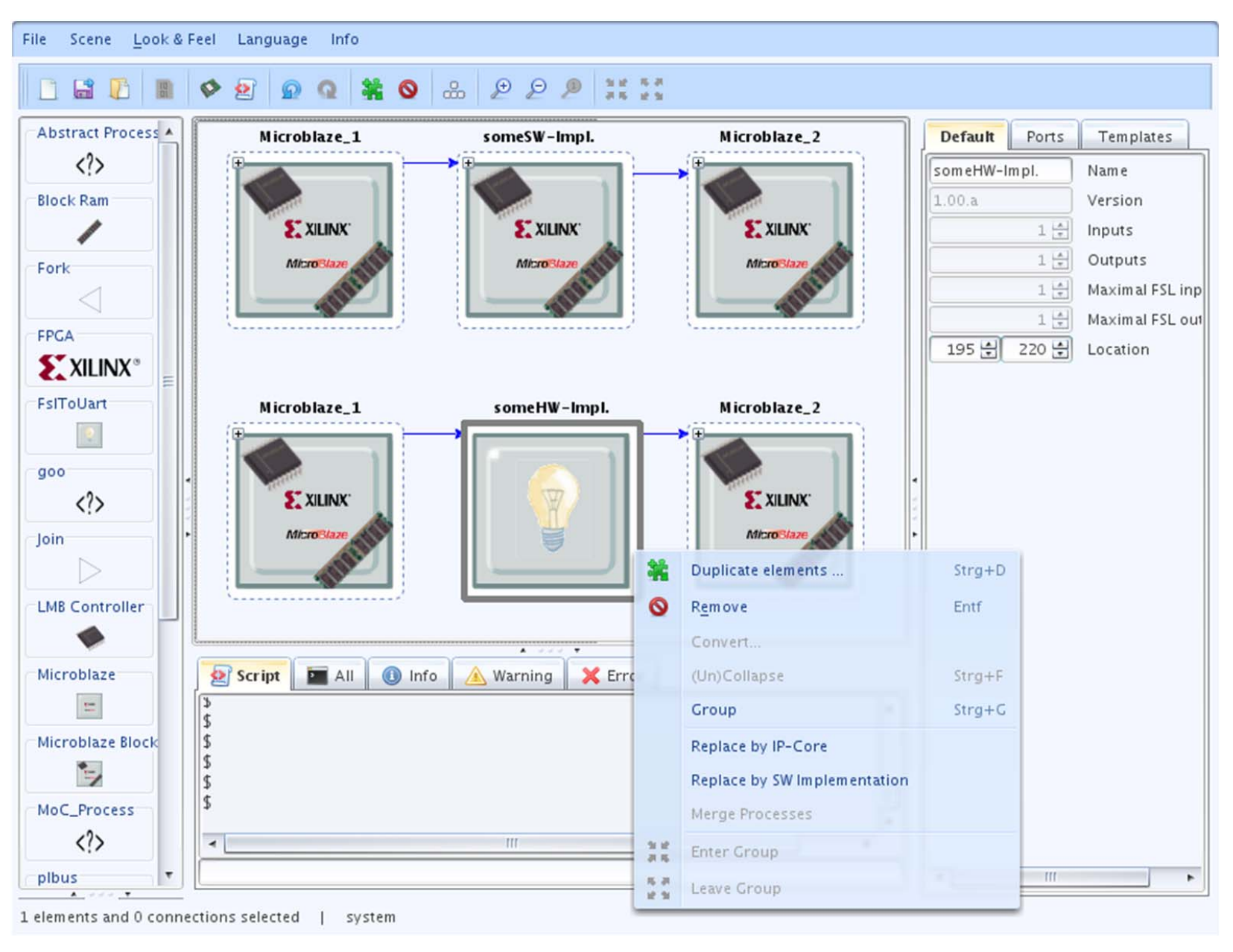
Partial Reconfiguration

- Avoids manual floor planning
- Automatic instantiation of partial reconfigurable regions on FPGA chip area
- Automatic resource analysis to determine sizes of partial reconfigurable regions



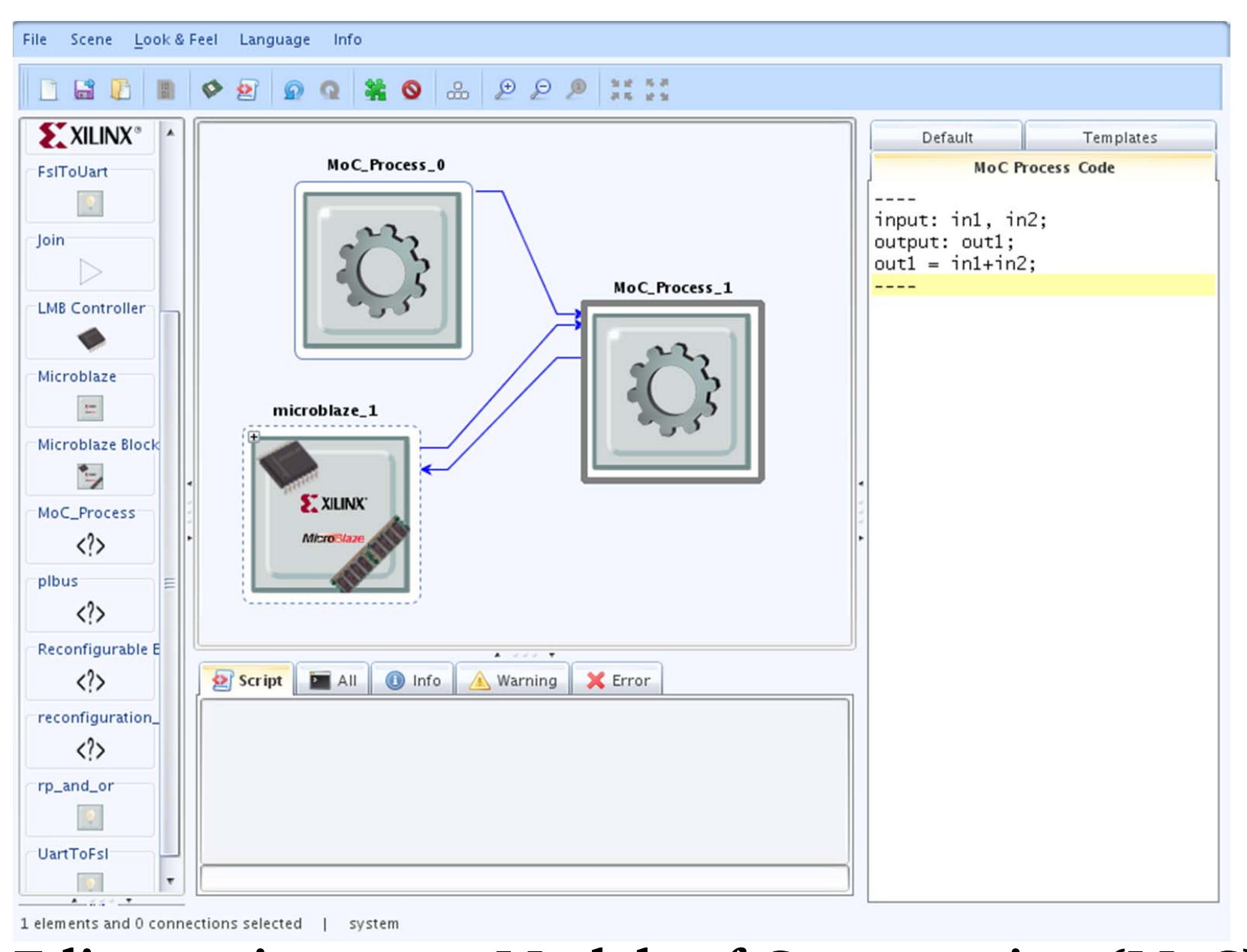
Handling of Embedded Software Projects

- Import and edit legacy software projects for embedded processors
- Create and edit new software projects
- Easy binding of SW projects to processors



Switching between Software and Hardware Implementations

- Allows for smooth and fast design space exploration
- Guided by a wizard to match interface connections



Editor to integrate Models of Computation (MoC)

- MoC processes may be partitioned to either SW or HW by the designer
- MoC processes may be connected to common modules such as embedded processors or HW IP cores

Additional Features

- Visual representation, editable at any time
- XML file structure for efficient file handling
- Wizard to import legacy IP cores or to create new ones
- Grouping of modules to functional units to use them as hierarchical design primitives
- Full compatibility to existing commercial tool suites, e.g. Xilinx Platform Studio
- Built-In IP cores for data path manipulation

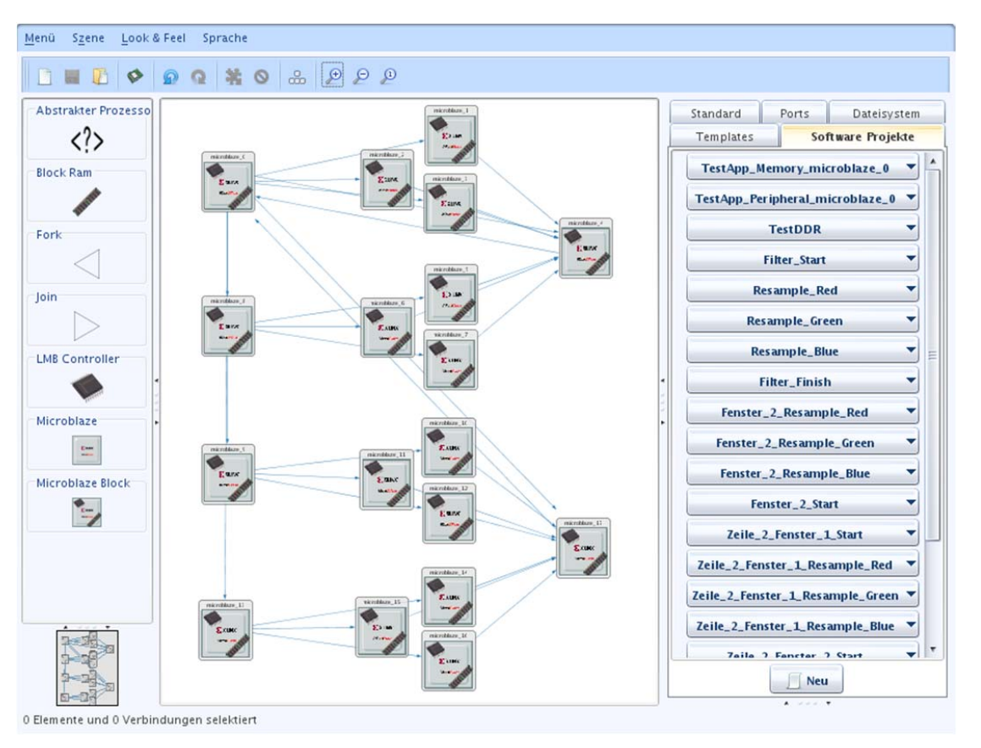
Comparison of Workflow Efficiency

	Xilinx XPS		FripGa	
Common operations	MC[1]	Time	MC[1]	Time
Add embedded processor to design	47	180 s	1	1 s
Establish module-to-module connection	12	28 s	1	1 s
Duplicate group of modules	n.a.	n.a.	2	6 s
Add partial reconfigurable modules	>100	>1800 s	9	25 s

[1] MC = mouse clicks

Application Example

- Parallelized FIR filter for image scaling designed in FripGa
- Sub-pixel resampling done either in hardware or in software
- Design with up to 18 soft-core processors created in less than one hour
- Easy switching between different implementations or levels of parallelization



Future Work

- Expand functionality of MoC editor, e.g., by exploiting known MoC transformations from system level to component level
- Exploit virtualization schemes to further abstract from strict software-processor bindings and automatically choose optimal number of processors employed in design
  (see poster "Scalable Multi-Core Virtualization for Embedded System-on-Chip Architectures" on DATE'12 Friday Workshop: Quo Vadis Virtual Platforms?)

Dipl.-Inform. Alexander Biedermann
Technische Universität Darmstadt
Integrated Circuits and Systems Lab
Hochschulstraße 10
64289 Darmstadt, Germany

Prof. Dr.-Ing. Sorin A. Huss
Technische Universität Darmstadt
Integrated Circuits and Systems Lab
Hochschulstraße 10
64289 Darmstadt, Germany