Visual Exploration of Simulated FPGA Architectures in Odin II

Konstantin Nasartschuk / Kenneth B. Kent / Rainer Herpers
University of New Brunswick
Faculty of Computer Science
Bonn-Rhine-Sieg University of Applied Sciences
p652d@unb.ca ken@unb.ca rainer.herpers@h-brs.de

Outline

- **Visualization framework for the workflow used in the VTR project.**
- **Exploration support for FPGA**
- **Visualization of the simulation process which is provided by Odin**

Motivation

Odin is part of the VTR workflow, where four tools are used to compile, optimize and route a Verilog hardware description onto a FPGA design. Each tool changes the initial circuit during the optimization process. Sub results are saved using the "Berkeley Logic Interchange Format(BLIF)". The file format is not designed to be human readable. A BLIF file explorer was implemented to support developers of the workflow.
The goal of this project is to improve the BLIF explorer by connecting it to Odin II and to visualize not only the circuit but also it's simulation. A visual simulation of a circuit allows not only to evaluate and explore the structure but also the functionality of a specific circuit.

Background

The VTR project consists of six different tools. Odin II compiles a Verilog source file and creates a hardware circuit. This circtuit is optimized using an architecture description file which represents the specific abilities of a FPGA device. ABC performs logic optimization on the circuit before being processed by T-VPack. The VPR6.0 tool is used to Place the logic blocks on the FPGA device and connect them according to the structure in an optimized way.

The BLIF explorer is able to visualize a circuit which is described by a BLIF file. Comfort functions such as search, highlight, remove, rearrange are included. It consists of multiple modules which are responsible for specific parts of the software. The BLIF explorer is created using the Qt framework and written in C++.

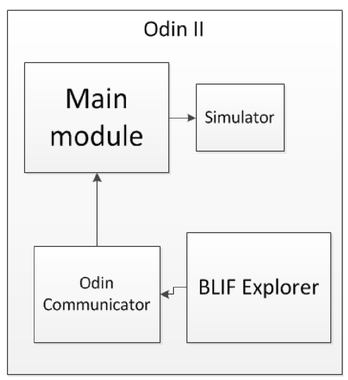
Problem

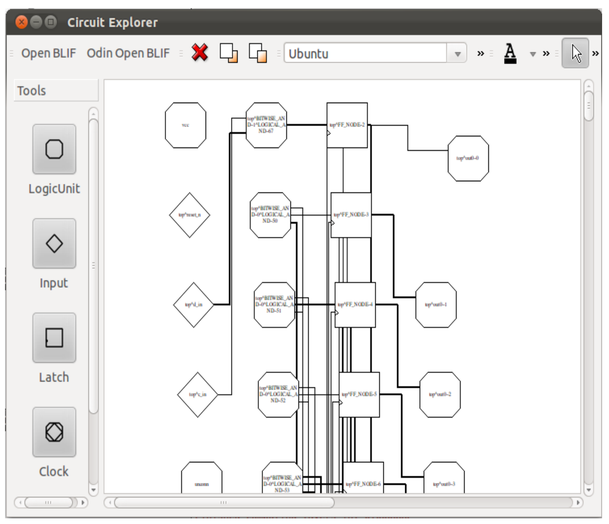
The project is divided into two parts:

1. **The BLIF Explorer is not capable of communicating with Odin II during run time. A solution has to be found to access the netlist, which is created by Odin II. This netlist can be used to connect the graph which is shown in the visualization to it's logic partner nodes. During the simulation process the output values of all logic units have to be found and shown in the visualization. Also the control of the simulation process should be provided using the graphical user interface.**
2. **Using the developers feedback functionality which could improve the exploration process have to be added. This considers visual as well as functional feedback. Examples are: showing clock inputs using a specific symbol, highlighting all connections of a logic block, arranging the graph in waves etc.**

Challenges

- **Odin II is written in C. The explorer is written in C++. Accessing all functions, which are needed in the visualization without changing the structure of Odin II or the BLIF explorer requires an additional module in the BLIF explorer.**
- **Odin II and the BLIF explorer are developed independently. The BLIF explorer uses a compiled version of Odin II.**
- **Processing times of Odin II and the BLIF explorer are added as each tool uses a blif file to create it's structure.**





Solution

- **Implementation of a communication unit which commands Odin II and offers its functionality to the software.**
- **Adding a reference of an Odin II node to every logic block in the visualized graph in order to recognize the logic node and it's visual representation**
- **Reading output status of each node and visualizing the current state of the connections**
- **Implementation of comfort functions which are requested by developers**

Result

The current version of the BLIF explorer is connected to Odin II and is able to access the netlist. It has access to simulation states and is capable of presenting them to the user. Access times were minimized as only Odin II needs to process a file. The internal netlist is used to create the visualized graph. Arrangement of the nodes in the graph was improved as well as shapes of different unit types added to the visualization.

UNIVERSITY OF NEW BRUNSWICK
Reconfigurable Computing Research Group
FACULTY OF COMPUTER SCIENCE