The proposed approach allows debugging of partial dynamic reconfiguration. It shows where and when FPGA areas are reconfigured at runtime.

1. Introduction

Hardware monitoring and debugging are very resource consuming activities with respect to human and machine resources. To reduce the amount of human resources spent on debugging, better tools for automatic integration, error identification, and data filtering are necessary. Solutions for debugging and monitoring of the IP (Intellectual Property) core’s computation [1][2] and for generic FPGA debugging [3] already exists.

Today’s high-performance System-on-Chip (SoC) and Network-on-Chip (NoC) [4] contain a great number of communication lines for their functional purposes. Those lines host information that goes from one component to another using a specific communication protocol and that can be interesting both for debugging and monitoring purposes. In a digital system, the communication between the IP cores is fundamental for being sure that the system works as expected. The identification of defects or bugs is possible if the designer has a complete view of the embedded system’s state at runtime. The most natural solution to this problem is to make the internal signals observable from outside.

The proposed approach allows debugging of partial dynamic reconfiguration. The user interface graphically represents the reconfiguration of FPGAs in real time. In detail it shows where and when FPGA areas are reconfigured. The information regarding where the modules are placed is read directly from the partial bitstreams using a SiLLis-generated listener component. This makes the visualization highly accurate and useful for debugging purposes.

2. Proposed Methodology

System monitoring can be performed by the use of components able to gather data from a selected set of data signals. This can be performed in many ways: one of the most common solutions is to sample the data at every clock cycle, store them in memory or transfer them directly to an external debug application. This approach is simple and easy to implement, but requires a lot of resources, especially regarding memory.

3. Simplified Language for Listeners

SiLLis (Simplified Language for Listeners) is a C-like text description language for listeners. It combines an easy grammar and a powerful set of instructions in order to increase the speed and the accuracy of the process of creating listeners and the inclusion of these in the system.
SiLLis is composed by several parts that are required to define the interface, the parameters, the variables and the behavior of the listener. These aspects are sufficient to define which lines must be monitored and controlled and in which way.

The language can describe three kinds of listeners:

- **Debug Component**: A listener is considered as a debugging component if it only reads data without writing.
- **Watch Component**: A listener able to read from a communication line and to write to a different one within the same system. It can be used to watch for undesired events.
- **Control Component**: A listener that can write to the same communication line it reads. This closes the feedback loop so that the listener can be used as a controller.

### 4. Reconfiguration Viewer System

Reconfigurable systems are interesting to prove the efficiency of the language. Therefore, we build an application in which a listener is used as reconfiguration monitor.

The scheme of the system is shown in Figure 2. The reconfiguration, called Virtex Configuration Manager (VCM) [5] is a self-developed, speed-optimized Configuration Manager used instead of the widely used HWICAP EDK-Component. Besides the self reconfiguration using the ICAP interface, it enables fast partial reconfiguration of Xilinx FPGAs using the dedicated configuration logic of RAPTOR2000 system [6].

The listener waits for the VCM master request to the SDRAM. When the request is performed by the VCM, the listener is activated and waits for valid data from the SDRAM controller. The data that goes from the SDRAM to the VCM composes the bitstream that will reconfigure the FPGA. Note that such components can be written in VHDL or in Verilog but SiLLis improves the speed and efficiency of the process. All the data sampled by the SiLLis Listener is transferred to the external application for parsing and visualization. With this method it is possible to visualize the portion of the FPGA reconfigured at runtime using a dedicated GUI.

![Figure 2: SiLLis listener used as reconfiguration monitor](image)

### 5. Conclusion

Custom monitoring components, called listeners, are able to reduce the amount of data that a debugging system has to manage. This improvement is performed by the introduction of filters in the process of monitoring that are easily describable in SiLLis. This language supports the user in describing the desired listener behavior.

### 6. References