


Bitfile Preservation - Generation Of Reusable Out Of Context Modules

Christian Stüllein, Norbert Abel, Udo Kebschull

Chair for Infrastructure and Computer Systems for Data Processing (IRI), Frankfurt University

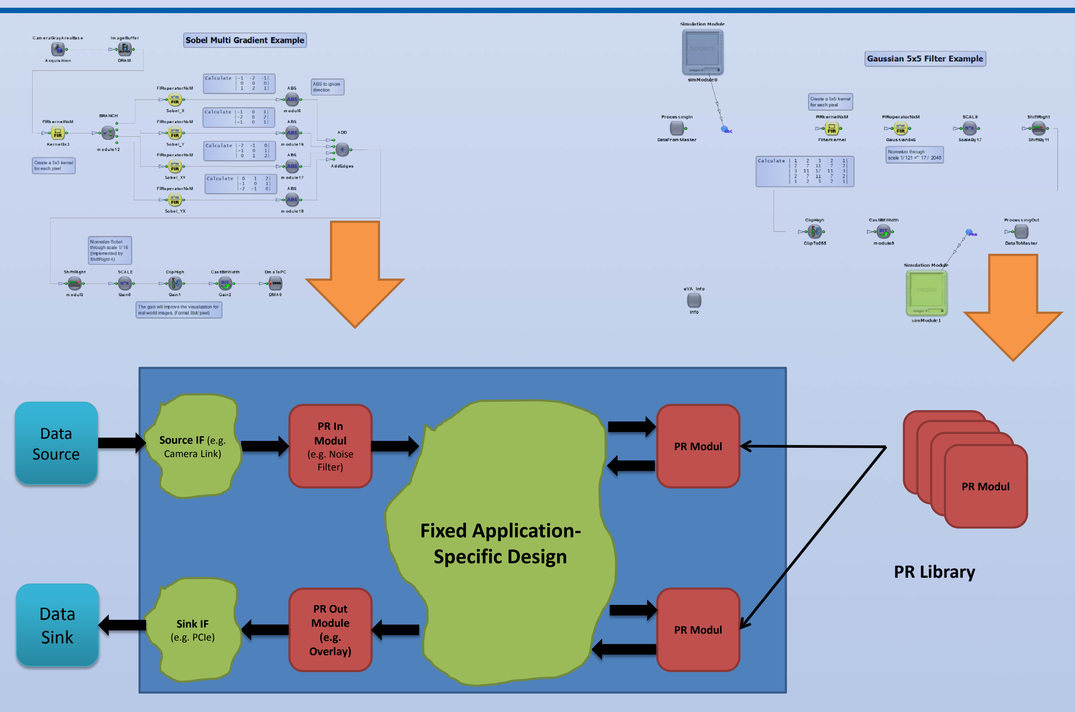
Abstract: This paper presents the idea of bitfile preservation which enables the re-use of partial bitfiles in different environments and at different positions of the FPGA without any re-compilation. This way, the behaviour of a system can be changed on the fly just by plugging together different pre-compiled modules (represented by partial bitfiles). These modules can be developed without any knowledge of the final surrounding system. They may even be provided by third-party vendors as closed-source components. Current implementations demonstrate that it is possible to enhance the standard partial reconfiguration vendor flow in a way that enables bitfile preservation. Moreover, the already established cooperations prove that bitfile preservation is highly relevant for contemporary FPGA industry.

Application:

With our industry partner we are working on a system, that enables even FPGA(PR) unaware engineers to design runtime reconfigurable image processing applications.

- Parts of the system can be **adjusted** by replacing modules **at runtime**.
- Modules can be built using the GUI or **delivered by third party vendors**.
- Modules can be **re-used** in **different applications** (different static systems) without any re-compilation.

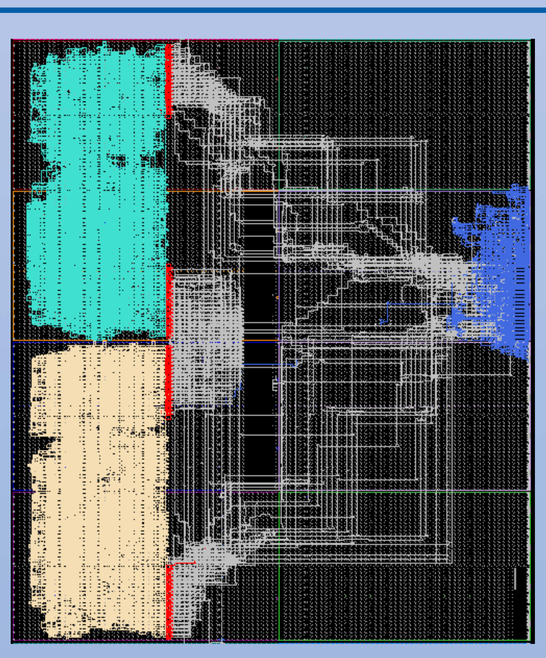
Need for pre-compiled “out of context” modules, that can be used in different designs without any re-compilation.



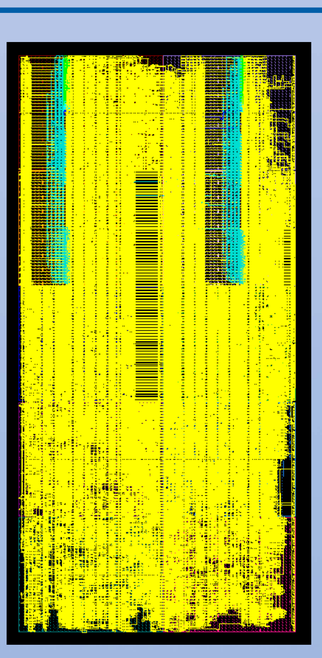
Prerequisites:

A few prerequisites need to be met in order to enable the use of partial bitfiles in different environments as well as their relocation.

- Using **Xilinx DPR flow**
- Equality of their **fabric footprint**
- Avoidance of **configuration frame sharing**
- Avoidance of **feed through routes** (*PRIVATE=ROUTE* option within the AREA_GROUP definition)
- **Interface Unification**: Placement constraints (*LOC*, *BEL* and *LOCK_PINS)* for the Proxy-LUT (within module) and decoupling register (outside module) of each signal. Routing between this two fixed anchor points is defined by directed routing (*DIRT*) constraints.

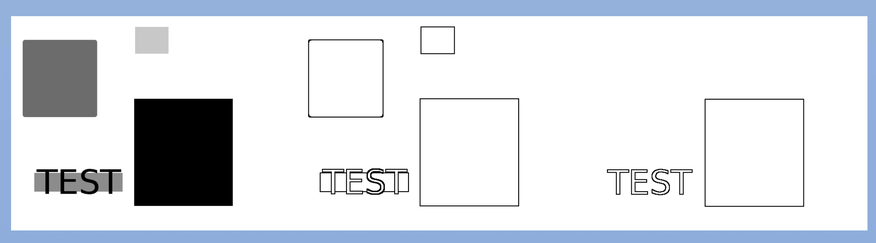


Virtex 6 LX240 – PCIe + 2 Module Test Design
(Fixed intervaces in red)



Virtex 6 LX130 – SiliconSoftware 2 Modules
Testdesign using *ROUTE=PRIVATE*

Results: Three modules (designed with Silicon Software’s VisualApplets) were compiled to partial bitfiles and dynamically loaded to different FPGA locations (and therefore to different stages of the processing pipeline) of a simple PCIe test design. These bitfiles could be re-used even if the static part of the design was changed.



a) Source image without noise
b) Only Sobel module loaded
c) Threshold (1) and Sobel(2) modules loaded

a) Source image with noise
b) Only Threshold module loaded
c) Gaussian (1) and Threshold (2) modules loaded

Next: Investigations according to different topologies (number, size and locations for PR modules) under consideration of the target platform, as well as flexible pipeline routing and zero frame drop reconfiguration.

ReCoSoC 2013

8th International Workshop on Reconfigurable Communication-centric Systems-on-Chip

Contact: stuellein@iri.uni-frankfurt.de