Deliverable – D18
IP Libraries

Grant Agreement No:	**214364**
Project acronym:	**GALAXY**
Project title:	**GALS InterfAce for CompleX Digital System Integration**
Funding Scheme:	**STREP**
Date of latest version of Annex I against which the assessment will be made:	**23.10.2007.**
Contractual Date of Delivery to the EC:	**30 Nov 09**
Actual Date of Delivery to the EC:	
Author(s):	**Lilian Janin (UNIMAN)**
Participant(s):	**UNIMAN**
Work Package:	**WP3**
Security:	**Public**
Nature:	**IP Libraries + Report**
Version:	1
Total number of pages:	12

Abstract:

This document describes deliverable D18 of the GALAXY project: Two IP libraries for use in the GALAXY tools.

The first library corresponds to task 3.2: implementing a library of GALS adapters that can be automatically instantiated in the IDE.

The second library corresponds to task 3.3: implementing a library of components that allow new users to easily build a functional asynchronous or GALS system. This library contains all the IPs needed in a tutorial for the GALAXY tools.

Function	***Responsibility***	***Date***	***Signature***
Written by:	Lilian Janin	30 Nov 2009	
Checked by:	Members of GALAXY Consortium		
Approved by:	-		

Reserved to EC

Approved by:			

CHANGE RECORDS

ISSUE	***DATE***	***§ : CHANGE RECORD***	***AUTHOR***
1	30-Nov-09	1st version	Lilian Janin

BIBLIOGRAPHIC RECORD

Project Number:	214364 GALAXY
Project Title:	GALAXY
Deliverable Type:	IP Libraries + Report
Deliverable Number:	D18
Contractual Date of Delivery:	30 Nov 2009
Actual Date of Delivery:	
Title of Deliverable:	IP Libraries
Work package contributing to the Deliverable:	WP3
Authors:	Lilian Janin
Abstract	This document describes deliverable D18 of the GALAXY project: Two IP libraries for use in the GALAXY tools. The first library corresponds to task 3.2: implementing a library of GALS adapters that can be automatically instantiated in the IDE. The second library corresponds to task 3.3: implementing a library of components that allow new users to easily build a functional asynchronous or GALS system. This library contains all the IPs needed in a tutorial for the GALAXY tools.
Keywords	GALS, asynchronous, software, tools
Confidentiality Level	Public
Name of Client:	EC
Distribution List:	GALAXY, EC, internet
Authorised by:	
Issue:	1
Document ID:	D18
Total Number of Pages:	12
Contact Details:	Lilian.janin@manchester.ac.uk

TABLE OF CONTENTS

1 INTRODUCTION

This document describes deliverable D18 of the GALAXY project: Two IP libraries for use in the GALAXY tools.

The first library corresponds to task 3.2: implementing a library of GALS adapters that can be automatically instantiated in the IDE.

The second library corresponds to task 3.3: implementing a library of components that allow new users to easily build a functional asynchronous or GALS system. This library contains all the IPs needed in a tutorial for the GALAXY tools.

2 REFERENCES

2.1 ACRONYMS

ASIP	Asynchronous-Synchronous IPs packaging format
FPGA	Field-Programmable Gate Array
GALS	Globally Asynchronous Locally Synchronous
GTK	GNU Toolkit
IDE	Integrated Design Environment
IP	Intellectual Property
VHDL	VHSIC Hardware Description Language

3 LIBRARY OF GALS ADAPTERS

This is the output of task 3.2: "Implement library of GALS interface IPs following the specified async IP format: multiple levels of abstraction, multiple languages for proper debugging in the IDE".

We refined this task based on the actual implementation of the tools. What was actually needed at this stage is a library of GALS adapters, which the IDE could use to automatically find an adapter when two components with non-matching interfaces need to be connected.

3.1 AUTOMATIC INSTANTIATION OF ADAPTER COMPONENTS

When the circuit designer attempts to connect two ports that do not match, or when two connected interfaces do not match anymore after some user manipulations, the IDE suggests an appropriate adapter component to be inserted between the ports.

The following port properties are used when checking for matching ports, and when looking for matching adapters:

- port type: asynchronous channel / synchronous net
- data width
- for nets: clock domain
- for asynchronous channels: handshake protocol

For synchronous nets, different cases can appear. As soon as an adapter is involved, the extra signals data_valid, adapter_full and the associated clock, are needed. We handle these signals by using port groups. The following port groups are recognised and used to match ports to adapters: "synchronous data in", "synchronous data out", "OCP master", "OCP slave".

The following cases are handled separately:

- If both ports are asynchronous channels: we check for handshake protocol and data width adapters.
- If both ports are synchronous nets, we insert clock domain adapters
- If one port is synchronous and the other port asynchronous we base our decision on the output of Workpackage 2 (Deliverable D3), where the following adapters were identified as being useful in a GALS context:
    - Synchronisers
    - FIFO GALS interface
    - GALS wrappers with pausible clocking

3.2 LIBRARY

The library of adapters is called *lib_automatic_adapters* (file: lib_automatic_adapters.asip.xml). It is required by the Galaxy IDE, and is also visible in the component library view, where components can be drag&dropped into the current design for manual instantiation.

The components are: asynchronous protocol converters, channels and wires' width adapters and synchronous-asynchronous adapters. Each of these components has a description of both sides of its interface (inputs and outputs), which the IDE automatically searches for when two connected interfaces do not match in the current design (usually after some user manipulations that modify one component interface).

The components are:

- **adapter_GALS_gray_fifo_S_A**: net+clock_domain → asynchronous channel
- **adapter_GALS_gray_fifo_A_S**: asynchronous channel → net+clock_domain
- **adapter_GALS_synchronizer_S_A:** net+clock_domain → asynchronous channel
- **adapter_GALS_synchronizer_A_S**: asynchronous channel → net+clock_domain
- **adapter_GALS_pausible_clocking**: net+clock_domain1 → net+clock_domain2
- **adapter_net_width**: net+width1 → net+width2
- **adapter_channel_width**: channel+width1 → channel+width2
- **adapter_channel_single_rail_dual_rail**: channel+protocol1 → channel+protocol2

3.3 Current limitations

Parameterised components are not yet supported by the ASIP format. As port width is an unbounded integer, this makes it harder to include port width adapters for any width. In the current release, only port width adapters useful in our demo examples are provided, but the user can easily add new port width adapters if required.

Only two handshake protocols are supported: single rail and dual rail (1-of-4 is manually implemented in the G3card demo with Silisitix' CHAIN interconnect).

4 LIBRARY OF ASYNCHRONOUS IP

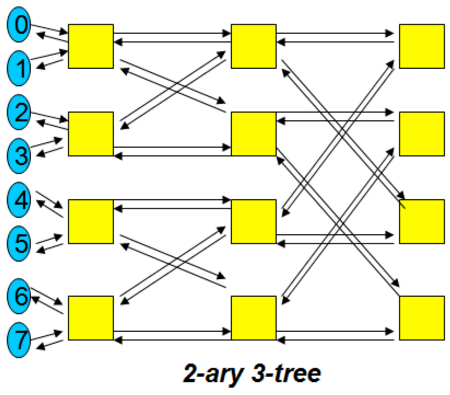
This is the output of task 3.3: "Implement library of asynchronous IPs (3.3) allowing new users to easily build a functional asynchronous or GALS system. Very small library limited to components needed in a tutorial. Expected to grow by encouraging community sharing of IPs via a website (similarly or in collaboration with to opencores.org)".

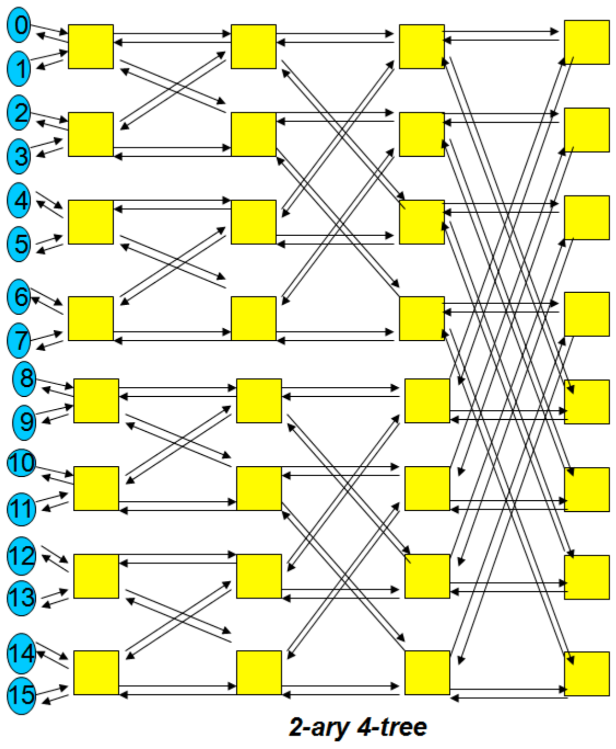
The components are distributed over two libraries (plus the *lib_xpipes* library containing the NoC building blocks): *lib_demo_components* and *lib_xpipes_nocs*.

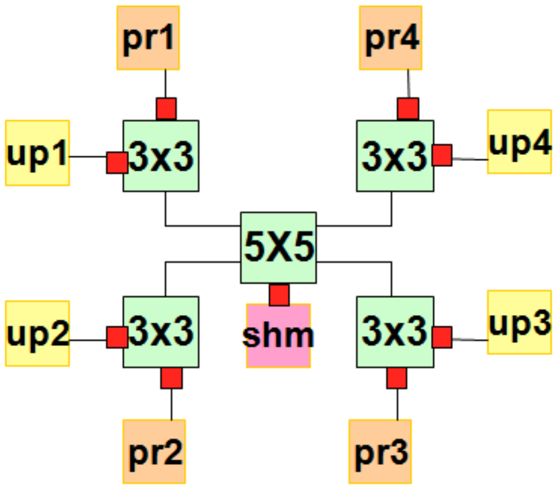
The following components are provided:

- **CalculatorCore**, described at the following abstractions and languages:
    - Transaction-level SystemC
    - Pin-level SystemC
    - Synthesisable Verilog
    - Synthesisable VHDL
    - Balsa (can be simulated at the Balsa level or synthesised as Verilog netlist)
- **JimsFpgaBoardKeyboard**: keyboard component described at the following levels of abstraction:
    - Behavioural C with GTK graphical interface (representing a keyboard) able to run on a computer host
    - Transaction-level SystemC linked to the same GTK interface
    - Synthesisable Verilog probing the keys of a real hardware keyboard, intended to be used on an FPGA board developed at UNIMAN (developed by Dr Jim Garside, hence the name of the component).
- **JimsFpgaBoardKeyboardInterface**: handwritten adapter between the FPGA bus and **JimsFpgaBoardKeyboard** component. Implementations:
    - Transaction-level SystemC
    - Synthesisable Verilog
- **JimsFpgaBoardLCD**: Same as keyboard but for LCD. Behavioural C with GTK interface, Transaction-level SystemC and synthesisable Verilog.
- **JimsFpgaBoardLCDInterface**: Same as keyboard interface but for LCD. Transaction-level SystemC and synthesisable Verilog.
- **AsipRouterHost**: Component described in SystemC+C, able to translate SystemC TLM transaction to the HLA backend, by following the ASIP protocol.
- **AsipRouterARM**: Component described in ARM assembly language, able to translate communications following the ASIP protocol from the serial port to the board's bus.
- **AsipRouterVirtex**: Component described in Verilog, able to translate communications following the ASIP protocol from the board's bus to the FPGA.
- Four pre-built NoC topologies for use with XPipes: **2-ary 4-tree**, **2-ary 4-tree,** an **application-specific** topology for our demo, and **3-ary 2-mesh** topology (other

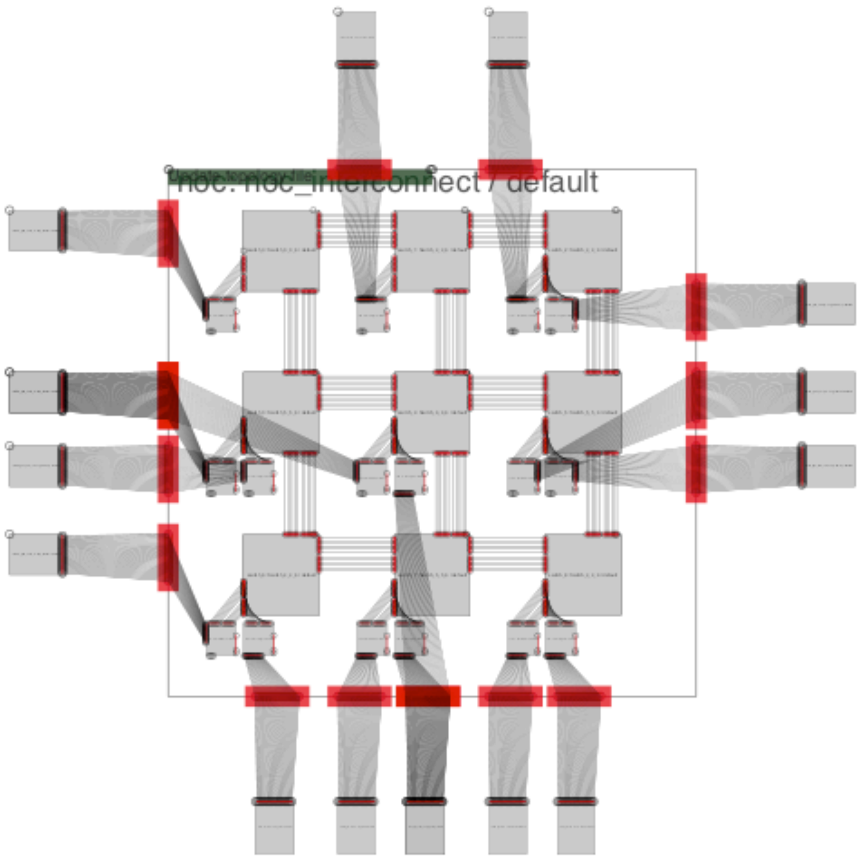
topologies can be constructed manually by using the building blocks provided in lib_xpipes).







Application-specific topology. Red blocks are network interfaces. *"pri"* stands for private memory *i*, *"upi"* stands for processor core *i*, *"shm"* stands for shared memory.
(NoC diagrams provided by UNIBO)



3-ary 2-mesh NoC topology, as seen in GALAXY IDE