Public with confidential appendices (Public version)

ASAM

Grant agreement no. 100265

Artemis Project

ASAM

Automatic Architecture Synthesis and Application Mapping

D 3.4: Method of and report on interconnects and memory structures construction

Due Date of Deliverable	**30th April 2011**
Completion Date of Deliverable	**21st April 2011**
Start Date of Project	**1stMay, 2010 – Duration 36 Months**
Lead partner for Deliverable	**UNICA**

Revision: v2.0

Project co-funded by the Artemis Joint Undertaking Call 2009		
Dissemination Level		
PU	Public	
PU/COA	Public with confidential appendices	x
PP	Restricted to other program participants (including Commission Services)	
RE	Restricted to a group specified by the consortium (including Commission Services)	
CO	Confidential, only for members of the consortium (including Commission Services)	

1 Table of contents

2 Introduction

This deliverable reports about the first year of activity performed within Task 3.4. Within this task, a design process is under development that aims at development of optimized interconnection and memory structures for a multi-ASIP system via iterative refinements. The iterative modifications of interconnect and memory structures are assumed to be driven by the constraints and objectives decided by the macro-architectural level. In particular, parameters such as memory capacity and word-length or physical interconnect bit-width, etc. can only be specified by the macro-architecture synthesis process after the application is distributed among several ASIP processors and the ASIP processors are actually designed. Based on the information provided by the macro-level and by successive evaluations of candidate interconnect and memory architectures performed by the lower-level prototyping infrastructure (developed within WP4 and described in deliverable D4.1), the interconnection and memory structures are created and refined. At the end, feedback consisting of the created interconnection and memory structures, as well as, their parameters is provided to the upper level, as depicted in Figure 1.

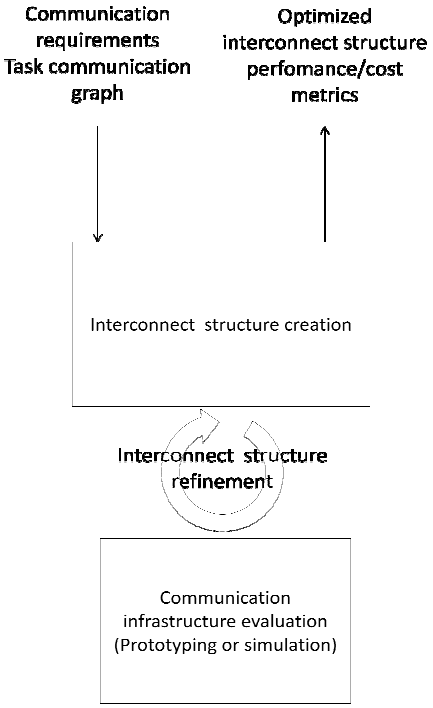

Figure 1: Prospective overview of the interconnect optimization process

During the first year, to this aim, we worked on providing the possibility of instantiating and evaluating a Network-on-a-Chip (NoC) topology as a communication substrate for a system, to be used as default interconnection fabric and then iteratively refined. Thus, the main outcomes of such activity are:

- the integration of a customizable NoC component library with Silicon Hive's components;
- Its extension aimed at supporting the computation model that will be used in the project;
- Its extension to provide support for fast prototyping, that is exploited within the evaluation platform described in deliverable D4.3;

- The instrumentation of the ASIP cores to enable evaluation of performances and bottlenecks related with its internal memories and interconnection networks;

Next year of activity will involve the definition of the algorithm driving the iterative refinement of the communication hardware structures and its integration with the rest of the environment.

3 Creation and instantiation of the communication modules

The Xpipes NoC component library, highly flexible library of component blocks, has been chosen as baseline reference for the development activity.
The library is suitable for the creation of arbitrary topologies, thanks to the capability of its modules of being almost completely configured at design time.
Xpipes, natively, includes three main components:

- switches,
- network interfaces (NIs),
- links.

The backbone of the NoC is composed of switches, whose main function is to route packets from sources to destinations. Arbitrary switch connectivity is possible, allowing for implementation of any topology. Switches provide buffering resources to lower congestion and improve performance. In Xpipes, both output and input buffering can be chosen, i.e. FIFOs may present at each input and output port.

Switches also handle flow control issues, and resolve conflicts among packets when they overlap in requesting access to the same physical links.

A NI is needed to connect each IP core to the NoC. NIs convert transaction requests/responses into packets and vice versa. Packets are then split into a sequence of flits before transmission, to decrease the physical wire parallelism requirements. In Xpipes, two separate NIs are defined, an initiator and a target one, respectively associated to system masters and system slaves. A master/slave device will require an NI of each type to be attached to it. The interface among IP cores and NIs is point-to-point as defined by the OCP 2.0 specification, guaranteeing maximum reusability and compliancy with the interface standards specified in Deliverable D2.1.

NI Look-Up Tables (LUTs) are used to specify the path that packets will follow in the network to reach their destination (source routing). Two different clock signals can be attached to NIs: one to drive the NI front-end (OCP interface), the other to drive the NI back-end (Xpipes interface). The Xpipes clock frequency must be an integer multiple of the OCP one. This arrangement allows the NoC to run at a fast clock even though some or all of the attached IP cores are slower, which is crucial to keep transaction latency low. Since each IP core can run at a different divider of the xpipes frequency, mixed-clock platforms are possible.

Inter-block links are a critical component of NoCs, given the technology trends for global wires. The problem of signal propagation delay is, or will soon become, critical. For this reason, xpipes supports link

pipelining, i.e. the interleaving of logical buffers along links. Proper flow control protocols are implemented in link transmitters and receivers (NIs and switches) to make the link latency transparent to the surrounding logic. Therefore, the overall platform can run at a fast clock frequency, without the longest wires being a global speed limiter. Only the links which are too long for single-cycle propagation will need to pay a repeater latency penalty.

Within Task 3.4, extensions to original Xpipes library were developed for adaptation and integration in the project, and to provide advanced communication capabilities required for hybrid (message-passing and shared-address space) model of computation and for fast prototyping.

Here follows a list of the main features that have been added to the library.

- Capability of initializing and handling DMAs (direct memory to memory transfers)
- Support for message passing model of computation
- Runtime reconfiguration of the routing strategy
- Insertion of performance counters inside NoC modules
- Switch-bypassing capabilities inside the switches

All the mentioned additional features required some modifications of the processor-to-NI interface circuitry.

Several dedicated adapters have been developed in this aim, allowing at the same time the seamless integration of the xpipes library (natively compliant with OCP) with the rest of the environment.

- programmable CIO-to-OCP adapter (CIO is Silicon Hive's proprietary protocol developed to allow host-cell and cell-cell communication inside the system);
- programmable LMB-to-OCP adapter (LMB is Xilinx's Local Memory Bus, used as interconnect for Xilinx FPGA-based embedded processor systems. LMB is the interface exported by the MicroBlaze processor instruction and data ports, to connect to on-chip block RAM, and is necessary when the Microblaze will be used as control processor during prototyping and emulation);

All these wrapper versions include some memory-mapped registers, that can be written by the connected processing element to set the operation mode for the communication actions that cannot be directly initiated by a dedicated instruction in the processor instruction set. For example, in case of message sending, the registers are written to set the source address, the size and the tag assigned to the required message.

Some address decoding logic is added inside the core in order to detect those load/store operations that are not intended to generate traffic over the network, such as accesses to memory mapped registers, to performance counters, to reconfiguration circuitry.

The functionality provided by the wrappers co-operate with other modules that have been developed from scratch within Task 3.4:

- programmable OCP to memory adapter: allows interaction between network and CIO or LMB slave devices. devices or between network and LMB devices; according to the programming of memory-mapped resources inside the wrapper, the Network Interface can access directly the memory. A portion of every local memory of every processor serves as buffer for temporary storing of communication messages;
- a master-and-slave (initiator-and-target) network interface: this module, added to the original xpipes library is used when message passing and memory-to-memory transfers have to be supported, to make network node capable of performing the operations related to both communication ends;
- hardware counters have been instantiated at the core interface and inside the switches, to detect the activity metrics needed to use the models described in Deliverable D3.1. This counters are needed to evaluate in detail the performances achievable with every candidate architecture.
- routing LUTs programmability: the LUTs that contain the directives related with the routing strategy for every source-destination pair in the system have been made programmable by local connected core or remotely, through the use of a special packet. This capability is used in WP 4 to enable runtime reconfigurability, as explained in Deliverables D4.3 to speed-up the prototyping process.
- programmable circuitry inside the NoC switch, enabling to establish a direct combinational path between the input and the output port of the switches, to avoid accounting for unneeded latencies when mapping multiple candidate NoC topologies onto the same prototyping platform. This feature is needed to reduce the overhead due to synthesis and implementation flow when using the FPGA-based platform for DSE purposes.

Prospectively, within next year of activity, the possibility of conveniently replacing specific paths inside the NoC topologies with bus-based communication infrastructures will be evaluated. To enable this possibility, adequate protocol converters have been implemented, since NoC refers to the OCP protocol while the bus infrastructure and the Silicon Hive's processors implement the Silicon Hive's proprietary CIO protocol.
Several bus-based communication structures can be instantiated directly at micro-architectural level.

Single-layer CIO bus

The single-layer CIO bus is a bus structure. The bus has a single arbiter, arbitrating between all the masters that request access to the bus. The bus has configurable data and address widths. The only supported protocols are Silicon Hive's proprietary Regular and Extended CIO protocols.

Multi-layer CIO bus

The multi-layer CIO bus is a bus structure. Each slave has an arbiter, arbitrating between different masters that request access to a slave. Thus, multiple masters may be active simultaneously, as long as they do not try to address the same slave. The bus has configurable data and address widths. The only supported protocols are Silicon Hive's proprietary Regular and Extended CIO protocols.

4 Memory blocks HDL creation and memory-related optimization

Task 3.4 envisions a micro-architectural optimization related to the memory blocks inside the system. In the ASAM project platform, memory-related optimization will mainly tackle internal program and data memories included in the ASIP processor architectural template. During the first year of activity, the generation of the HDL of such memories has been integrated inside the FPGA-based evaluation environment. The signals that are relevant for counting memory accesses and to trace the activity figures needed to estimate the power consumption, have been identified and connected to dedicated counters during preliminary experiments. This will enable during next year of activity the use of the feedback provided by the evaluation platform for the optimization of the memory blocks inside the system. The HDL code of the ASIP processors has been also instrumented, by means of the integration in the design flow of new custom tcl scripts, to export in output the signals related with the activity of internal bus structures in the Processor Architecture Template. In this way, it will be possible, in the next research steps, to easily and effectively estimate the benefits and the costs related to different internal communication architectures. Namely, activity figures can be extracted for the Argument Select Network and for the Result Select Network, implementing respectively the connection from register files output ports to issue slot inputs and vice-versa.