HINOC: A HIERARCHICAL GENERIC APPROACH FOR ON-CHIP COMMUNICATION, TESTING AND DEBUGGING OF SOCS

Thomas Hollstein, Ralf Ludewig, Heiko Zimmer, Christoph Mager,
Simon Hohenstern, Manfred Glesner

Darmstadt University of Technology
Institute of Microelectronic Systems
Karlstrasse 15, D-64283 Darmstadt, Germany
{hollstein|ludewig|zimmer|glesner}@mes.tu-darmstadt.de

Abstract This paper presents a new generic system architecture and design methodology for the design, debugging and testing of complex systems-on-chip (SoC). Starting from a hierarchical generic system architecture, platforms for dedicated application scenarios will be customized. In order to be able to handle very complex submicron designs, the system is based on a globally asynchronous and locally synchronous (GALS) concept. The problem of the increasing functionality versus outer access capabilities ratio is faced by novel embedded and combined debugging and test structures. The integration of debugging possibilities is essential for an efficient co-design of SoC integrated hardware and software, especially for systems with integrated reconfigurable hardware parts.

Keywords:
Networks-on-Chip, Silicon Debug, Built-In Self Test

1. Introduction

Increasing system complexities, driven by ongoing technology improvements and application requirements, create a demand for highly efficient system design methods. Furthermore time-to-market becomes the primary factor for economical success ([Jager, 2002]). In order to gain the required design productivity, design reuse and efficient behavioral synthesis methods are indispensable. Advanced methods for low power design will be essential for an increasing number of battery-driven applications.

Currently complexity is mainly handled by *intellectual property (IP)* based approaches. Existing design components are used in several projects and frequently designs are enhanced and completed by buying IP components from

Please use the following format when citing this chapter:
Hollstein, Thomas, Ludewig, Ralf, Zimmer, Heiko, Mager, Christoph, Hohenstern, Simon, Glesner, Manfred, 2006, in IFIP International Federation for Information Processing, Volume 200, VLSI-SOC: From Systems to Chips, eds. Glesner, M., Reis, R., Indrusiak, L., Mooney, V., Eveking, H., (Boston: Springer), pp. 39-54.

external suppliers. At present the effort for the integration of external IPs (IP components) into SoC designs can be high, especially if wrappers have to be developed in order to adapt the component's ports to the system's communication architecture and protocols. A common way to cope with the IP block integration problem are platform-based design approaches, where a fixed or limited generic communication architecture is presumed. These platforms are adapted and scaled in every design generation to fulfill the currently required system constraints (incremental adaption). The advantage of state-of-the-art platform-based design approaches is that the migration effort between different design generations is limited. Major changes of the platform architecture are in most cases complex and very costly. The integration of IP components may be difficult if the components are available on register transfer level or on layout level, since the clocking scheme of the IP block may differ considerably from the timing of the blocks designed inhouse.

On an long term view, efficient system-level synthesis methods will be needed as well as IP-based design. In the recent decade synthesis methods have been presented on different levels of abstraction. Behavioral synthesis tools, mapping behavioral specifications on register-transfer level are in the meantime commercially available. Since the target space of high-level-synthesis is extremely huge, improved methods are currently still under construction. On system level several codesign methods for combined hardware-/software systems have been presented. Compared to software synthesis (compilation), automatic hardware transformations are much more complex, since the number of design parameters is comparatively high (area, performance, power) as well as the extent of implementation possibilities. Nevertheless in future synthesis methods will be a helpful complement to reuse approaches.

Concerning future SoC design challenges the following aspects have to be considered (SIA roadmap 2001 [SIA, 2001]): for large chip die sizes it won't be possible to transmit signals over long on-chip distances within only one clock cycle. Additionally it cannot be assumed any more, that all SoC components are running with the same clock frequency and phase. This results in a demand of communication-centric design. Communication structures with asynchronous communication of synchronously operated SoC components (Globally Asynchronous, Locally synchronous (GALS)) have to be developed in order to face the requirements of huge SoC systems, comprising a large number of components([Muttersbach et al., 2000, Sgroi et al., 2001]). Furthermore the functional density (amount of functionality related to the port width for outer access (e.g. number of pins)) of SoCs is permanently increasing. Therefore an outer access (controllability and observability) for manufacturing tests is becoming more difficult. Furthermore the debugging of heterogeneous SoCs, especially in the case of malfunction, is difficult.

The aspect of power minimisation is not focused in this paper. For an description of our investigations concerning SoC low power communication we'd like to refer to [Murgan et al., 2003].

This contribution is organized as follows: in the next section future SoC design challenges based on ongoing research of different groups are analysed in detail. Based on this analysis in section 3 a novel network-on-chip (NoC) communication architecture for future SoCs is presented. Section 4 describes a new concept of embedding combined on-chip debugging and testing structures. Further on in section 5 simulation results of the SystemC NoC-model are presented and discussed with respect to packet-oriented extensions. A new design flow for NoC-based HW/SW Codesign is presented in the last section. Finally we end up with some conclusions and an outlook on future investigations.

2. Design Challenges

SoC/NoC Design

Due to permanently increasing technology capabilities future on-chip integrated systems will be composed of a large number of subcomponents. As stated in the introduction, a central clock supply won't be feasible any more and a lot of heterogenous components have to be interconnected. Therefore global intermodule connections will become asynchronous, connecting synchronously operating subdomains (GALS structure). New methods are required to ensure the testability of such systems: locally embedded and independently operating testing structures for each synchronous subdomain with a central control and evaluation of the results.

Generally, future SoC design methodologies have to face the following aspects:

- system specification/modeling
- system simulation
- design methods for components (IP-based design, synthesis approaches)
- overall system architecture and communication structure
- design verification
- debugging capabilities
- testability concepts

The design of SoC communication architectures will be a central task which has to face the following demands:

- general approach, which is generic and scalable
- customizable to any kind of application topology
- adaptability to different application classes
- data transmission between different clock domains (synchronisation)
- flexible use and configurability of communication resources
- coverage of quality of service (QoS) aspects
- provision of required communication services

In future the on-chip communication will become more vulnerable to transmission faults. Decreasing feature sizes and voltages let single event upsets occur more frequently. Therefore networks-on-chip (NoC) have to be designed fault tolerant [Dally and Towles, 2001, Zimmer, 2002].

Generally for the design of NoCs, existing know-how from the field of computer networks can be evaluated. But in difference the topology for NoCs is fixed, the parameters of interconnections are known, and the amount of buffer memory in network nodes is limited. Therefore it seems to be reasonable to have several local merges in adjacent OSI protocol layers.

In the context of current research investigations several approaches for the structuring of SoC communication networks have been proposed. In [Wielage and Goossens, 2002],[Sgroi et al., 2001],[Benini and Micheli, 2002] and [Dally and Towles, 2001] basic NoC issues are discussed. Several proposals for SoC architectures have been presented: NOSTRUM [Kumar et al., 2002], SOCBUS [Wiklung and Liu, 2003], PROTEO [Saastamoimen et al., 2002], SPIN [Benini and Micheli, 2002], PROPHID [Guerrier and Greiner, 2000], MESCAL [Sgroi et al., 2001], a RAW (MIT) [Taylor et al., 2002], torus architecture (NTNU) [Natvig, 1997] and HiNoC [Hollstein et al., 2003] (among others).

In the following section we present a new NoC approach, which combines a GALS concept with a new hierarchical multilayer network structure. In contrast to other approaches, it allows the generation of heterogeneous architectures and the integration of any kind of existing design platforms based on a constant grid size communication approach.

Testing and Debugging

To ensure the correct function and in order to detect manufacturing errors it is indispensable to test an ASIC after fabrication. This is done by loading test patterns in the internal registers, performing one clock cycle and comparing the resulting pattern with the simulation results (Fig: 1). Therefore the internal registers are replaced by special scan registers that are linked together as a shift register in a so called scan path. In the traditional approach the sink and

the source of the test patterns are realized off-chip. This requires the use of external Automatic Test Equipment (ATE). Special attention has to be paid to the generation of the test pattern necessary achieve a good fault coverage.

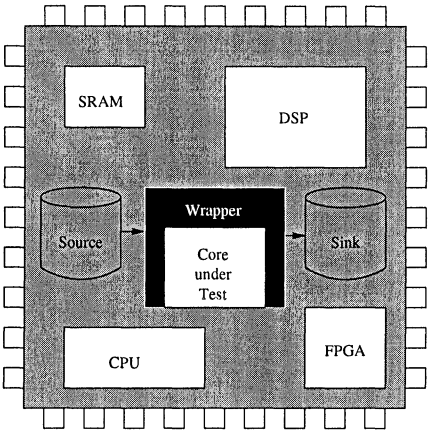

Figure 1. SoC Testing

Due to the shrinking feature size that allows for higher complexity and higher clock frequencies the requirements for the ATE speed and memory size are continuously increasing. Therefore the ATE solution can be unacceptable expensive and inaccurate. Thus on-chip test solutions, often referred to as build-in self-test (BIST), are becoming more and more popular.

Three additional components have to be implemented on the chip in order to perform BIST: a test pattern generator, a test response analyzer and a BIST control unit. The classical way to implement a test pattern generator is to use linear feedback shift registers to generate pseudo-random test patterns. In the test response analyzer the results from the applied test patterns are analyzed and compared to a pre-stored table of valid results. Due to some drawbacks like low fault coverage, long test application time and high area overheads, several proposals have been made to combine these pseudo-random tests with deterministic patterns [Jervan et al., 2002].

BIST allows at-speed tests and eliminates the need for an external tester. It cannot only be used for manufacturing test, but also for periodical field maintenance tests.

For complex systems it is also necessary to add debugging capabilities to the programmable components. For embedded system debugging the IEEE-ISTO Nexus 5001 Forum Standard defines a "Global Embedded Processor Debug Interface" [Nexus 5001 Forum, 1999]. The following basic needs when debugging an embedded processor are defined in the standard:

- Logic Analysis:
  - Access instruction trace information
  - Retrieve information on how data flows through the system
  - Assess whether the embedded software is meeting the required performance level
- Run Control:
  - Query and modify the processor state/registers when the processor is halted
  - Support breakpoint/watchpoint features in hardware and software

Prior to this, some JTAG-based approaches were published (e.g. [de la Torre et al., 2002] and [Jung et al., 2002]). A comparator is connected to the address and data bus to support break points and a simple controller is added for single step operation mode. The data of the internal registers of the processor can be shifted out and in by using the scan chain from testing. This technique requires only a small amount of additional hardware, but does not fulfill all requirements (like trace informations) listed above.

For reconfigurable components the same requirements are valid as for embedded processors. For FPGA-prototyping a commercial solution is available [Synplicity, 2003], which uses the resources of the reconfigurable unit to provide the debugging capabilities. It is also possible to use hardware-based debugging and facilitate only the routing resources to get access the internal signals.

For future SoCs it will be necessary to provide debugging units for all reprogrammable/reconfigurable units. Linking this units to each other and thereby enable the units to communicate with each other will greatly improve the debugging performance of the development engineers.

3. HiNoC: A Novel Network-on-Chip Architecture

In recent times simple communication structures have been used within SoCs, since the number of connected components has been small [Sgroi et al., 2001]. Future applications will comprise of a large number of heterogeneous processing units, which have to be interconnected efficiently [Kumar et al., 2002].

We have developed a new hierarchical NoC topology which is capable to implement GALS systems in a very efficient way. The system is, in difference to [Kumar et al., 2002], structured in two levels (Fig. 2).

The top level is an asynchronous mesh structure. This topology is a special case of a general torus structure, which can be implemented more efficiently (smaller and similar delays between routing units).

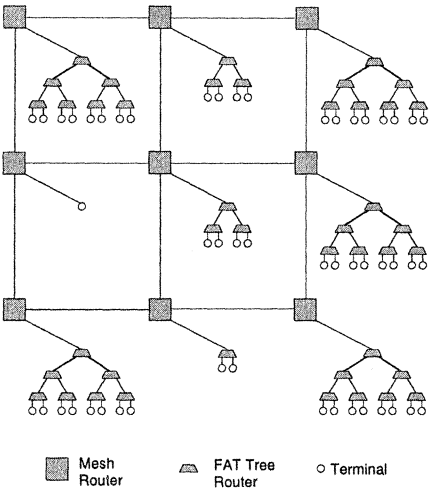

Figure 2. HiNoC: Hierarchical Topology of Communication Network

The general HiNoC concept is to have a basis for an automatic generation of application platforms. Based on a system-level application simulation, a customized mesh will be generated by leaving out several of the mesh nodes. The resulting structure will allow to interconnect applications comprising components with heterogeneous granularity (Fig. 3).

Adjacent routers in the HiNoC (Hierarchical NoC) mesh are connected by two unidirectional connections (one in each direction) with generic bitwidth. Furthermore the asynchronous communication is handled by a handshake protocol. Each network router (as far as it is not an network edge router) is connected to four adjacent routers and to a second layer node. This second level of the hierarchy is formed by FAT trees with generic depth. Each FAT tree is in a general case a binary tree which is capable to provide a full connect of all its leaf terminals (connection to processing nodes). In our implementation the connectivity of of the FAT tree can be thinned out in order to have a trade-off between full connectivity and the amount of required wires and multiplexers. FAT trees are a very cost efficient topology for implementing VLSI systems ([Leiserson, 1985]). The communication within the FAT tree is completely synchronous and it has an asynchronous port towards the level one network router. In the special case that the FAT tree consists of one leaf node only, this node is directly attached to the mesh router. All FAT trees connected to mesh routers may have different sizes and different thin out factors. In difference to the definitions in [Tanenbaum, 1996], the HiNoC terminal concept basically in-

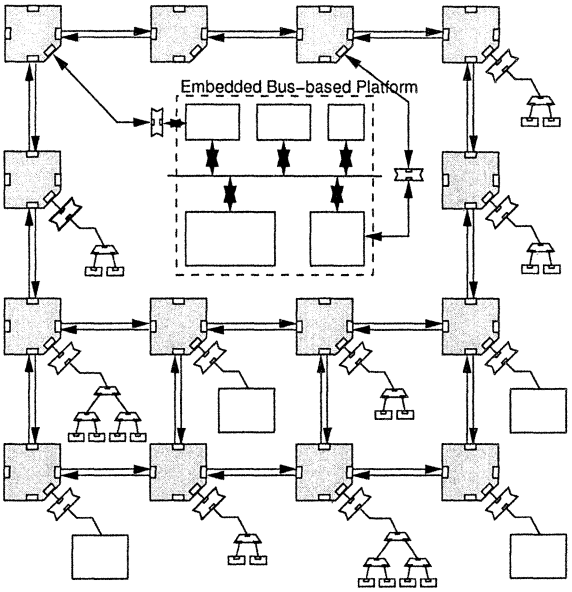

Figure 3. Customized Mesh (FAT trees are not depicted for the sake of simplicity)

cludes the possibility of multiple terminal connections of processing elements to different FAT tree leaf terminals (Fig. 3).

Of course, the FAT tree can be replaced by any kind of proprietary communication architecture (e.g. crossbar or buses) or complex components. Existing design platforms can be integrated completely as NoC attached components. If the size of an embedded platform exceeds the area defined by the router distances, one or several routers can be left out in order to integrate the platform component.

HiNoC offers two data transmission modes, which can be interleaved in order to gain best use of available communication resources:

Packet-Switched Communications: Packets are routed by a local routing algorithm (local connections required only) and transmitted by application of a virtual cut-through / wormhole switching scheme. Stucked packets are resolved and backtraced to the packet source. Packet transmission uses the available network links efficiently at the cost of a non-deterministic latency.

QoS-based Communications: if circuit-switched data transmission is applied, then dedicated virtual communication channels are set up with the full link bandwidth or any binary divided portion of it. This mode provides

Quality-of-Service based communications, which are required for multimedia applications (e.g. video streaming). The setup of the connection is done by a control packet which is sent towards the stream target router, reservating the requested bandwidth on each link (if possible). Having reached its sink, the packet is back-traced to its origin and activating the channel on each passed mesh link. If a connection has been set up, the bandwidth is guaranteed for the full transmission of a data stream (Quality-of-Service, QoS).

The routing is done based a combined static and dynamic metric. The static information states, which target node of the mesh can be reached via a specific mesh router output link at which metric (distance). If parts of the SoC/Mesh are powered down (power reduction), the static routing tables are updated in order to adapt the routing behaviour to the new NoC configuration. Additionally adjacent mesh routers exchange information about the current traffic situation by sending control packets in free time slots (dynamic metric).

Fig. 4 depicts the basic functional schematic of the first version of a level one mesh router.

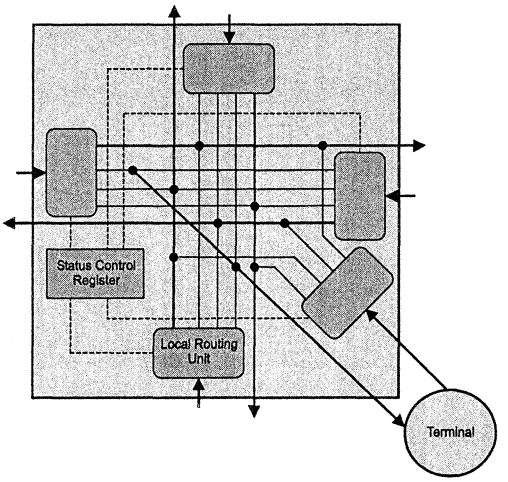

Figure 4. HiNoC Mesh Router Structure

The input FIFO can be enhanced in order to implement several priority levels. In an advanced implementation this input buffer design has been migrated to a middle buffer design with virtual channels, which are groupwise assigned to one of the potential four output connections.

For the routing of streams and packets an efficient local routing algorithm is applied, which also considers the possibility of routing around high density spots and which is avoiding cycles.

The mesh and FAT tree models are implemented in SystemC and the simulator is writing a log file on all network events. The log file can be processed by a JAVA based graphical viewer and stepped through offline.

4. Integrated Test and Debugging Concept for SoCs

Comparing the requirements for build-in self-test and debugging support for SoC components, it becomes obvious that the main requirement, the access to the internal registers, is the same. Furthermore running a self test is possible only if the tested component is not in an operational mode. For running the test procedure it has to be in a special system maintenance mode. On the other hand debugging only make sense when the component is in normal operational mode. Thus we have developed an integrated hierarchical concept, which benefits from resource sharing for both tasks. The method provides controllability and observability over several levels of hierarchy of the SoC topology.

Having a test/debug unit connected to internal registers and external input/output signals of SoC components, additional applications besides build-in self-test and debugging are possible:

- On-chip signal analysis: The interface signals of a component can be analysed to extract statistical data from the communication. This can be used for power estimation and optimization by measuring the activity and calculating the temporal and spatial correlation of the interface signals [Ludewig et al., 2002].
- Event logging: Special events could be logged during the whole period of operation. This can be used for mission critical components (e.g. in medical devices).
- Dynamic system reconfiguration: additional embedded memory for test and debug purposes can also be used to store configuration data for (partial) system reconfigurations, which are triggered by dedicated events. The reconfiguration can be performed without interfering with other components of the SoC.

If the embedded test and debug units are interconnected by global network-on-chip communication resources, new collaborative functionalities become possible. For instance an embedded processor could be stopped if a breakpoint condition in a reconfigurable component is reached.

For SoCs with more than one clock domain a hierarchical structuring of the interconnection of the test and debug units is proposed, so that all of these

units which are located in one clock domain are connected to a communication controller that implements the communication with controllers in other (potentially asynchronously running) clock domains. This also reflects the architecture of the proposed HiNoC architecture in which the components connected to one FAT-tree are clocked synchronous, while the components connected via the switched network could belong to other clock domains. Therefore one communication controller has to be integrated in every FAT tree/synchronous domain.

5. Results

The goal of a first SystemC implementation of the HiNoC communication architecture had been the analysis of the setup and the achieved bandwidth of connection-oriented data transmissions (build up connection, transmit data, cancel connection). Therefore first we instantiate a 4x4 mesh within our generic SystemC model and set the height of the FAT tree to zero. In order to stress the net, all network terminals permanently try to send 500 data words to a randomly selected destination address. The following parameters have been analysed: the average relative bandwidth (active data transmission time) of all terminal's sending processes:

$$t_{av,rel} = \frac{1}{n_t} \sum_{i=1}^{n_t} \frac{t_{send,i}}{t_{measure,i}} = \frac{1}{n_t} \sum_{i=1}^{n_t} t_i \tag{1}$$

and the minimum relative bandwidth $t_{min,rel}$ over all i nodes. n_t represents the number of terminals (in our case 16). Furthermore we have been tracking the average waiting time $t_{wait,av}$, which nodes had to wait (because of blocked resources) after finishing a transmission until they could start the next data send cycle. $t_{wait,max}$ stands for the maximum waiting time which occurred during the whole simulation. Fig. 5 depicts $t_{av,rel}$ under the assumption, that the transmission of 500 data words is clustered to 5 or 25 blocks of equal size, which are independently transmitted. The partitioning of the transfer reduces the $t_{wait,max}$ values by 40% to 50%. Of course $t_{av,rel}$ is decreased, since much more connection setups have to be processed (overhead). If the mesh size is varied, the time parameters show an expected quadratic dependency (Tab. 1).

The modification of the maximum waiting time is increasing correlated to these results. Of course the experiment is very pessimistic, since a reasonable mapping of intensively communicating terminals on adjacent positions on the grid can reduce the problem dramatically. Further improvement can be achieved by enhancing the concept to a two layer NoC architecture with synchronous subdomains with synchronous intra module communication using the FAT tree structure. For heavily communicating pairs of terminals an assignment to the same synchronous domain generates a huge gain in speed, since

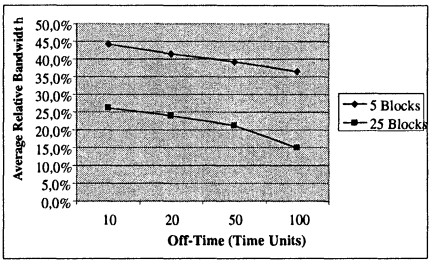

Figure 5. Measured Average Bandwidth for different Terminal Parameters

Mesh size	tav, rel	$t_{min,rel}$
4x4	41,6%	35,0%
6x6	33,1%	21,4%
8x8	24,7%	7,2%
10x10	18,2%	1,0%

Table 1. Influence of the Mesh size

synchronous communication is much faster than asynchronous data transmission (requires a certain effort to avoid metastability of registers). Table 2 compares the simulation results of two different 64 terminal network-on-chip configurations. The first configuration is a 4x4 mesh with four terminal FAT trees

Mesh size	$t_{av,rel}$	$t_{min,rel}$
4x4x4	50,0%	21,9%
8x8x0	24,7%	7,2%

Mesh size	Average max. wait time	$\max(t_{wait,max})$
4x4x4	1500	7230
8x8x0	5686	17657

Table 2. Simulation results: pure mesh versus two-level HiNoC architecture

attached to each mesh router. Additionally the assumption is applied, that 75%

of the communication can be routed within a FAT tree (simulation of intelligent mapping). The second topology is a 8x8 mesh without FAT trees. For the given assumption the resulting average bandwidth in the 4x4x4 topology is 50% and the maximum waiting time is comparatively low.

6. The proposed Dynamic Co-Design Flow

Fig. 6 depicts a new design flow based on communication-centric hyper-platforms. In the static design phase (the SoC hardware design phase) a set of applications to be run on the target system is profiled by simulation. Based on the profiling information, a design constraint library and an existing component implementation library, a set of required execution components is determined (Mapping Step). In a second Placement Step the required IP components are placed on the hyper-platform and the NoC is customized, considering the communication closeness of the IP blocks. After manufacturing, a dynamic recon-

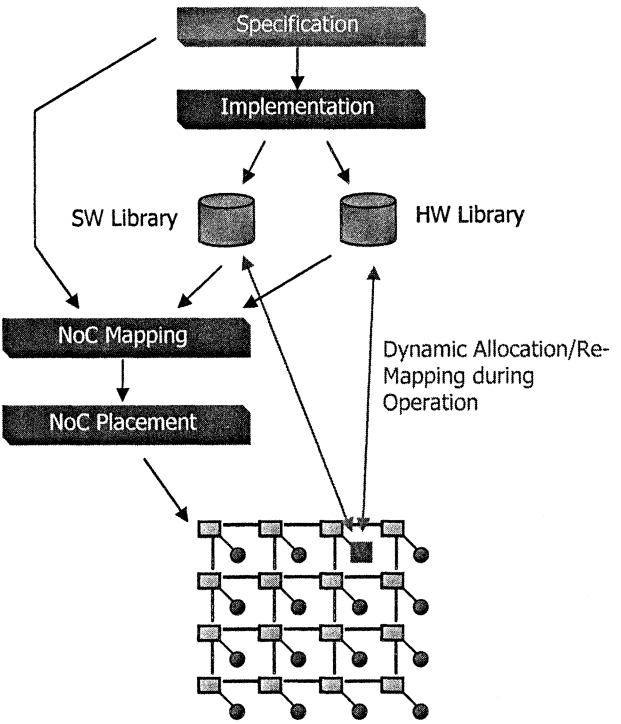

Figure 6. DesignFlow: Static and Dynamic Hardware/Software Co-Design

figuration of the system can be done based on the flexibility provided by the use of processor (software) and reconfigurable hardware IP components. Possible reasons for the need of system reconfiguration during operation can be:

- availability of new implementations optimized to power and performance
- reaction on changed constraints in the system's environment
- improved power efficiency due to low battery status in mobile devices
- additional services to be executed on the system, implementation requires a dynamic re-assignment of resources

The dynamic hardware/software reconfiguration of a hyper-platform is defined as "Dynamic Hardware/Software Co-Design". Two modes of dynamic reconfiguration can be distinguished:

1 minor reconfiguration for inclusion of additional services. An on-chip task scheduler will assign the new tasks to available resources or organize a minor rearrangement of task assignments.

2 major reconfiguration of the whole system: has to be done by a CAD environment from outside

For the dynamic system reconfiguration an implementation database (HW, SW, FPGA) for concrete application tasks is required. This database, which will typically not be complete, provides information about possible implementations of system tasks on IP blocks and the related design properties (e.g. throughput, resolution, power consumption).

7. Conclusions and Future Work

In this contribution we have presented a consequent hierarchical and generic topology for future GALS system-on-chip architectures. The generic approach can be used to customize SoC platforms for dedicated application classes based on simulation profiling results. Furthermore we have presented an hierarchical approach for an embedding of on-chip debugging and testing functionality in GALS topologies. The existing SystemC model/simulator is currently enhanced to a mixed-level simulation environment (SystemC/VHDL co-simulation)for irregular mesh configurations. An automatic generation of application domain specific platforms is focused as next step. The build-in debugging and test topology will be connected to existing probing techniques for power monitoring. Furthermore the new testability concept will combined with classical hierarchical test approaches for circuit components.

References

[Benini and Micheli, 2002] Benini, L. and Micheli, G. D. (2002). Networks on Chips: A New SoC Paradigm. *IEEE Computer*, Vol. 35:70–78.

[Dally and Towles, 2001] Dally, W. J. and Towles, B. (2001). Route Packets, Not Wires: On-Chip Interconnection Networks. In *Proc. of the Design Automation Conference (DAC 2001)*, pages 684–689.

[de la Torre et al., 2002] de la Torre, E., Garcia, M., Riesgo, T., Torroja, Y., and Uceda, J. (2002). Nonintrusive debugging using the JTAG interface of FPGA-based prototypes. In *Proceedings of the 2002 IEEE International Symposium on Industrial Electronics (ISIE 2002)*, volume Vol. 2, pages 666–671.

[Guerrier and Greiner, 2000] Guerrier, P. and Greiner, A. (2000). A Generic Architecture for On-Chip Packet-Switched Interconnections. In *Proc. of Automation and Test in Europe Conference and Exhibition 2000*, pages 250–256.

[Hollstein et al., 2003] Hollstein, T., Ludewig, R., Mager, C., Zipf, P., and Glesner, M. (2003). A hierarchical generic approach for on-chip communication, testing and debugging of SoCs. In *Proc. of the VLSI-SoC 2003*, pages 44–49.

[Jager, 2002] Jager, S. (2002). "System on Chip"-Design: Technische und Ökonomische Herausforderungen. Studienarbeit, Darmstadt University of Technology.

[Jervan et al., 2002] Jervan, G., Peng, Z., Ubar, R., and Kruus, H. (2002). A Hybrid BIST Architecture and its Opimization for SoC Testing. In *Proc. of the Int. Symposium on Quality Electronic Devices*. IEEE.

[Jung et al., 2002] Jung, D.-Y., Kwak, S.-H., and Lee, M.-K. (2002). Reusable embedded debugger for 32 bit RISC processor using the JTAG boundary scan architecture. In *Proc. of IEEE Asia-Pacific Conference on ASIC 2002*, pages 209–212.

[Kumar et al., 2002] Kumar, S., Jantsch, A., Soininen, J.-P., Forsell, M., Millberg, M., Öberg, J., Tiensyrja, K., and Hemani, A. (2002). A Network on Chip Architecture and Design Methodology. In *Proc. of VLSI Annual Symposium (ISVLSI 2002)*, pages 105–112.

[Leiserson, 1985] Leiserson, C. (1985). Fat Trees: Universal Networks for Hardware-Efficient Supercomputing. *IEEE Transaction on Computers*, Vol. C-34(No. 10):892–901.

[Ludewig et al., 2002] Ludewig, R., Garcia, A., Murgan, T., and Glesner, M. (2002). Power Estimation based on Transition Activity Analysis with Architecture Precise Rapid Prototyping System. In *Proc. of the 13th IEEE Int. Workshop on Rapid System Prototyping (RSP)*, pages 138–143.

[Murgan et al., 2003] Murgan, T., Petrov, M., Ortiz, A. G., Ludewig, R., Zipf, P., Hollstein, T., Glesner, M., Ölkrug, B., and Brakensiek, J. (2003). Evaluation and Run-Time Optimization of On-Chip Communication Structures in Reconfigurable Architectures. In *Proc. of 13th International Conference on Field Programmable Logic and Applications (FPL2003)*, pages 1111–1114.

[Muttersbach et al., 2000] Muttersbach, J., Villinger, T., and Fichtner, W. (2000). Practical Design of Globally-Asynchronous Locally-Synchronous Systems. In *Proc. of the Sixth Int. Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC 2000)*, pages 52–59.

[Natvig, 1997] Natvig, L. (1997). High-level Architectural Simulation of the Torus Routing Chip. In *Proc. of International Verilog HDL Conference, California.*

[Nexus 5001 Forum, 1999] Nexus 5001 Forum (1999). IEEE-ISTO Nexus 5001 Forum Standard for a Global Embedded Processor Debug Interface. http://www.nexus5001.org.

[Saastamoimen et al., 2002] Saastamoimen, I., SigÃijenza-Tortosa, D., and Nurmi, J. (2002). Interconnect IP Node for Future System-on-Chip Designs. In *Proc. of the First IEEE Int. Workshop on Electronic Design, Test and Applications (DELTA 02).*

[Sgroi et al., 2001] Sgroi, M., Sheets, M., Mihal, A., Keutzer, K., Malik, S., Rabaey, J., and Sangiovanni-Vincentelli, A. (2001). Addressing the System-on-a-Chip Interconnect Woes Through Communication-Based Design. In *Proc. of the Design Automation Conference (DAC 2001)*, pages 667–672.

[SIA, 2001] SIA (2001). SIA roadmap 2001. http://public.itrs.net.

[Synplicity, 2003] Synplicity (2003). Synplicity. http://www.synplicity.com.

[Tanenbaum, 1996] Tanenbaum, A. (1996). *Computer Networks.* Prentice Hall.

[Taylor et al., 2002] Taylor, M. B., Tim, J., Miller, J., and et. al., D. W. (2002). A Computational Fabric for Software Circuits and General-Purpose Programs. In *IEEE Micro.*

[Wielage and Goossens, 2002] Wielage, P. and Goossens, K. (2002). Networks on Silicon: Blessing or Nightmare? In *Euromicro Symposium On Digital System Deisign (DSD 2002)*, pages 423–425.

[Wiklung and Liu, 2003] Wiklung, D. and Liu, D. (2003). SOCBUS: Switched Network on Chip for Hard Real Time Embedded Systems . In *Proc. of the Int. Parallel and Distributed Processing Symposium.*

[Zimmer, 2002] Zimmer, H. (2002). Fault Modelling and Error-Control Coding in a Network-on-Chip. Studienarbeit, Darmstadt University of Technology.