Modeling and Verification of Globally Asynchronous and Locally Synchronous Ring Architectures

Sohini Dasgupta, Alex Yakovlev

Modeling and Verification of Globally Asynchronous and Locally Synchronous Ring Architectures

Sohini Dasgupta, Alex Yakovlev
School of EECE, University of Newcastle upon Tyne, NE1 7RU, UK
E-mail: {Sohini.Dasgupta,Alex.Yakovlev}@ncl.ac.uk

Abstract

The goal of this paper is to demonstrate a prevalent global deadlock situation resulting from a local deadlock in a GALS ring architecture. We present a novel design for building systems which will be tolerant to such deadlocks arising in the local modules. This paper, concentrates on the modeling of the proposed design methodology and its correctness is proved with the help of a public domain verification tool.

1. Introduction

Future VLSI systems will be based on system-on-chip concepts, involving multiple clocked domains, e.g. synchronous processor cores. The on-chip modules will be glued together by interface logic that must facilitate high speed communication between synchronous modules operating at different clock frequencies. Such heterogeneous systems have a high requirement of a reliable high speed communication scheme, and mitigation of synchronization failure. Designing systems with such characteristics is a difficult task. Globally Asynchronous and Locally Synchronous (GALS) architectures takes advantage of industry standard synchronous methodology with individual clock domains and of self timed interface to cross clock boundaries. An important issue related to designing GALS systems is the dearth of proven design methodologies for the asynchronous communication network that encapsulates each synchronous island to aid data transfer.

The design process must be based on a system model which would aid the traversal from specification to implementation. Petri net modeling allows sound representation of the functionality of the system. There exists various model checking tools to verify the correctness with respect to certain desired properties, refining the system, at every step, leading to efficient synthesis of the system. Correctness of communication protocols play a key role for GALS architectures. The traditional synchronous validation schemes, e.g. simulation and testing, are not compatible or viable to verify the correctness. Formal verification is becoming the most practical way to ensure the correctness of designs.

Point-to-point GALS architectures have been a major impediment in providing efficient throughput and greater freedom in composing existing Intellectual Property(IP) cores of different speed and different types of interfaces. The use of a simple topologies like fork, join, bus and ring, is a possible solution to the above problems. The design architecture proposed by Villiger et. al. [1], is a step towards the right direction but has some inherent limitations. The ring transceiver in [1], consisted mainly of two parts, a router that decides where the incoming packets have to go and an arbiter that either allows an incoming packet from the preceding ring transceiver, or allows the output port of the host circuitry to feed a packet into the ring. The ring design methodology encounters a deadlock in the entire system, if an input port of a participating synchronous module is by any way blocked or defective. The occurrence of such a phenomenon will block the whole ring, as the pipeline stages get filled up with data packets from the dead node backwards to all the senders. This drawback is deterrent to the performance and reliability of GALS architectures. This paper extends GALS multi-point scheme presented in [1] to handle the limitations posed by the previous designs by introducing reliable intermodule communication through ring architectures and achieve maximal throughput, while preserving self-timed operation.

2. GALS Ring Architecture

2.1. The Overall Design

A ring architecture design which handles the problem stated in the previous section is depicted in Fig.1(a). The ring transceiver mainly consists of a ring access controller, which decides whether to allow an incoming packet from the previous transceiver or a new data packet pushed into the channel by the output port of the host synchronous module, shown in Fig.1(b).

2.2. The Ring Transceiver

As opposed to the ring transceiver in [1], the proposed design of the ring transceiver does not have a router, which decodes where the incoming packet needs to go. This is solely because, the packet due to a particular module has been put onto the channel, dedicated to that module, right from the start of its journey. Therefore there are as many channels as there are synchronous modules. The ring access controller receives both the incoming data from previous transceivers and from its host circuitry, as shown in Fig.1(b). At this point it decides whether to pass or insert data into the channel with the help of an arbiter. The input

Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE'05)
1530-1591/05 $ 20.00 IEEE

port and the output port of the synchronous modules can carry out processes concurrently as they are independent. Hence the module can send and receive data at the same time. In our design, the output port sends data packets to the input port of its host, in addition to sending it to other participating modules.

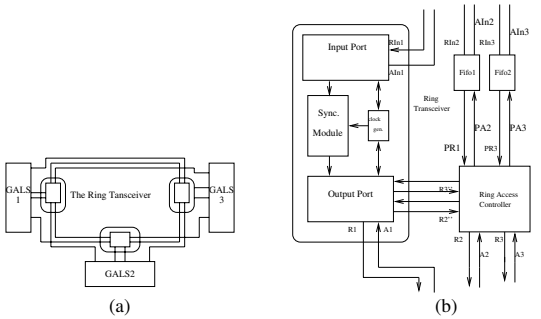

Figure 1. (a)Ring Architecture(b)Transceiver

2.3. Ring Access Controller

Fig.3 shows the Petri net (PN) representation of the ring access controller of a system consisting of three participating synchronous modules. The requests arriving from the previous transceiver, Req2 and Req3, exclusively try to win the arbiter grant on arb2. On reception of the grant signal on either PR2 or PR3, the request proceeds to latch the data to be sent to the appropriate module.

2.4. The Input Port

The PN of the channel leading to input port is depicted in the Fig.2. There exist, dedicated channels, to each input port. The shown in the figure, the request R1 due to arrive at the input port of Module1 has no connection with the other two channels dedicated to modules 2 and 3. Hence, requests R2 and R3 continue to traverse the ring, unaffected by the failure of the input port occurrences of Module1. Hence with the above method, global deadlock arising from local deadlock can be prevented.

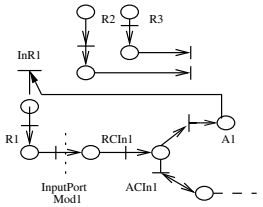

Figure 2. Input channel for Mod1

2.5. The Output Port

The output port generates data packets and inserts them in the channel to be transmitted to the appropriate synchronous module. Before feeding packets into the channel, the output port decides, where it is due and inserts it accordingly in the respective channels. Therefore, an additional circuitry is required to decode the addresses (not shown in the figure). The pipeline stages, on each channel, hold data sent from any module, only to liberate it after the reception of the acknowledgement signal from the receiving module. The output port either sends to or stores data on appropriate channels depending on the arbitration circuit (as shown in Fig.3). The output port can send data packets to the input port of its host on channel InR1. This channel does not proceed to contest for an arbiter grant on arb1, since it creates its own channel and stores its data in the FIFO, and proceeds towards the input port of Module1 (shown in Fig.2).

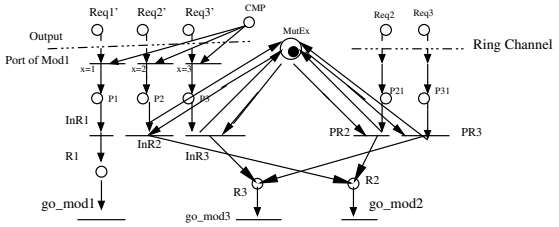

Figure 3. PN for RAC Control Unit

3. Verification of GALS Ring Architecture

Model checking is an approach to formal verification used to determine whether a model satisfies the system specification. The model checking can be done by using two parameters, the system model and the property that the system must satisfy. In our design, for the verification we consider 1-safe nets, i.e. a place $p \in P$ may hold a maximum of one token for a certain marking. This is done to reduce the complexity of verification. The tool used to verify the design is CLP[2]. The choice of the verification tool is based on its expressiveness and analysis power.

There are several types of analysis that can be performed using CLP: mainly, reachability and functional analysis. Our verification procedure focuses on reachability analysis and deadlock check.

4. Results

This section presents the verification results of the design of the Ring Transceiver in [1], Des1, and the new design, Des2, proposed in this paper. The results obtained when the input port of one of the participating module is faulty, in both designs, is shown in the Table below. As a first step of verification, the system models are unfolded using PUNF[2]. The output of PUNF is provided as input to CLP. The analysis proved that the new design was deadlock free, while the previous design experienced global deadlock. We are currently involved in the circuit synthesis of the ring transceiver.

Mod	|s|	|t|	|B|	|E|	DSt	DTr	Live
Des1	60	39	1526	799	√	√	x
Des2	50	35	2034	947	x	x	√

|s|=No. of States, |t|=No. of Transitions
|B|= No. of conditions, |E|=No. of Events
DSt=Dead States, DTr= Dead Transitions

References

[1] Villiger, T., H, Gurkaynak, F.K.: Self timed ring for globally asynchronous and locally synchronous systems. In Proc. of the ninth International Symposium on Asynchronous Circuits and Systems (2003).

[2] Khomenko, V.: Model checking based on prefixes of petri net unfoldings, PhD thesis, University of Newcastle, (2003).