A Formal View of Multicomputers

José A. Gallud†, José M. García‡and Francisco J. Quiles†

†Departamento de Informática, Universidad de Castilla-La Mancha,
Escuela Universitaria Politécnica de Albacete,
Campus Universitario s/n, 02071 Albacete, SPAIN.
email:{jgallud,paco}@info-ab.uclm.es
‡Departamento de Informática y Sistemas, Facultad de Informática,
Campus de Espinardo, Universidad de Murcia, 03080 Murcia, SPAIN.
email:jmgarcia@dif.um.es

Abstract

Formal Methods and Techniques are widely used in communication systems and distributed computing. Today, such techniques are not sufficiently employed in other systems as industrial processes or hardware design. In this paper we propose a methodology to apply the Lotos technique in computer design. Thus, we describe a particular case based on specifying a classical router algorithm in massively parallel machines. The e-cube routing is a deadlock-free algorithm which serves as a valid reference for our purpose. Lotos tools are used to validate the system, in particular we have used LOLA for analysis and design testing.

1: Specifying Computer Behaviour Using Lotos

A number of advantages have been attributed to the use of formal specifications. These advantages cover from enhanced insight into and understanding of specifications, up to help in verification of the specifications and their implementations. Formal methods are intended to be used by software and hardware designers to describe systems, and in particular, to verify that the system has the desired properties [19].

As we can see in [5], the design of special systems as life-critical systems with fault-avoidance is only intellectually defendable by using formal methods. Formal methods are of particular importance in the development of real-time systems [16]. Many other systems [13] can be designed using formal techniques in order to reduce both high cost and time related to the development of complex systems. Finally, formal methods are specially recommended for security and safety systems.

Lotos [3] is playing an important role in exploring the behaviour of diferent kinds of systems. Communication protocols definition and distributed computing are two of the main areas within which formal specification is widely used. Lotos, Estelle and SDL have been chosen to describe important architectural and functional design features related to software engineering [20]. Lotos Formal Description Technique allows software designers to give a precise relation between initial and final states of any application, without describing implementation details.

Formal specification techniques are generally beneficial [12] [4] because a formal language makes specifications more concise and explicit than informal ones [21]. These techniques help us to acquire greater insights into the system design, dispel ambiguities, maintain abstraction levels, and determine both our approach to the problem as well as its implementation.

In this paper we intend to study a special application of the Lotos formal technique to multicomputer design. In a former work [9] we showed the basis of a methodology for specifying diferent problems related to multicomputer design. Multicomputers are a known class of massively parallel machines that have thousand of processors interconnected through a communication network [1]. The nodes do not physically share memory, each one using a local memory. This hardware feature forces nodes to communicate by message-passing technique through the network [17].

The basic function of an interconnection network is to transfer information among communicating nodes of a multicomputer in a efficient manner. In a broad sense, routing refers to the communication methods and algorithms used to implement this function: how to set up a communication path in the network, how to choose a path from many alternatives, and how to handle contention for resources in the network [18].

An important aspect in the design of a routing algorithm is the possibility of deadlocks. Deadlocks occur as a result of the simultaneous occurrence of four conditions:*Cyclic wait*,*Mutual exclusion*, *No preemption* and *Hold and wait*. The resources in routing are the channels and buffers, so deadlock can result if a communication request is allowed to hold the resources allocated to it while waiting for additional resources to become available. A logical channel cannot be shared among nodes.

Deadlocks can appear both in circuit-switched and packet-switched networks. Similarly, both store-and-forward and wormhole approaches are susceptible to produce deadlocks. Various approaches can be used to deal with deadlocks: deadlock avoidance, deadlock detection and deadlock prevention. A detection approach can be used to break a deadlock once it has occurred, however this option is not popular because the channel selection must be repeated, resulting in large and unpredictable latencies.

The unique approach in interconnection networks is to prevent deadlocks by careful design of the routing algorithm. Deadlock avoidance studies the new resource allocation graph for each request in order to detect deadlocks. If a deadlock appear,then the request is rejected. As can be seen, this method introduces large and prohibitive latencies. Deadlock prevention is focused on the four conditions we have showed above. The solution is to introduce constraints in order to eliminate a condition. An easy approach is impose an order in which resources are allocated to a communication request, avoiding the cyclic-wait condition.

Both circuit-switched and packet-switched networks can prevent deadlocks by using one of these methods [18]: The virtual-channel technique (wormhole routing), that is, the capacity of a physical channel is multiplexed among multiple virtual channels. By partitioning the available buffers in a node into multiple classes and controlling the allocation of buffer classes to packets (store-and-forward). By reducing the path selection process in order to avoid cyclic-wait situations, an example for adaptative routing can be found in [8]. The existence of deadlocks in routing algorithms can be studied by constructing the resource allocation graph (channel or buffer dependence graph). A deadlock can appear in the system if the resource allocation graph has a cycle [11] and conversely, the allocation occurs if the graph has no cycles. Most of deadlock-free algorithms have been obtained by analyzing the buffer dependence graph.

In this paper we present an alternative to graphs for deadlock prevention and detection based on the use of the Lotos formal technique. The tools and methods applied to communication networks have successfully used in interconnection networks, and the resource allocation graph is no exception indeed formal techniques should be no exception either. Such a discussion is important in the design of optimal routing algorithms. Formal methods can help designers to find deadlock-free algorithms and, by extension, to develop multicomputers with a rigorous method, with all the advantages of formal methods [7]. We show this by describing the specification of a deadlock-free algorithm, namely, the row-column deterministic algorithm.

The rest of the paper is as follow. In section two we show the case study of a routing algorithm. This routing algorithm is specified using Lotos in section three and, finally, in section four we describe our conclusions and future work.

2: A Case Study:Routing Algorithms

In a multicomputer, the way the nodes are connected to one another varies from one machine to another. In a direct network, each node has a point-to-point, or direct, connection to some number of other nodes, called neighboring nodes. Such an aspect defines a main feature of multicomputers, the topology. Interconnection networks usually have regular and fixed topologies: hypercubes, torus and meshes are the most known topologies. We have choosen a popular topology from the k-ary n-cube family: the 3-ary 2-cube or 2D torus.

In a multicomputer, all the interprocessor communication functions are usually handled by a router. The router interconnects a number of external channels to other nodes in the network, and one or more internal channels to the local processor. Our 2D torus has nine nodes and each node has four external channels and two internal channels. When a packet arrives at the router from an external channel it may either be destined to the local node or may be forwarded on by another external channel.

A valid taxonomy of communication methods used in static networks can be found in [15]. All methods assume a form of distributed control since centralized control needs a power node to make the routing decision and thus is not a practical scheme. Such a taxonomy distinguishes between three classes:

1. *Path setup:* Defines how -statically or dynamically- the path between two nodes is set up. The dynamic approach is usually divided into circuit-switching and packet switching approaches.
2. *Routing path selection:* Defines the problem of choosing a path from among the many alternatives.Basically there are two approaches: deterministic and adaptative.
3. *Network control flow:* Defines the techniques used to regulate the movement of packets from node to node and the efficient use of the network resources. The best known techniques are the following: Store-and-forward, Virtual cut-through and Wormhole.

This classification can help us to characterize a particular router algorithm, and then we can carry out the study of its properties. Thus, the specification of the routing algorithm defined in this paper has the next features: the path is set up dynamically, the routing path selection is deterministic and, finally, the technique used to network control flow is wormhole. In wormhole routing, a message is divided into a sequence of fixed-size units, called *flits*. We have choosen a popular deterministic algorithm: the e-cube or XY algorithm. Such an algorithm offers a main feature, namely, it is deadlock-free.

We want to study the deadlock problem, and how Lotos can solve it. So, in this paper we present both a torus topology and a router algorithm specification in order to analyze deadlock states. We assume all nodes always select a valid destination, and especially a node in the border.

As we have noted above the channel dependence graph models deadlocks in wormhole but it is sometimes tedious to construct the graph and check for cycles [18]. A classic example of proving deadlock freedom of some algorithms can be found in [6].

3: The e-cube Algorithm Specification

Lotos specifications are networks of processes that are activated concurrently and communicate through shared gates or channels. A Lotos specification has two different parts: the abstract data type (ADT) definition and the system behaviour definition by specifying the temporal relationship between every synchronization on the gates. We can indicate the types used within the specification: boolean, natural numbers from 1 to 9, and two special types called Message and Chan. ADT's are not important in this paper.

The specification can be studied by distinguishing between two parts: the specification of the torus topology and the specification of the row-column router.

The specification of the topology is defined after the Lotos word *behaviour*. We have defined a Lotos expression with nine Lotos processes which represent each nine nodes of the multicomputer. This Lotos expression models the torus topology and the expression defines channel joinning two processes.

On the other hand we have the expressions related to the second part, the e-cube algorithm specification. Such a definition is provided by three Lotos processes called *enc*, *comp* and *cross*. Each node of a 3-ary 2-cube has a number of modules: the router, a local memory and a processor are the most important. The router has to perform two main tasks: routing an incoming packet and crossing flits from ingoing to outgoing channels. Once a header flit of an incoming packet is processed, the outgoing channel must be selected. This task is defined with the Lotos processes *enc* and *comp*. Once an outgoing channel has been selected (if the channel is free), the router has to cross the incoming flits up to the outgoing channel. It can be seen that while the router is crossing a message, a new message can arrive via a free channel. In this case such a process should be repeated to manage the new incoming message.

With available Lotos tools, we can analyze and test the desired behaviour. A basic method has been used to test the specification. Such a method includes a number of possible communication paths for a particular node. Among other we can study the following communication directions:

1. Forward a node in the same row
2. Backward a node in the same row
3. Forward a node in the same column
4. Backward a node in the same column
5. Simultaneous combinations of these four cases

Lotos analysis can be done using step-by-step execution with tools such Lite or LOLA [14]. This operation helps us to detect particular problems, but the global space is too large and it is impossible to cover all of them. Testing allow us to reduce this space state by composing the specification with a test case. The next specification is a simple test case to check the simultaneous communication among four nodes: from node1 to node3 and from node4 to node5. The LOLA test operation answers with MUST PASS, MAY PASS or REJECT when it composes a specification with an acceptance test.

```
process two1[cia,cid,ci2,c1,c2,c6,success]:NOEXIT:=
cia!flit3;c1!libre;c1!flit3;cid!flit5;c2!libre;c2!flit3;c6!libre;ci2!libre;
ci2!flit3;
c6!flit5;cia!flit;c1!flit;ci2!libre;c2!flit;ci2!flit5;ci2!flit;
cid!flit;c6!flit;ci2!flit;
cia!end;c1!end;c2!end;ci2!end;cid!end;c6!end;ci2!end;
success;
stop
endproc
```

The result shows that there is no rejection:

```
    Analysed states       = 27
    Generated transitions = 27
    Duplicated states     = 0
    Deadlocks             = 0

    Process Test = two1
    Test result  = MUST PASS.

                   successes = 1
                       stops = 0
                       exits = 0
               cuts by depth = 0
```

All nine nodes in the torus work in the same way, the behaviour is defined by the same Lotos process *comp*. This aspect allows us to execute a limited number of test cases to check the desired behaviour and to detect routing limitations. In this case we have deadlocks because the specification does not define virtual channels. As we can see above the e-cube algorithm is deadlock-free if the network has two virtual channels. This particular feature of our specification can be detected as a result of a test rejection on LOLA.

4: Conclusions

This paper has analyzed how to apply formal techniques to computer design, specially to multicomputer design. In such systems the communication network plays an important role in determing the overall performance of the system. Formal techniques can help us to study a number of problems related to communication network design. The application of formal methods to hardware design has been increasing in recent years, some examples of which are given in [2] and [10]. Our study of the network has focused on the much discussed problem of designing a router algorithm.

An important issue in the design of a routing algorithm is the possibility of deadlocks. Deadlock appears when four conditions are true in the system simultaneously: mutual exclusion, no preemption, cyclic wait and hold and wait. The way to treat deadlocks is prevent them by careful design of the routing algorithm.

Many earlier works have studied such a problem by using the channel dependence graph in order to obtain a deadlock-free algorithm in different ways. Our proposal is introduce the use of the Lotos technique as a valid and easier method to design deadlock-free algorithm. Thus, we have showed the specification of a deterministic e-cube algorithm in a 3-ary 2-cube topology. Lotos tools have been used to validate the system by using the test operation.

In future works we will be studying both other router algorithms and other topology definitions. In this way, we are looking for new Lotos expressions to define greater topologies, since our Lotos expression is difficult to scale. All these new expressions can be analyzed and verified with different tools.

References

[1] W.C. Athas and C.L. Seitz. Multicomputers: Message-passing concurrent computers. *IEEE Computer*, 21(8):9–24, August 1988.

[2] G. Barret. Model checking in practice: The t9000 virtual channel processor. *IEEE Transactions on Software Engineering*, 21(2):69–78, February 1995.

[3] T. Bolognesi and E. Brinskma. Introduction to the iso specification language lotos. In *The Formal Description Technique Lotos*, pages 23–73. Elsevier Science Publishers B.V., 1989.

[4] J.P. Bowen and M.G. Hinchey. Seven more myths of formal methods. *IEEE Software*, March 1995.

[5] R.W. Butter. Formal methods for life-critical software. In *AIAA Computing in Aerospace 9 Conference*, pages 319–329, 1993.

[6] W.J. Dally and C.L. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Transactions on Computers*, c-36(5):547–554, 1986.

[7] J. Davies and M. Wallis. On the formal specification and verification of network algorithms. In *Proc. of VII Formal description techniques*, pages 81–97, 1994.

[8] J. Duato. Deadlock-free adaptative routing algorithm for multicomputers: Evaluation of a new algorithm. In *Proc of Third IEEE Symposium in Parallel and Distributed Processing*, pages 840–848, 1991.

[9] J.A. Gallud. Formal specification of multicomputers. *Lecture Notes in Computer Sciences*, pages 380–390, February 1996.

[10] G. Gopalakrishnan and R. Fujimoto. Design and verification of the rollback chip using hop: A case study of formal methods applied to hardware design. *ACM Transactions on Computer Systems*, 11(2):109–145, 1993.

[11] K.D. Gunther. Prevention of deadlocks in packet-switched data transport systems. *IEEE Transactions on Communications*, 29(4):512–524, 1981.

[12] A. Hall. Seven myths of formal methods. *IEEE Computer*, pages 11–20, September 1990.

[13] M.G. Hinchey and J.P. Bowen. *Applications of Formal Methods.* Prentice Hall International Series in Computer Sciences, 1995.

[14] S. Pavon J. Quemada and A. Fernandez. Transforming lotos specifications with lola: The parametrized expansion. In *Formal Description Techniques I*, pages 45–55. IFIP/North-Holland, 1988.

[15] H.T. Kung. Network-based multicomputers: Redefining high performance computing in the 1990's. In *Proc. of the Decennial Caltech Conference on VLSI*, pages 30–41, 1989.

[16] R. Kurki-Suonio. Stepwise design of real-time systems. *IEEE Transactions on Software Engineering*, 19(1), January 1993.

[17] L.M. Ni and P.K. Mckinley. A survey of wormhole routing techniques in direct networks. *IEEE Computer*, pages 62–76, February 1993.

[18] H.J. Siegel. *Interconnection Networks for Large-Scale Parallel Processing.* McGraw Hill, 1990.

[19] O. Tanir and V.K. Agarwal. A specification-driven architectural design environment. *IEEE Computer*, pages 26–35, June 1995.

[20] J. van de Lagemaat. *Lotosphere, An Overview on Research and Achievements '91.* Ellis Horwood, 1991.

[21] J.M. Wing. A specifier's introduction to formal methods. *IEEE Computer*, pages 8–24, September 1990.