A Case Study in Multilevel Architecture Modelling using HDLs

Pauline C. Haddow, Lasse Natvig*
Department of Computer and Information Science
The Norwegian University of Science and Technology
7034 Trondheim, Norway
pauline@idi.ntnu.no, lasse@idi.ntnu.no

Abstract

The art of both software and hardware modelling over the years has led to the development of more and more complex or detailed models. However the question is, is a complex model a better model? In architectural terms, the level of complexity of a model is said to increase as we move from system-level to gate-level. A model may represent a complete system or just part of it, in such a way as to meet the goals of the model.

In this work, we present two modelling approaches using HDLs — the first using the simulation feature of HDL to provide system-level modelling and the second focusing on logic synthesis to provide gate-level modelling. Both approaches model the same architecture, an asynchronous controlled multicomputer router.

1 Introduction

Hardware Description Languages (HDLs) provide us with the freedom to model systems at various levels of detail.

Simulators may be written in HDL to model large complex designs at the system-level. Computer simulation of simplified models and prototypes of these complex systems is an established technique to unveil design errors at an early stage, as well as improving the cooperation with the end user during development. HDLs such as VHDL and Verilog, provide simulation as an integrated part of the language.

HDL tools often provide inbuilt simulators which check the signal paths of an RTL design written in HDL. This HDL-code may also be written as synthesisable code and as such, may be synthesised into a gate-level design by a logic synthesis tool.

In this work we develop two models of a multicomputer router using the features described above for HDLs. The architecture chosen for our study is the Torus Routing Chip (TRC), a multicomputer router, designed by Dally and Seitz [DS86]. A multicomputer router offers a sufficiently complex architecture suitable for both modelling techniques. The asynchronous nature of this architecture adds an interesting element to this work.

The first model can be said to be an algorithmic or behavioural approach at the system level, modelling the functionality of the router chip and the performance of the interconnection network. The second is a structural approach focusing on how the TRC can be implemented as a collection of modular building blocks. It models the same functionality but is closer to the actual hardware.

A brief description of the TRC is given in section 2 and in section 3 the two modelling ap-

[0]Part of this work has been undertaken while the author was on leave at Nordic VLSI ASA, Tiller, Norway

proaches are presented in more detail. Testing is discussed in section 4. In section 5 we compare the advantages and disadvantages of the two modelling approaches. This comparison is expanded in section 6 with respect to performance metrics achievable and modelling goals. In section 7 we summarise our findings in this work.

2 The Torus Routing Chip (TRC)

In a message-passing multicomputer each node contains an autonomous processor and local memory. Interprocessor communication occurs by routing messages explicitly through an interconnection network (network). An interface — a router, is required to control the flow of information between each processor and the network.

Each processor divides messages into units called packets before sending. Transmission of each packet follows a technique termed wormhole routing where the sections of a packet, flits, flow through the network in a pipelined fashion.

The TRC is connected in a two-dimensional unidirectional torus network where each node has 4 neighbouring nodes. A deterministic routing algorithm termed x-y addressing is implemented. As such, a packet is routed first in x and then in the y direction. Many packets may coexist in the network leading to the possibility of contention for outgoing channels. The TRC uses a blocking method to stop further transmission of a packet when contention arises.

To avoid deadlock — a circular waiting condition, the TRC uses the concept of virtual channels (VCs) [DS86, Dal92]. Two virtual channels are demand multiplexed onto each physical channel.

These features are used to develop both models. However, the level of detail required to express these features differs in the respective model.

3 Modelling Approach

There is much disagreement in the HDL community as to the choice between VHDL and Verilog. However, what is important for this work is that HDL languages contain the features required to meet the models' requirements. Therefore we chose to implement each model in the language currently available in the organisations of the two developers - Verilog for the behavioural model (section 3.1) and VHDL for the synthesisable structural model (section 3.2).

3.1 Architecture Simulation of the TRC (VERsim)

In this approach, a Verilog simulation model of the TRC has been implemented. The goal being to create a behavioural simulation enabling system level performance results to be obtained for different traffic patterns.

The system module defines the TRCs, processors and channels (or wires) of the system. The processors contain tasks for sending and receiving of packets. A detailed model of the processor connected to each of the TRCs is not included as the focus of the model is on the TRCs themselves. As such, the processors and the interconnection network provide an environment to test the functionality and performance of the TRCs. To provide testing in a realistic environment, a 16 TRC network is simulated.

The central part of the TRC module is three concurrent processes each listening on one of three input channels x_in, y_in and proc_in. This is modelled using three infinite loops (initial forever). These tasks describe the functionality within the TRC which handles both the receipt of data, routing decisions and forwarding of data either to the local processor or towards the destination i.e. transmission to the relevant neighbouring node.

In addition the model handles the self-timed control signals, blocking and implementation of the virtual channels. Control of the external channels connecting the TRCs is modelled following the two cycle signalling convention based on a pair of Req and Ack lines. When an incoming packet cannot be sent to the required output channel due to another packet using it, the packet is blocked i.e. the task suspends and resumes when access to the output channel is granted. To handle virtual channels, each of the input and output tasks are divided into two parallel tasks.

A more detailed description of this model can be found in [Nat97].

3.1.1 Performance Analysis within VERsim

Throughput is a measure of the traffic handling capabilities of the network. This may be expressed either as the rate at which packets injected into the system reach their destinations or the number of packets reaching their destinations within a fixed time period. In the former case results are normalised to the number of packets per simulation cycle whereas in the latter case results are expressed in terms of the number of packets per time period (number of simulations cycles).VERsim assumes the latter case. Both these expressions assume that a start up period is given to allow the network to stabilise.

Latency, in VERsim, is modelled as the number of simulation cycles taken for a packet to traverse the network. This is dependent on the number of TRC's and channels crossed on route to the destination. Constant delay values chosen for the TRCs and channels reflect the fact that in current technology the delay incurred traversing a router is much greater than that incurred through a connecting channel [Aga91].

Test programs that stimulate the network of TRCs and gather statistics for system performance are included in the processor module as a verilog task. As such, the same test program is driven at each processor in the network. These test programs are the source for traffic originating at a node.

Currently performance results in VERsim are expressed in simulation cycles. As such the results are independent of the timing of the simulator itself. However, when system parameters are available, from VHDLsyn, results will then be expressed in nanoseconds.

3.2 Modelling a TRC using Synthesised HDL Building Blocks (VHDLsyn)

In VHDLsyn the goal was to provide a flexible platform for modelling a router to a level of detail that both the router itself and variations in the design could be modelled and analysed.

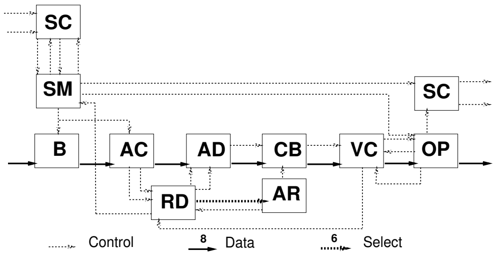

Figure 1: Torus Routing Chip - Data and Signal Path

It was chosen to represent a router as an interconnection of a set of modules. As such it can be said to be a structural approach. A module is not necessarily a single router component but may be made up of a number of components. Each module is characterised with key parameters that effect the design of the module.

A register-transfer level behavioural description of each module has been written in VHDL and synthesised into a gate level design using the Xilinx Foundation Series (Foundation). Simulation of the resulting net-lists has also been undertaken to check for correct behaviour. Layout and timing details were then obtained through the XACT Design Manager within Foundation.

Although the TRC was built using CMOS technology, FPGA technology has been selected as the implementation technology for the model. FPGA technology was chosen to provide a flexible implementation media in keeping with the goals of the model.

A system-level router structural description is written using the Performance Description Language (PDL) developed at the University of Cincinnati [VMM96]. This program describes the interconnection of the modules and includes descriptions of the modules themselves. Each PDL module description provides a frame into which a gate-level design is included. A Perl program converts the XNF net-list format provided by Foundation into the PDL net-list format.

The choice of PDL instead of VHDL at the higher level was again to provide increased flexibility in the model. PDL is designed specifically for generic performance modelling enabling design instances to be compiled and analysed.

The combination of PDL's ability to provide generality in design and VHDL's ability to realise gate-level, technology specific designs allows the model to easily realise design variations. Further information regarding this modelling technique may be found in [Had97].

The router modules required to describe the TRC are as follows: AC, an address comparator; AD, an address decrementer; AR, an arbiter; B, a buffer; CB, a crossbar; RD, a routing decision module; SC, a signal converter; SM, a signal manager and VC, a virtual channel controller. Figure 1 illustrates the structural description of a single path given in the PDL code. A similar description is required for each of the paths through the router i.e. one for each input channel.

The asynchronous nature of the TRC both externally — on the communication channels to neighbouring nodes and the local processor and internally — between and within individual modules added a significant amount of complexity to the model.

3.2.1 Performance Analysis within VHDL-syn

The key performance goals for this model are Cost and Latency.

Since FPGA technology has been chosen, cost is represented by the number of CLBs (combination logic blocks) in the overall design. This figure is just a summation of the number of CLBs estimated for each synthesised module as generated by Foundation. These figures represent synthesised designs optimised for speed and area.

As described in section 2, packet transmission is of a pipelined nature. The first part of the packet — the header, encounters decisions in the router, whereas the rest of the packet — the body, just follows the header through each router on route to the destination. As such, the flits in the body of the packet encounter a smaller delay traversing the router than that of the header. As such, two figures for latency through a router are derived. The longest path delay, which includes the delays through the decision modules represents the delay for each of the header flits. The shortest path delay, ignoring decision modules, represents the delay encountered by the flits of the body.

A simplifying assumption made for the router layout is as follows: if the total number of CLBs required in the TRC design is less than the number of CLBs available on the chosen FPGA chip then each module may be laid out as in the layout provided in Foundation for the individual modules.

This assumption is based on the fact that in newer FPGAs the abundance of routing paths allows designs which use up to perhaps 90% or more of the available CLBs to be routed without the need for extra long wires. This is unlike earlier versions where this analysis may have required restrictions to perhaps only 60% of the CLBs. As such latency at the module level is provided by the longest path delays generated by Foundation. These delays are then fed into the respective module frames provided by the PDL code. The PDL program then calculates the longest and shortest path delays through the TRC design. Network latency is also derived within the PDL program. This represents the delay for a packet crossing a network and depends on the the delay through each TRC, the delay of the connecting channels and the packet itself.

At present throughput is not included in the analysis due to the lack of loaded network features in the module.

4 Testing

4.1 VERsim

The first approach, simulates the behaviour of each individual TRC within the structure of the network. With almost 200 parallel activities in the system we began with simple tests to aid debugging of the model.

During the progress of the project we have moved towards more elaborate tests. An important test is where each processor sends messages to randomly chosen destinations at randomly chosen times. The source and destination address are stored in the message so that the receiver can check that each transmission was correct. Such random tests showed to be crucial as they gave rise to situations which tested the deadlock avoidance logic.

4.2 VHDLsyn

VHDLsyn is a two level hierarchical model and as such testing began with testing at the lowest level of the design — the modules.

For each module, correct synthesis of the RTL description to the gate level is assumed as this is taken care of by Foundation. Efficiency of RTL-code may effect synthesised results. Therefore, although optimisation within synthesis was selected, it is not assumed that the code is optimal. However in this work, the aim is to compare the two approaches and not the efficiency of the code. Behavioural correctness within the synthesised modules is not assumed and each module has been simulated after synthesis to test for correct signal behaviour. This identified many errors within the asynchronous behaviour of the modules.

Behavioural testing of the modules after conversion to PDL-net-list format has also been undertaken. Finally behavioural testing of the router itself was tested as described in section 4.3.

Incoming Channel	**Available Outgoing Channels**
P_OUT	P_IN, Xo_VC1, Yo_VC1
Xi_VC1	P_IN, Xo_VC1, Xo_VC0, Yo_VC1
Xi_VC0	P_IN, Xo_VC0, Yo_VC1
Yi_VC1	P_IN, Yo_VC1, Yo_VC0
Yi_VC0	P_IN, Yo_VC0

Table 1: Test Set 1: Output Choices under Deterministic Routing

Incoming Channels	**Outgoing Channel**
P_OUT, Xi_VC1, Xi_VC0, Yi_VC1, Yi_VC0	P_IN
P_OUT, Xi_VC1, Yi_VC1	P_IN
Xi_VC1, Xi_VC0	Xo_VC0
P_OUT, Xi_VC1, Xi_VC0, Yi_VC1	Yo_VC1
P_OUT, Yi_VC1, Yi_VC0	Yo_VC0

Table 2: Test Set 2 : Arbitration Priority for Competing Input Channels

4.3 Testing for Behavioural Equality

Tests are required to check for correct behaviour, that is that in each model the router responds to incoming messages in the manner described in the specification. By response we refer to the order that incoming messages are handled and the choice of outgoing channel.

As stated earlier, the routing algorithm implemented in the TRC is a form of restricted routing. This means that not all outgoing channels are available for a given incoming packet. We therefore first identified the possible outgoing channels for a given incoming packet, as shown in table 1, and tested to see if based on a given destination address, it was forwarded to the correct output channel. For the purpose of these tests an unloaded network is assumed. A loaded network does not change the correctness of the routing decision for a deterministic routing algorithm.

	VERsim	**VHDLsyn**
Language & size (no of lines)	Verilog (2000)	VHDL/PDL (3000)
Development Time	3 man months	5 man months
Complexity	medium	medium / high
HW Realisability	medium / poor	very good
Scalability	good	poor
Flexibility	poor	good
Execution Time	fast	fast

Table 3: Comparison of Model Characteristics

The second set of tests involved a loaded network with more than one packet competing for the same output channel. Again we identified those incoming channels which may compete for a given output channel and sent requests from each of the required input channels to the given output channel. The arrival sequence in the results highlighted the correctness of the arbitration implementation.

A second arbitration level exists within the router for arbitrating the multiplexor controlling the two virtual channels which share each physical channel. If both VCs have packets waiting, then multiplexing between the packets occurs on a flit basis. That is that each may send one flit before control is passed to the other waiting VC. As this level of detail is not implemented in VERsim, comparison tests have not been undertaken. However, testing in VHDLsyn showed correct behaviour.

5 Discussion

In table 3 which compares the weaknesses and strengths of the two modelling approaches, a strong trade-off between scalability and flexibility is suggested.

With scalability we mean the possibility of expanding the model for analysing larger systems i.e. with an increased number of nodes. Although increasing the number of dimensions has often been regarded as a scalability factor, recent research has moved away from high diameter networks which can be difficult and costly to build [OKS97]. As such, this factor is not included in our analysis.

An important feature of VHDLsyn is it's flexibility. The use of parameterised modules implies the possibility of looping through various design alternatives for exploring (parts of) the space of possible designs in search for an optimum design. In addition, the building block nature of the model allows swapping of blocks (modules) to allow for design variations and the addition of new features.

This increased flexibility, along with the increased complexity of VHDLsyn and limitations in the PDL tools currently available, does not allow for modelling larger systems within a reasonable development time.

Asynchronous control particularly within the router has given rise to increased development and testing time in VHDLsyn. It is believed that a synchronous example would have reduced the development time. In VERsim the choice of synchronous or asynchronous control does not adversely effect the development time.

Neither of the two models showed excessive execution times in the tests undertaken illustrating the suitability of the two HDL languages for both these modelling approaches.

In a high-level simulation too much thought on HW-realisation may make it difficult to master the complexity at the current phase of the design process. On the other hand too little thought may result in a model not realisable in hardware or with only modest performance. The initial aim for VERsim was to test out Verilog as a tool for system-level modelling. That is, to provide a test-bed to try out different network sizes and different traffic patterns. As such, HW realisation was not the primary focus. However, if the focus is on HW realisation it is possible to write a

	VERsim	VHDLsyn
Cost	medium / poor	very good
Throughput	good	currently restricted
Latency (unloaded)	medium	very good
Latency (loaded)	medium	currently restricted

Table 4: Achievable Performance Metrics within the Models

high-level simulator more attuned to this goal.

6 Evaluation

Modelling using HDLs may be classed into two types. The first type is a performance model. With this we mean that the model is written to study the performance of the underlying architecture under various conditions without necessarily a requirement to implement the variations in hardware. The second type is to create a model as a step towards a hardware implementation. Here we assume that the specifications are already made and that the modelling tool aids work towards an implementation that meets these specifications. A relatively new type of modelling which may be considered as a subtype of the second type is the creation of executable specifications. That is that from the earliest stage in a design process, ideas are expressed in HDL, again with a view to eventual realisation of the design.

Our focus in this work has been on performance modelling with the additional aim that VHDLsyn be as close to a hardware realisation such that realistic estimates of cost may be obtained. This characteristic also enables the model to be used as a design platform for testing out design variations for actual implementations.

Goals within performance modelling may be categorised by the type of performance results being sought. In table 4 we compare the two models for their abilities to handle different performance metrics.

If cost is an issue then a more complex model is required so that more realistic estimates of cost may be obtained. This need not necessarily be in the form of VHDLsyn but may also be in the form of a parameterised cost model as in [Chi93].

To measure throughput it is important that the model offers the ability to accept a variety of data inputs so that both the effect of different loads on the network and different traffic patterns may be tested. VERsim offers these possibilities. VHDLsyn, on the other hand, is limited due to the data interface and statistical gathering limitations currently in PDL. However this problem is currently under investigation.

Important features in latency analysis include the size of the network, the packet length and the traffic to be tested. Much research has been conducted in this area and VERsim may be said to represent a typical simulator model for this purpose with the exception that an HDL language is used in place of a high-level language.

The purpose of creating a more detailed model with respect to latency is to improve the accuracy of results sought. As stated in [Chi93] the complexity of a router is an important factor not generally taken into consideration for latency calculations. Also it is important to distinguish between the delay encountered by header flits and that encountered by body flits [BP93]. A more detailed analysis can meet these goals.

7 Conclusion

Two performance models describing the Torus Routing Chip (TRC) have been developed. The advantages and disadvantages of these two models are discussed and a number of modelling goals for which each of these models are suited are identified.

It is worth mentioning a couple of interesting features highlighted in this work. The main difference between a modern programming language and Verilog is the task concept in Verilog which should not be confused with procedures in programming languages. This created some problems during the writing of VERsim but these problems were eventually solved.

It has been interesting to test out the synthesis and simulation tools of Foundation with an asynchronous design. Synthesis tools support only a subset of the language and their design is based on RTL which is inherently a synchronous design methodology [Rus95]. As such, it does not preclude the logic synthesis of asynchronous designs but makes the task somewhat more complicated.

The increased complexity of working with asynchronous design and the lack of an VHDL-code level simulator within Foundation has made debugging a labour intensive job. However, since this approach first breaks the design into modules for synthesis, both creation and testing of the design is more manageable. Although synthesised versions of the modules are now available, more thorough simulation work is required to ensure correct signal flow both within and between the modules.

The work on the models will continue, and we expect that experience with the integrated use of the two models will result in adjustments (calibrating) at the two levels.

The latency for a message through one TRC is a typical example of a parameter that would benefit VERsim from calibration from VHDL-syn. The effect of blocking within the TRC itself can also be used to calibrate throughput in VERsim. On the other hand realistic traffic modelling in VERsim can aid development of more detailed blocking features in VHDLsyn.

References

[Aga91] A. Agarwal. Limits on Interconnection Network Performance. *IEEE Trans. on Parallel and Distributed Systems*, 2(4), Oct. 1991.

[BP93] S. Balakrishnan and D. K. Panda. Impact of Multiple Consumption Channels on Wormhole Routed k-ary n-cube Networks. In *Proc. of the Intern. Parallel Processing Symposium '93*, pp 163–167, 1993.

[Chi93] A. A. Chien. A Cost and Speed Model for K-Ary N-Cube Wormhole Routers. In *Proc. of Hot Interconnects '93*, Aug. 1993.

[Dal92] W J Dally. Virtual-Channel Flow Control. *IEEE Transactions on Parallel and Distributed Systems*, 3(2) pp 194–205, March 1992.

[DS86] W. J. Dally and C. L. Seitz. The Torus Routing Chip. *Distributed Computing*, 1 pp 187–196, Jan 1986.

[Had97] Pauline C Haddow. A Generalisaton of Router Chip Design. In *5th Euromicro Workshop on Parallel and Distributed Processing*, pp 307–313, Jan. 1997.

[Nat97] L Natvig. High-Level Architectural Simulation of the Torus Routing Chip. In *Proceedings of the 6'th International Verilog HDL Conference (IVC'97), Santa Clara*, pp 48–55, April 1997.

[OKS97] M. Ould-Khaoua and R. Sotudeh. Performance Evaluation of Hypermeshes and Meshes with Wormhole Routing. *Journal of Systems Architecture*, pp 345–353, March 1997.

[Rus95] A. Rushton. *VHDL for Logic Synthesis: An Introductory Guide for Achieving Design Requirements*. McGraw Hill, 1995.

[VMM96] B. Vemuri, R. Mandayam, and V. Meduri. Performance Modelling Using PDL. *IEEE Computer*, pp 44–53, April 1996.