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.

Keywords: Architecture modelling, High level description languages, Multicomputer router, Simulation, Logic syntheses, Asynchronous design

1 Introduction

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

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

Simulators may be written in HDL to model large complex designs at the system-level. HDLs such as VHDL and Verilog, provide simulation as an integrated part of the language. 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. Simulation may also be used for performance analysis with respect to a given application or for system refinement analysis.

HDL tools often provide inbuilt simulators. These may be used to check the signal paths of an RTL design. The user can provide stimulus to the design either by using a test-bench or interactively using the simulator tool and the simulator generates results most often output in the form of graphical waveforms. As such, these tools provide a valuable testing environment for an RTL design.

HDL-code may also be written for logic synthesis. Logic synthesis tools place a restriction on the code to follow the coding style and subset implemented within the logic synthesis tool. Presently, commercial synthesis tools are limited to the RT level although system level tools are being developed. RTL synthesis enables HDL descriptions to be synthesised into gate level designs. As such, this takes a design one step nearer to an actual hardware implementation.

In this work we develop two models of a multicomputer router, the first using the simulation features of HDL and the second using RTL logic synthesis to describe an existing architecture. The architecture chosen for our study is the Torus Routing Chip (TRC), a multicomputer router, designed by Dally and Seitz [5]. 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 especially for the logic synthesis model.

The first model can be said to be an algorithmic or behavioural approach to modelling the functionality of the router chip and the overall performance of the interconnection network at the system level. The second is a structural approach based on modelling the TRC 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 approaches 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 and in section 6 we present a more detailed description of how flexibility is achieved in VHDLsyn. The model comparison is expanded in section 7 with respect to performance metrics achievable and modelling goals and includes a number of experimental results achieved in these models. In section 8 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 where 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) [5, 4]. 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. That is that the chosen language offers the features required to create a system-level simulator and that the language offers logic synthesis at the RTL level. As both languages offer these features either language could have been used for both models. The language choice for each model was therefore left to the discretion of the given implementor. As such the most obvious choice, without any concrete evidence to support one language over the other, was to choose the language and associated tools 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). Therefore, it should be borne in mind that this work is not a trade-off between VHDL and Verilog but describes an application of HDLs in a modelling environment.

3.1 Architecture Simulation of the TRC (VERsim)

In this approach, a Verilog simulation model of the TRC interconnection network has been implemented. The goal being to create a behavioural simulation from which system level performance results may be obtained for different traffic patterns and for different interconnection network sizes.

The top level model, system, 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 interconnection network and the TRC itself. As such, the processors and the interconnection network provide an environment to test the functionality and performance of the TRCs.

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 [7].

3.1.1 Performance Analysis within VERsim

Throughput is a measure of the traffic handling capabilities of the network. There are two main methods for calculating throughput. The first, considers a fixed load — number of packets injected into the network. Throughput is expressed as a normalised rate — number of packets per cycle, based on the ratio of the number of packets injected into the system to the time taken for these packets to reach their destination. The second assumes a fixed time period — number of simulation cycles. A start-up period is given to allow the network to stabilise i.e. to build up traffic within the network. Then a count is taken of the number of packets reaching their destination within the time period. This may again be normalised to the number of packets per cycle. It should be noted that during this time period packets will continue to be injected into the system. VERsim assumes the latter case.

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 TRCs and channels crossed on route to the destination and the length of the packet itself. Two constant values are used to represent the delays through the TRC and channels respectively. These values reflect the fact that in current technology the delay incurred traversing a router is much greater than that incurred through a connecting channel [1].

To provide more realistic results, results from VHDLsyn will be used to calibrate VERsim. Latency, longest path delay, at the router level will be used to calibrate the delay at each router in the network simulation. This will not only improve the interconnection network latency results obtained but also the throughput results. For a given message to reach it's destination, it will encounter a number of router delays dependent on the number of routers traversed on route to it's destination. A more accurate router delay estimate will lead to improved accuracy in the estimate for time taken for each message to reach it's destination. As such, the computed rate at which messages reach their destination will be more realistic.

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

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

The aim of VHDLsyn is not to represent an exact copy of the TRC but a sufficiently accurate copy that cost and performance analysis may be undertaken at a lower, more detailed level than in a more typical parallel computer performance modelling environment such as that implemented in VERsim. It also provides a platform for design space exploration. With these goals in mind, it is believed that assumptions may be made regarding implementation details where the behavioural characteristics and principal design decisions made are not effected. As such, with the limited low level information available for the TRC, it is still believed that a reasonable more detailed performance model is realisable.

Although the original TRC was built using 3μ 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. Therefore, it is not intended to compare the results obtained with the original implementation but to explore design variations and provide VERsim with calibration figures for these different design instances.

It was chosen to represent the router as an interconnection of a set of modules. Each module can be said to be a building block for a router construction. As such, it can be said to be a structural approach. A module is not necessarily a single component but may be made up of a number of components. A module may be refined, replaced or their interconnection may be varied to realise different router instances. It should be noted that in this work, reference to components refers to standard components found in the digital libraries of computer aided design tools.

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; OP, an output controller; RD, a routing decision module; SC, a signal converter; SM, a signal manager and VC, a virtual channel controller.

A register-transfer level behavioural description of each module has been written in VHDL. These descriptions have been restricted to that acceptable for logic synthesis. Using logic synthesis on the module designs brings the model one step nearer to a hardware implementation and therefore more realistic performance data are achievable.

Each module is characterised with key parameters, design parameters, that effect the design or number of units of a given module required — see figure 2. Module refinement occurs through changes to variables or constants within the VHDL code, re-compilation and logic synthesis to produce a new design instance. It is often the case that a design change effects more than one module. As such, it was decided to put the design parameters into a package available from a library included in all the module descriptions. Where possible, all references to design decisions within the modules refer to the constants and variables of the package. As such, for any number of parameter changes which effect the design of a module or modules, the package is first updated and then the module(s) are recompiled and re-synthesised to give a new set of designs for the router design instance. Other parameter changes do not effect the design of a particular module but the number of instances of a given module required in the router.

The tool used for this part of the work is Aldec's Xilinx Foundation Series (Foundation) [11].

A system-level router structural description is written using the Performance Description Language (PDL) developed at the University of Cincinnati [10]. This program describes the interconnection of the modules and includes a more general description of each of the modules. This description includes both design issues effecting the design of the underlying modules and also other issues effecting the design of the router itself. As such, each PDL module description provides a frame into which a gate-level design is included and receives and generates performance data for performance analysis purposes. The existence of performance information parameters within the generic description provides a flexible performance modelling environment.

A Perl program has been created to convert XNF net-list formats into the PDL net-list format. Also creation of a conversion program for EDIF format is currently being undertaken. As these two net-list formats are perhaps the most accepted net-list formats the use of the model is not restricted to the Foundation modelling environment.

To create a TRC router instance and generate router level or interconnection network level performance results these modules must be interconnected as specified in the original TRC design. Figure 1 illustrates the structural description of a single path given in the PDL code. A similar

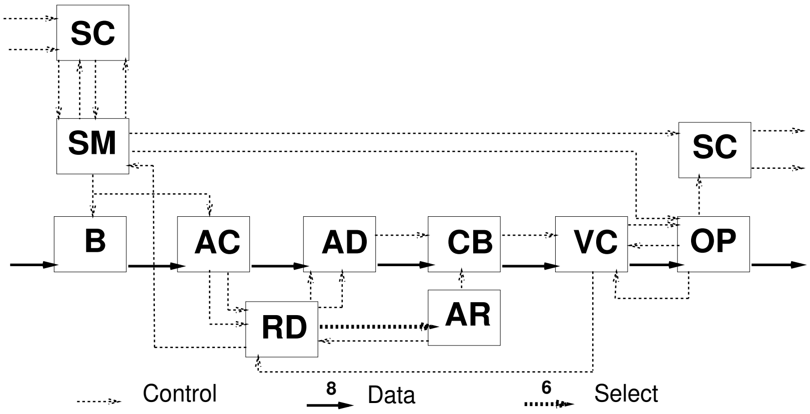

Figure 1: Torus Routing Chip - data and signal path

description is required for each of the paths through the router i.e. one for each input channel. The data path, as shown, flows from the input buffer to the output controller. The modules that do not lie on the data path provide asynchronous control both external and internal and include the decision making elements of the router. It should be noted that in the router all paths flow through the crossbar and it's arbiter and as such only one unit of each of these modules is required.

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.

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 [6].

3.2.1 Performance Analysis within VHDLsyn

The key performance goals for this model are cost and latency.

As stated earlier it was chosen to use FPGA technology as the implementation technology to obtain gate-level performance data. In FPGA technology, routing has a considerable effect on the overall delay. The first question raised by this fact is whether a design can be implemented or not. This question is basically in two parts. Firstly, are there enough CLBs available to meet the design requirements and secondly can the chosen CLBs be routed? The first part is easy to solve in that the number of CLBs in a design must be within the limit available for the device chosen. The second part is more complex. This is dependent on the number of routing channels available between CLBs and the placement of the logic to be routed. In earlier devices, a rough estimate was that if less than 60% of the available CLBs in a device were required for a particular design then the design could be routed. However, in newer devices such as the XC4036 device chosen in this work, the increased number of routing channels increases this estimating factor to perhaps around 90% of the FPGA to be routed.

It is assumed in this work that as long as the total number of CLBs in the router design is within the 90% limit suggested that the entire router can be routed on a single FPGA. For each module, the number of CLBs required for the given design described in VHDL is obtained after logic synthesis. A balanced optimisation for speed and area using level 1 optimisation — maximum optimisation, was selected. As this model offers cost evaluation in addition to latency (delay) evaluation and with this a possibility to trade-off cost against delay, it was chosen not to prioritise speed or area in the model. However, re-synthesis of the given modules may be made if a more optimal solution for either cost or delay is required in a given model instance.

Although these newer devices have greater routing capacity, two questions need be raised regarding delay estimation. As stated earlier, it is only the modules of the module set that are synthesised into gate-level designs and not the complete router design. Therefore, for a given module, if the routing step is undertaken can it be guaranteed that the same placement and routing will be achieved when several modules are placed together on an FPGA? Tests conducted within this work have shown that the percentage of routing with respect to logic for a number of designs may vary considerably. Therefore, how can an estimate for routing between modules be found? The answer to these questions is that comparison of modules after they have been routed has no simple solution. However, a solution that may be used both within and between the modules is to normalise the routing.

A normalised estimate for routing is available in Foundation for any given device by stopping the design process after mapping and before placement and routing. This is based on best case routing and gives a normalised routing factor of 0.984ns for the device chosen. Best case routing assumes that all communication within a design occurs between neighbouring CLBs i.e. there are no long communication wires. Therefore, for a given module the longest path delay is based on the number of CLBs on the path and the number of interconnections (nets) where the number of nets is equal to the number of logic levels minus one. However, the delay figures obtained for each module are based on longest path delay from any input to any output pad and as such include the delays at these pads. These delays are subtracted from the delay figures obtained for each module such that the longest path delay figure refers to the delay results reflect the longest path delay from input to output ports.

It is also assumed that interconnection of modules follows this best case estimate using the normalised net-value for each interconnection. Again the figure used is 0.984ns. It may be surprising that an estimated net be used to this precision when a 1ns estimate could have been used instead. However, as stated the delay figures obtained from Foundation are based on this estimate. To change the normalising factor within the modules from 0.984 to 1.0 would involve finding the difference in the delay of the total nets in a given design based on these two factors and adding this delay to the delay estimate. As such, rounding up the scaling factor just involves more work and less precision and therefore not justifiable. Also, as this factor is used within the modules, it is decided to use the same precision between the modules for consistency.

Design analysis based on a best case value are very optimistic and in practice is not realisable. However, as a normalised value it provides a base for fair comparison between different design instances, a goal of this work.

The delays and cost estimates generated for each module are fed into the respective module frames provided by the PDL code. At the router level, 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 a packet encounter a smaller delay traversing the router than that of the header. As such, two figures for latency through a router are required: the longest path delay — the delays through the decision modules, and the shortest path delay — ignores the decision modules. The former represents the delay seen by a header flit and the latter represents the delay seen by a body flit. These delay figures are calculated within the PDL program along with the network latency — the longest path delay for a packet crossing a network. This later figure depends on the delay through each TRC, the delay of the connecting channels and the packet itself. Again the bets case net value is used for the channel delay.

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. A four by four unidirectional torus was used for testing of VERsim. 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 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 off course effect synthesised results. Recognised coding techniques to avoid extra flip flops and latches have been used to improve coding where possible. However, although optimisation within the synthesis tool was selected, it is not assumed that the the resulting net-list is optimal due to possible inefficiencies in the code. However in this work, the aim is to compare the two approaches and not the efficiency of the code.

Unlike the synthesis process, behavioural correctness within the synthesised modules may not be assumed. Each module has therefore been simulated after synthesis to test for correct signal behaviour. This identified many errors within the asynchronous behaviour of the modules. Each module has also been tested along with those modules that effect it's behaviour to check for correct timing of the signals being sent between modules. It was found that individual modules which were shown to function correctly on their own using their respective test vectors often showed incorrect behaviour when their inputs were driven by the driving module itself. That is that instead of input arrival being specified in test benches,

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

arrival depended on the timing of the driving module(s).

Behavioural testing of the modules after conversion to PDL net-list format has also been undertaken to ensure that no errors were introduced during the process of conversion. Behavioural testing at the router level was tested as described in section 4.3.

4.3 Testing for Behavioural Equality

To test the complete router design, in either implementation, it is important to ensure that the router responds to incoming messages in the manner described in the specification. Response in this situation refers 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. The possible outgoing channels for a given incoming packet were therefore identified, 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

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

Table 2: Test Set 2 : Arbitration priority for competing input channels

is assumed. A loaded network does not change the correctness of the routing decision where a deterministic routing algorithm is implemented in the router.

The second set of tests, shown in table 2, involved a loaded network with more than one packet competing for the same output channel. Again those incoming channels which may compete for a given output channel were identified and requests sent 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 at the multiplexor controller. This module controls the selection between the two virtual channels which share a 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 FIFO buffer. As this level of detail is not implemented in VERsim, comparison tests have not been undertaken. However, testing in VHDLsyn showed

	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

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 [8]. As such, this factor is not included in our analysis.

In VERsim systems of various sizes can easily be constructed by modifying the Verilog module 'system'. This module consists of a collection of

TRC instantiations and their interconnections forming a torus structure of the desired size. This can easily be modified for a given network size by numerous cut and paste operations in any text editor. However, a simple program for generating the network structure should be easy to write and certainly would be a more elegant method.

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 — see section 6.

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.

For a larger system, in system-level simulation too much thought on hardware 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, hardware realisation was not the primary focus. However, if the focus is on hardware realisation it is possible to write a high-level simulator more attuned to this goal.

6 Achieving Flexibility in VHDLsyn

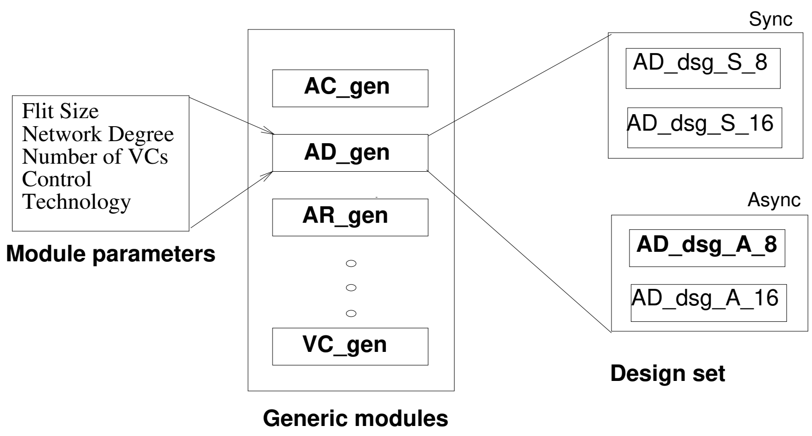

Figure 2: Relationship between parameter choice and module design

As stated in section 5, a key feature of VHDLsyn is it's flexibility. That is both it's ability to change parameters and realise new design instances and the building block nature of the model encouraging design reuse. The latter resembles the virtual component concept where much current research in the hardware design community is focused. The building blocks used in the TRC model are part of a larger module set derived to describe a range of router designs. That is that each module represents a feature present in one or more of the routers. By identifying these features as self contained modules it aids design reuse in the development of newer router designs. The TRC design is put together by selecting a subset of the module set which meets the TRC specification. The design may also be varied by swapping modules from the module set or creating new modules as required.

A set of design parameters has also been developed from a study of key issues in router design. These parameters are used to select a design for each of the modules in the TRC. This process is described in figure 2 for the case of the address decrementer module. The design parameters effecting the address decrementer module are listed on the left hand side — module parameters, and the available designs in the design set to the right. Available designs are those designs which have been implemented in VHDL compiled, synthesised and converted to PDL net-list format. For the original TRC design with a flit size of eight and asynchronous control, AD_dsg_A_8 is selected. However, when exploring the design space it may be the case that the required design is not included in design set. If this is the case then the required design may be created and the design set expanded to include this design.

A 'package' of constants holding a list of the design parameters is held in a VHDL library created for this purpose. The VHDL code written for each of the modules refers to the items in this library. This means that a design issue effecting a given model is automatically updated after the package is updated and the module recompiled. Synthesis of the module will then result in a new design to be added to the design set. Many design variations may be realised due to the flexibility of VHDL code descriptions. However, it should be noted that variations in technology choice are limited to the synthesis tools available to the model user wishing to expand the design set.

The design choices effect not only the design of a given module instance but also the number of units of a module in the given design. In figure 2, the design parameters flit size and control refer to router design issues which effect the complexity of the address decrementer implementation. However, network degree and number of virtual channels describes the effect of the interconnection topology on the router. In other words, it describes the number of data paths which traverse the router. As the address decrementer lies on the data path, the number of data paths in the design describes the number of address decrementer modules required in the design. Since the features of the interconnection topology are also described in the parameter choice, topology changes can also be realised

	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

using the method described above.

7 Evaluation

Modelling using HDLs may be classified into two categories. The first 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 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 sub-type 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 [3].

In VHDLsyn, as stated earlier, costing is based on the number of CLBs — combinational logic blocks, in the design. To illustrate costing within the model, a cost analysis of the TRC both with an 8 bit flit, as in the original router, and with the flit size varied to 16 bit and 32 bit respectively.

This example is chosen, as it gives a somewhat simple example of both the flexibility to make design changes within the model and the importance of costing in performance modelling. The flit size represents the data bus through a router design and therefore a proposal to increase the bus width to increase performance is a fairly typical performance enhancement. However, as can be seen from the figures in table 5 this performance enhancement does not come for free.

The modules of table 5 are the same modules shown in figure 1 illustrating a path through the TRC. It should be noted that the virtual channel module, VC, consists of the components fifo, multiplexor and multiplexor controller referred to as Fifo, Mx and Ctrl respectively.

It is important not only to consider the increased cost of individual modules but the effect on the overall cost of the router as well. A number of modules are not effected by this change and therefore their respective cost figures are unchanged. These are the modules that do not lie on the data path through the router. The cost of these modules added up to approximately half the total cost in the 8 bit case and this was reduced to one third in the 32 bit case. As such, the changed modules have a proportionately larger effect on the overall cost. As shown in the results, an increase in flit size to 32 bits more than doubles the cost of the router. If cost is an issue for a particular design then it is important that this cost increase is supported by the performance increase obtained by the design change.

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 (traffic volume) and different traffic patterns may be tested. VERsim offers these possibilities.

VHDLsyn's ability to handle traffic is limited due to the data inter-

Symbol	8 bit (# CLBs)	16 bit (# CLBs)	32 bit (# CLBs)
AC	5	9	19
AD	4	8	16
AR	34	34	34
B	4	8	16
CB	40	71	135
OP	2	2	2
RD	55	55	55
SCo	12	12	12
SCi	18	18	18
SM	8	8	8
VC			
— Fifo	62	100	194
— Mx	5	9	17
— Ctrl	6	6	6
TRC Cost	255	344	532

Table 5: VHDLsyn cost results for various flit sizes

face 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.

As stated in [3] 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 [2]. A more detailed analysis such as that given in VHDLsyn can meet these goals.

Symbol	Delay (ns)	levels (#)	Type
AC	6.1	4	data
AD	5.4	3	data
AR	56.5	19	dec
B	5.4	3	data
CB	12.4	7	data
OP	10.5	5	data
RD	66.6	25	dec
SCo	34.4	12	ctrl
SCi	43.6	18	ctrl
SM	31.8	14	ctrl
VC			
— Fifo	82.2	24	data
— Mx	6.9	4	data
— Ctrl	30.2	10	dec

Table 6: VHDLsyn delay results for the modules of the TRC router

Table 6 illustrates the delay results obtainable for the TRC modules within VHDLsyn. The modular structure of these results makes it easier not only to compare individual modules but also which parts of the design effect the delays. Off course the latency(delay) of the TRC is based on the longest path delay across the router but it is also interesting to see which types of modules make up the larger delays in the circuit. Three main module types that are identified are control modules (ctrl), decision making modules (dec) and data modules (data). It is interesting to note that the data modules make up only a third of the total delay of all the modules. Decision making modules make up 40% and control makes up just under 30%. The large percentage of control related modules is due to the asynchronous nature of the circuit. Asynchronous control increases the complexity of a design.

The delay figures are expressed in two parts to reflect both the delay and logic levels crossed. As stated earlier, the net delays are based on a best case value which gives rise to very optimistic results. If a user of the model wishes to reduce the optimism in the design, the normalised rate may be scaled to the users factor of choice.

8 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 is suited are identified.

One important 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 [9]. 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 a 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 are more manageable.

Another interesting feature is the use of FPGA as the implementation technology. FPGA may be said to be essentially a synchronous technology and as such it is not expected that the speedups sought in an asynchronous design will be achieved using this technology. However, as FPGA is often used as a prototype technology it is important to show that it may be used for both synchronous and asynchronous designs allowing for the fact that speedup factors will not be achieved but testing of correct design functionality can be achieved. This is often the main test for a prototype i.e. does it work.

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 VHDLsyn. On the other hand realistic traffic modelling in VERsim can aid development of more detailed traffic contention features in VHDLsyn.

Acknowledgement

The authors are grateful for the technical advice of Jan Anders Mathisen regarding the gate-level modelling approach described in this work.

References

[1] Agarwal, A. Limits on interconnection network performance. *IEEE Trans. on Parallel and Distributed Systems 2*, 4 (Oct. 1991).

[2] Balakrishnan, S., and Panda, D. K. Impact of multiple consumption channels on wormhole routed k-ary n-cube networks. In *Proc. of the Intern. Parallel Processing Symposium '93* (1993), pp. 163–167. Ohio State University.

[3] Chien, A. A. A cost and speed model for k-ary n-cube wormhole routers. In *Proc. of Hot Interconnects '93* (Aug. 1993). Dept. CS, University of Illinois at Urbana-Champaign.

[4] Dally, W. J. Virtual-channel flow control. *IEEE Transactions on Parallel and Distributed Systems 3*, 2 (March 1992), 194–205.

[5] Dally, W. J., and Seitz, C. L. The torus routing chip. *Distributed Computing 1* (Jan 1986), 187–196.

[6] Haddow, P. C. A generalisaton of router chip design. In *5th Euromicro Workshop on Parallel and Distributed Processing* (Jan. 1997), pp. 307–313.

[7] Natvig, L. High-level architectural simulation of the torus routing chip. In *Proceedings of the 6'th International Verilog HDL Conference (IVC'97)* (April 1997), pp. 48–55.

[8] Ould-Khaoua, M., and Sotudeh, R. Performance evaluation of hypermeshes and meshes with wormhole routing. *Journal of Systems Architecture* (March 1997), 345–353.

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

[10] Vemuri, B., Mandayam, R., and Meduri, V. Performance modelling using pdl. *IEEE Computer* (April 1996), 44–53.

[11] Xilinx founation series. *http://www.aldec.com/*.