Formal Verification of the RCMP Egress Routing Logic

Palanisamy Murugesh and Sofiène Tahar

Concordia University
Dept. of Electrical & Computer Engineering
Montreal, Quebec, H3G 1M8, Canada

E-mail: {m_palani, tahar}@ece.concordia.ca

Abstract

In this paper, we present our results on the formal verification of an Egress traffic module using CTL model checking. The design considered is a commercial product obtained from PMC-Sierra, Inc. and is used by the RCMP (Routing, Cell counting, Monitoring, Policing) process in a network port interface for an ATM switch fabric. The available behavioral level VHDL design was translated to a synthesizable Verilog set and verification was carried out using the VIS (Verification Interacting with Synthesis) tool. One design error which could lead the system into a hangup state was detected through the verification of a set of liveness and safety properties. The design error was also confirmed through Synopsys-VSS and Verilog-XL simulations.

I. Introduction

ATM (Asynchronous Transfer Mode) provides a high-speed, low delay, multiplexing and switching network to support user traffic, such as, voice, data and video applications. In an ATM network, the RCMP process both the Ingress and Egress cell traffic in a network port interface for an ATM switching fabric. Main application areas for the RCMP [7] are in: 1) Edge switches, 2) Enterprise switches, 3) Core switches, and 4) Access Muxes and Residential Broad-band switches. The RCMP performs the following ATM layer functions: 1) Header translation and cell append for routing, 2) Policing, 3) Cell counting, and 4) Routing for 64K virtual channels. The Egress Routing Logic of RCMP does header translation of ATM cells and aids multicasting, thereby avoids congesting the switch fabric.

This paper particularly deals with the formal verification of the Egress routing logic of RCMP using Symbolic Model Checking [4]. Symbolic model checking provides a fully automatic verification method and allows certain properties expressed in a branching-time temporal logic, CTL (Computational Tree Logic), to be verified completely automatically [2]. Although symbolic model checking algorithms provide efficient property checking results, the automatic verification is limited by the size and the complexity of the design due to state space explosion.

In a previous work [6], we presented the RTL modeling of the RCMP Egress logic. Simulation results were obtained using Synopsys-VSS and Verilog-XL respectively. In this paper, we use a symbolic model checking tool, VIS (Verification Interacting with Synthesis), to carry out property checking on the Verilog behavioral description. Through both simulation and formal verification, a small design error which could violate the system behavior was successfully detected and corrected.

The rest of the paper is organized as follows. Section 2 describes the behavior and VHDL implementation of the Egress routing logic. Section 3 discusses about the CTL properties verified and Section 4 presents the model checking results. Section 5 concludes the paper.

II. Egress Routing Logic

The Egress routing logic of RCMP discussed in this paper does only header translation of ATM cells and does not perform any traffic shaping. A typical application of RCMP Egress routing logic is shown in Figure 1. The main function of this block is to route the cell traffic from the output of the RCMP to either the switch or one of the 32 Egress PHYsical ports, and to provide necessary control signals. The Egress routing logic performs routing using first prepend word of the cell and supports cell format of one prepend and 27 words (16-bit wide). The first prepend word contains a routing bit that indicates whether the cell should be routed to the switch or back to the Egress direction. It also contains a 5-bit PHY address for the Egress cells. The reason for using the first prepend word is to allow the Egress logic to determine the routing of the cell before the remainder of the cell arrives, thus avoiding the need to buffer the cell. The prepend word is stripped off by the Egress logic.

It also supports back-pressure control (referred as stuttering mechanism in the design) from the Switch. Assumptions like having the same clock and same data width (16-bit) for both the port interface and the Switch interface greatly simplified the design. The cell traffic coming from the switch fabric are looped back into the input cell interface of the RCMP, which treats these cells as if they are sourced from another PHY port. The RCMP takes care of the extended cell format used in the switch core by removing the appended cell octets and producing 53-octet ATM cells for the PHY layer. This is done by VC (Virtual Circuit) look-up and performing address translation and multicasting if appropriate. The advantage of doing multicasting in Egress is that it avoids congesting the Switch.

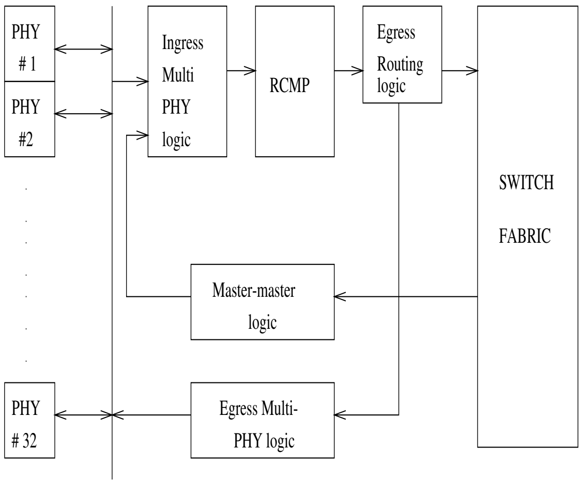

Fig. 1. A Typical Application of RCMP Egress Routing Logic

The main features of the Egress routing logic are as follows [7]:

- Support of Backward Routing of cells from the RCMP output. Decoding the prepend word to determine whether a cell is destined to the switch fabric or one of the PHY devices in the Transmit direction.
- Support of either Direct Multi-PHY interface or Indirect-addressing Multi-PHY interface for up to 32 PHY devices.
- Support of cell format with one prepend and 27-words (16-bit wide). Prepend word is stripped off.
- Support of back-pressure flow control (referred as stuttering in following sections) from the Switch.

The VHDL design of the Egress routing logic was realized using six concurrent processes, doing the functions of output assignment, state decoding, PHY and switch interface decoding and the word-counting [7]. The block diagram of the VHDL implementation can be found in [6]. The interface to the RCMP consists of the control signals which flag the cell availability, cell-start and read-enable, including 16-bit data and 2-bit parity. A cell-availability-decode block produces the cell-available signal and cell-start signal to the switch. It also senses the read-enable signal from the switch and the ready-signal from the PHY ports. A state-decode block generates write-enable signals to the 4 PHY ports and also a 5-bit address for the 32 ports connected through Egress Multi-PHY logic. The RCMP acts as a "slave" interface which receives the read-enable signal while the Switch fabric acts as a "master" sourcing the read-enable signal. The output data and the parity are connected to both the Switch and the PHY devices. Finally, a datapath retiming and output assignment block takes care of the transfer of the input data to the output at the appropriate clock cycle by providing internal buffering.

The RCMP routing logic operates in three different addressing modes, viz. Direct, Indirect and Switch interface addressing modes. In all these modes of operation, the Egress logic acts as a "slave" interface (i.e. the read-enable signal is sourced from the PHY ports or the Switch fabric) in the data transfer. In direct addressing mode, the cell transfer occurs in the Egress direction to the four PHY devices connected directly. In Indirect mode, up to 32 PHY devices can be interfaced with some extra decoding logic using 5-bit PHY address generated by the Egress logic. With the Switch interface, after the decoding of the prepend word, the Egress logic asserts the cell-availability signal and waits for the read-enable signal from the Switch fabric. With its assertion, the control is directly transferred to the Switch and the rest of the ATM cell is transferred. When the switch deasserts just after the first word or just after the last word during the cell transfer, an internal buffering mechanism called stuttering is used which holds the data till the switch is ready again to read. The cell count will stop decrementing until the read-enable signal is asserted again. The stuttering mechanism is only provided for the first and last words of a cell transfer in the switch mode. Interested readers can refer to [7] for more details on the timing diagram specifications of all the above modes of operation.

The state diagram representation of the Egress routing logic design is shown in Figure 2. It performs the following functions:

(1) It asserts the read-enable signal to the RCMP to read out the first prepend word, and then halts the cell read by deasserting the read-enable signal (states IDLE and GO).

(2) It examines the routing bit to connect the control signals from either the Switch or the Egress Multi-PHY logic (state PRE1).

(3) It strips off the prepend word for Egress cells. The prepend word is stripped off to empty this word from the RCMP output FIFO but not actually transferring the data to the destination (state PAUSE).

(4) It starts the cell transfer. If the cell is destined to the Switch, the Switch controls the read-enable signal to the RCMP. Otherwise, the Egress logic controls both the read-enable signal to RCMP and the write-enable signal to the PHY ports (state TRANSFER_CELL).

(5) The postpend word is required for other purposes and is not appended to the Egress cells by the RCMP. The Egress logic hence stops the cell transfer without postpend stripping and waits for the cell-availability signal from the RCMP.

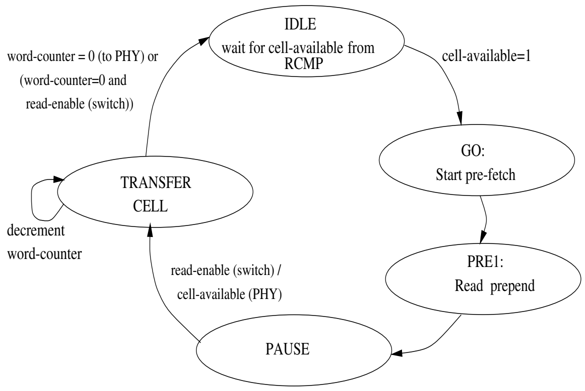

Fig. 2. State Diagram of the Egress VHDL Design [5]

III. Properties Description

VIS (Verification Interacting with Synthesis) is a tool that integrates the verification, simulation, and the synthesis of finite state hardware systems [1]. It uses a Verilog front-end and supports fair CTL model checking, language emptiness checking and hierarchical synthesis [2]. The translated Verilog description of the Egress logic was converted to a BLIF (Berkeley Logic Interchange Format)-synthesizable subset of Verilog by introducing non-determinism and symbolic variables. A stimuli-process was also incorporated into the code to generate inputs at the corresponding instances.

CTL properties comprising liveness and safety properties were considered for the verification, similar to the verification work in [3]. The safety properties included language emptiness checking and invariant checking. In the following CTL expressions, "!", "*", "+"and "→" denote logical "Not", "And","Or" and "Imply", respectively. "AG" and "AF" are CTL quantifiers meaning for all paths a formula holds in every state and for all paths a formula eventually holds in future states, respectively.

Safety Properties

Three fairness constraints (Buchii type) were specified, covering the fair paths. Similarly, invariant properties, which must be satisfied throughout the behavior of the system, were specified. They are described below:

Fairness Constraints: The following fairness constraints must be satisfied infinitely often along all the fair paths in the system.

Fair#1: $!(p\_state = pause);$

Fair#2: $!(cellcount[7:0] = 28);$

Fair#3: $!(rca\_s = 1);$

The first fairness constraint mentioned above specifies that the system must eventually leave the "pause" state. Similarly, the other two constraints state that the cell count and the signal *rca_s* should eventually take on values other than the mentioned ones.

Invariants: Properties, which model the invariant behavior of the system, must be satisfied for the proper synchronization of every cell transfer either to the PHY device or to the Switch. Following are two example invariants.

Inv#1: $(cellcount[7:0] = 0) \rightarrow (p\_state = idle);$

Inv#2: $((p\_state = transfer\_pre1) \rightarrow (cellcount[7:0] = 27));$

The first property (invariant) ascertains that the state of the system is reseted to "idle" when the cell count becomes zero. Similarly, second invariant property checks that, whenever the system is in "transfer_pre1" state, the cell count assumes the mentioned value.

Liveness Properties

Besides the safety property (invariant) checking, we checked a set of liveness properties covering the relevant expected behaviors of the Egress routing logic. In the following we describe five such properties.

Liveness#1: In the direct addressing mode, if the routing tag (*bwd_tag*) corresponds to the PHY device #1 and the system is in the "pause" state, then eventually the system must move to a "transfer_cell" state and the write-enable signal ($phy1wrenb$) for that device must be asserted.

$$AG(((bwd\_tag[7:0] = b00000011) * (p\_state + pause)) \rightarrow AF((p\_state = transfer\_cell) * (phy1wrenb = 0)));$$

Three more similar properties can be specified for the other three PHY devices in the direct mode.

Liveness#2: After the fetch of the first prepend word ($rsoc = 1$), the rest of the ATM cell is transferred to the output with the first word notification ($osoc = 1$) in all the fair paths of the system.

$$AG((rsoc = 1) \rightarrow AF(osoc = 1));$$

Liveness#3: In the indirect addressing mode for the PHY devices, the transfer of the ATM cell takes place along with the correct decoded address at the address port ($padd[4:0] = bwd\_tag[7:3]$).

$$AG((bwd\_tag[2:0] = 3) \rightarrow AF((p\_state = transfer\_cell) * (padd[4:0] = bwd\_tag[7:3])));$$

The above property takes care of the addressing of 32 PHY devices in the indirect addressing mode in the Egress direction.

Liveness#4: With the routing tag carrying the addressing of the Switch and the cell availability signal (*mca*) to the switch enabled, the system transfers the rest of the cell with the assertion of the read-enable signal from the switch.

$$AG(((bwd\_tag[7:0] = b00000000) * (mca = 1)) \rightarrow AF((p\_state = transfer\_cell)*(mrdenb = 0)));$$

Liveness#5: This property checks for the proper transition of the system from a "pause" state to a "transfer_cell" state.

$$AG((p\_state = pause) \rightarrow AF(p\_state = transfer\_cell));$$

IV. Model Checking Results

Initially, the Egress Routing logic was successfully verified against the fairness constraints to remove the presence of any non-trivial behavior. Similarly, the invariants were checked successfully. The liveness properties modeling the ATM cell transfer in all the three modes of operation (Direct, Indirect Multi-PHY modes and Switch mode) were checked on the Verilog model. The verification of all the properties were done employing dynamic variable ordering, which reduces the CPU time by choosing an optimized order for the variables [3]. The CPU time (in seconds), memory usage (in MB), and the number of BDD nodes allocated for every property checking are shown in Table I. Experimental results were obtained on an 296 MHz UltraSparc2 with 768 MB of memory.

Verification	CPU Time	Memory	BDDs
Fair#1	89.2	6.3	4820221
Fair#2	88.2	8.2	7626808
Fair#3	302.4	12.2	22424387
Inv#1	20.1	3.3	8286368
Inv#2	15.3	2.9	5132624
Liveness#1	70.2	5.8	1989467
Liveness#2	23.4	3.9	3985431
Liveness#3	25.5	4.5	2436219
Liveness#4	29.6	6.3	8065623
Liveness#5	400.4	6.0	7341856

TABLE I
Property Checking Results

During the property checking of *Liveness#5*, a system hangup error in the "pause" state was notified by VIS producing the corresponding error trace. It was found that improper synchronization between the processes led the system to this state and was rectified. This was also confirmed through VSS and Verilog-XL simulations [6]. The experiment results for *Liveness#5* in Table I are for the corrected model.

	Combinational Nodes	Latches
Original Model	1460	131
Reduced Model	1088	87

TABLE II
Latch Reduction Using Partial Abstraction

To obtain a possible latch reduction, abstraction was applied wherever necessary. The input and output data were abstracted out to 8 bits from 16 bits. The cell count register was reduced from 8 bits to 5 bits as it takes integer values from 0 to 28. Further reduction in the input/output datapath was not possible because of the routing tag transferred in the same datapath [5]. Using this latch reduction approach, the total number of latches in the design was reduced from 131 to 87 (see Table II). We noticed, however, that even with partial latch reduction employed, the verification CPU time and memory usage were not considerably reduced for the properties described in the previous section. Table III illustrates two verification examples obtained using the reduced model with latch reduction.

Verification	CPU Time	Memory	BDDs
Fair#3	263.1	8.6	15826146
Liveness#5	373.4	4.6	5611521

TABLE III
Property Checking Using Latch Reduction

V. Conclusions

In this paper, we presented the formal verification of the Egress routing logic of the RCMP through property checking using the VIS tool. A design error which could lead the system to a deadlock "pause" state was detected and rectified through the verification of CTL properties on the Egress Verilog model. The CTL properties covered most relevant behaviors of the model in all the addressing modes except the "stuttering mechanism". This was not possible because it involves the introduction of timing delays in the input assignments to the model while VIS does not accept a timed Verilog model. However, the correctness of the stuttering mechanism was established through Verilog-XL simulation. Formal verification and simulation are hence a perfect complement to each other to achieve a higher coverage of errors. The verification in VIS was done in a reasonable CPU time as the design consisted of only 131 latches in total [6].

References

[1] R. Brayton et al. VIS: A System for Verification and Synthesis, Technical Report UCB/ERL M95, Electronics Research Laboratory, University of California, Berkeley, December 1995.

[2] E. M. Clarke and E. A. Emerson. Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic, *ACM Transactions on Programming Languages and Systems*, Vol.8, No.2, April 1986, pp. 244-263.

[3] J. Lu and S. Tahar: Practical Approaches to the Automatic Verification of an ATM Switch Fabric using VIS, *Proc. IEEE 8th Great Lakes Symposium on VLSI (GLS-VLSI'98),* Lafayette, Louisiana, USA, February 1998, IEEE Computer Society Press, pp. 368-373.

[4] K. L. McMillan. *Symbolic Model Checking,* Kluwer Academic Publishers, Norwell Massachusetts,1993.

[5] M. Palanisamy and S. Tahar. Functional Verification of RCMP Egress Routing Logic, Technical Report, Department of Electrical and Computer Engineering, Concordia University, Montreal, Canada, September 1998.

[6] M. Palanisamy and S. Tahar. RTL Design and Verification of the RCMP Egress Routing Logic, *Proc. IEEE Canadian Conference on Electrical & Computer Engineering(CCECE'99),* Edmonton, Alberta, Canada, May 1999.

[7] PMC Sierra, Inc. RCMP Egress Routing Logic in VHDL, March 1997. http://www.pmc-sierra.com/register/rcmp800.asp