ISSN: 2395-1680 (ONLINE)
DOI: 10.21917/ijme.2022.0226

OPTIMISED META-HEURISTIC QUEUING MODEL IN VLSI PHYSICAL DESIGN

L. Mary Florida[1], S. Brilly Sangeetha[2] and K. Krishna Prasad[3]
[1]*Department of Mathematics, St. Xavier's Catholic College of Engineering, India*
[2]*Department of Computer Science and Engineering, IES College of Engineering, India*
[3]*College of Computer Science and Information Science, Srinivas University, India*

Abstract

In this paper, the Markov M/D/1/B queuing model that has been created will be discussed. One approach to evaluating the effectiveness of a router is to use a model of a queue that is analogous to the one that we presented above. Having said that, this can be accomplished in a variety of different ways. According to the results of the studies, altering the service rate has a discernible bearing not only on the amount of data the system processes but also on the effectiveness of its operation. For the purpose of demonstrating the viability of our methodology, we built an output-queuing router using FPGA. When utilising this method, it is possible to observe that the queues and the control unit of the router take up an excessive amount of space on the silicon. This is a consequence of the fact that this method is utilised. The difference in effectiveness between a theoretical model and a real prototype is only 2%. This is a very small margin. The study implemented our design on a Xilinx FPGA Vertix II Pro family 100K chips. In the following subsections, we present the router FPGA synthesis results and evaluate its performance.

Keywords:

Antenna, Equivalent Circuit Modelling, RLC Circuit, Series Resonance, Q Factor, Bandwidth

1. INTRODUCTION

Multiple processors and computers, working in parallel, have the capacity to give exceptionally high performance at a relatively low cost [1]. The rate at which individual processors in a multiprocessor or multicomputer system may communicate with one another is an extremely important performance metric [2]. When it comes to communication, multiprocessors that have more than 64 nodes require multistage interconnection networks that have $n \times n$ switches (typically, $2 \leq n \leq 10$) [3].

In multi-computers, point-to-point dedicated connections are handled by a limited number of communication coprocessors [4]. These coprocessors have n-1 ports that connect to neighbouring nodes and one port that connects to the local application processor that can communicate in both directions [5]-[6]. The conception of high-performance small $n \times n$ switches is very important to the development of multiprocessor and multicomputer systems [7]. Because a large system requires a great number of these $n \times n$ switches, there is a considerable incentive to build each switch as a distinct VLSI chip [8]-[10].

Networks-on-chip, sometimes known as NoC, is the name of the latest trend in the industry of interconnecting modules contained within silicon chips [11]. The router is one of the most important modules in any NoC-based architecture, as the design of the router has a direct impact on the overall performance of the system [12]. Because of the significant impact that routers have on performance, thorough modelling and analysis of router designs are very necessary in order to supply the designer with a better understanding of network performance at higher abstraction levels and an early estimation of that performance [13]. It is necessary to approach queue modelling from a different angle than the majority of computer network researchers have in order to correctly represent NoC [14]. This is required in order to adequately describe the NoC [15].

NoCs are noticeably superior to computer networks in terms of both speed and the amount of silicon they require [16]. In the past, it has been shown that queues and control units use more than 86 [17]. 2% of the silicon area of the router [18]. This shows how important it is to choose the right queue parameters [19].

When applied early on in the design process, good queue modelling makes it easier to make more precise parameter estimates [20]-[25]. It was described how a delay model for NoC routers may be used to investigate the effects of changing the queue size and the number of ports on the throughput and delay of the router [16]. The model was shown as part of the presentation [27]. In this work, a M/D/1/B queuing model for a NoC router is developed, and many implementation options are examined.

The 2D model provides an accurate assessment of the performance of a queue when a deterministic service rate is applied to the queue [28]. Applications that deal with multimedia and telecommunications usually call for a predictable access rate between the many IP cores that make up the network [29]. This queue can be used to accomplish whatever it is looking to do [30]. This demonstrates the feasibility of constructing such a queue on silicon using multiple different strategies [31].

2. METHODS

The Fig.1 presents an illustration of the architecture of an n-port output queuing router. The Fig.1 illustrates two distinct architectures for output queuing routers. Both of these are shown. The switch fabric (SF), the input buffer, the output queue, and the output buffer are the four key components that make up the data path of the router. The controller is implemented with the help of several rather simple finite state machines. The presence of a packet-ready (PR) signal on a router input port indicates that data packets are coming asynchronously. The purpose of the operation plays a role in determining the rate at which packets arrive.

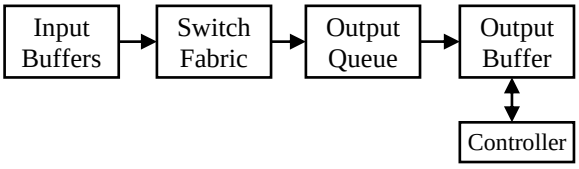

Fig.1. Output queuing router architecture

After reading the packet header, the controller will then forward the data to the correct output queue so that it may be processed. The routing tables of the controller are utilised in the

production of configuration commands. When there are n input ports and n output ports, a type of buffer known as a FIFO is utilised. At the output port, a round-robin scheduling mechanism serves the backlogged queues in a sequential order.

A signal denoted as Packet Sent (PS) is used to transfer data packets to the subsequent hop in the network. It is possible for output queue routers to experience packet loss due to destination statistics. As a consequence of this, bursty behaviour, in which packets arrive in adjacent clock cycles from the same input port and target the same output port, could lead to the loss of a packet. We can figure out the size of the queue by looking at how the source burstiness is spread out. This makes it less likely that any packets will be dropped.

3. M/D/1/B QUEUE MODELING

A significant number of routers that are based on NoCs use an M/D/1/B queuing mechanism. In the case of dynamic memory refresh routines, deterministic service rate queues are the most appropriate type of queue to use for the traffic characteristics that exist between the memory and the microprocessor. In this section, a queue model for routers that has a deterministic service rate and a random arrival rate is constructed. The arrival probability a, the loss probability L, and the service rate μ for a queue of size B are the inputs that drive the model. Assuming that T is a discrete time step, we will assume that only one packet will arrive at each time step and only one will leave.

$$T < \min\{\tau_a,\tau_s\}$$

where

τ_a - minimum inter-arrival time and

τ_s - service time.

When a Markov chain analysis is performed, the states of the state transition diagram are represented as columns. These columns reflect the occupancy of the queue, beginning with the state S_0, which is empty, and ending with the state S_B, which is totally full. The number of rows in the diagram transitions is determined by the packet service rate. This model utilises round-robin scheduling, which ensures that the service rate remains constant at $1/n\tau$.

3.1 OBJECTIVE FUNCTION

When dealing with algorithms that are based on a single solution, it is essential to do incremental evaluations of the objective function in order to cut down on the evaluation complexity of the neighbourhood. In light of the quadratic assignment problem, we will make an effort to find an answer to this question throughout this lesson. The QAP has a wide variety of applications, some of which include the synthesis of images, the analysis of data, the design of VLSIs, and the positioning of facilities. The following phrases are a concise explanation of the problem.

Discover an object-to-location mapping, denoted by m:$O \rightarrow L$, that reduces the value of the objective function f to the minimum, given n individual objects. $O = O_1, O_2,...,O_n$, a set of n locations denoted by $L = L_1, L_2,... L_n$, a flow matrix denoted by C, in which each element c_{ij} represents a flow cost between the objects O_i and O_j, and a distance matrix denoted by D, in which each element denotes a distance between the objects O_i and O_n.

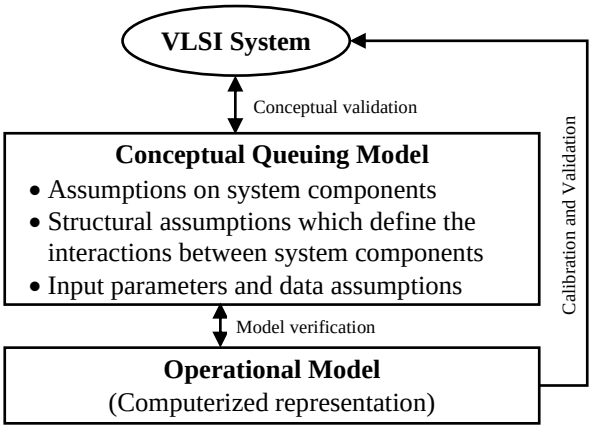

Fig.2. Modelling of NoC using queuing Model

An incremental function needs to be defined in order for the following encoding and the move operator to work properly. A solution is encoded as a permutation of n numbers $s = (l_1, l_2,..., L_n)$, and this is done. The integer l_i represents the position of the object O_i, and it can be written as: The move operator is used to swap the positions of two elements of the permutation.

4. RESULTS AND DISCUSSION

Matlab is being utilised in the process of developing a piece of software that will be responsible for generating the curves required to express these relationships. The queue size and service rate requirements vary depending on the NoC topology that is being used.

In the following section, we'll take a look at the inner workings of an FPGA-based router. To illustrate the usefulness of the queue architecture that we proposed, we will use the planning and execution of NoC application development as a case study. The next subsections will discuss the difficulties encountered during design and implementation.

The implementation of queues frequently makes use of FIFO buffers. It is possible to select either a synchronous or an asynchronous design; a one-port or a dual-port memory; whether all packets arrive with equal priority or an arbitration rule; the type of scheduler at the output (such as static priority, round robin, etc.); and many other options. The implementation of FIFO allows for a wide variety of architectural choices.

Words arranged in a list are the fundamental components that comprise one memory. These dimensions (the buffer length) are chosen by the designers so that they accurately represent the traffic characteristics. The length of each word determines the width of a packet in equal measure (assuming a fixed packet size). The performance of the system may suffer if an inaccurate estimate of the buffer length is used in conjunction with a burstiness ratio that is greater than the capacity of the buffer. On the other hand, an excessive amount of silicon area is required if the buffer length is significantly larger than the actual number of bursty packets. This causes an inefficient use of silicon resources. Because of this, the modelling method is very important when deciding the length of the buffer to use.

Within the context of this experiment, the controller is responsible for managing an internal system clock as well as a FIFO, and external interfaces are synchronised with the internal

system clocks. In the control register, you will find a set of flags that display the current status of the FIFO as well as pointers that allow you to write to or read from any memory address.

There have been a lot of different FIFO buffer topologies and IP cores presented. The problem of synchronisation, the difficulty of the controller, and the constraints imposed by available space are some of the design concerns that go into the creation of NoC systems. Using our model, there are a few different approaches one can take to get the most optimal queue design parameters. An output-queuing router for NoC applications uses an asynchronous FIFO as a queue.

In order to get the optimal queue size, it is necessary to define the following design factors first: the chance of a packet arriving (a), the probability of it being lost (L), and the number of ports that are required (n). The traffic model that is being used will determine how likely it is for a packet to arrive at its destination. Although a number of different traffic models are used to define the simulation packet transfer rate, the target application has a considerable influence on which traffic model is advised.

MPEG-2 video programmes that are based on NoC, for instance, feature self-similar communication between the modules that make up its internal structure, although other distributions have been used for a variety of different target applications. The researchers expressed the simulated traffic arrival rates by using a traffic model called a uniform model, which is an example of a traffic model that is used by researchers to evaluate the performance of designs. Depending on the application, there are many different methods in which traffic models can be put into practise.

4.1 DESIGN-COST TRADEOFFS

Given a specified shortage or waiting cost function the analysis is straightforward.

$$Min\ TC = WC + SC$$

WC = Expected Waiting Cost (shortage cost) per time unit

SC = Expected Service Cost (capacity cost) per time unit

TC = Expected Total system cost per time unit

4.2 LINEAR WAITING COSTS

Expected Waiting Costs as a function of the number of customers in the system

$$WC = C_w \sum_{n=0}^{\infty} nP_n = C_w L$$

Expected Waiting Costs as a function of the number of customers in the queue

$$WC = C_w L_q$$

C_w = Waiting cost per customer and time unit

C_wN = Waiting cost per time unit when N customers in the system

4.3 ANALYZING SERVICE COSTS

The expected service costs per time unit, SC, depend on the number of servers and their speed

$$SC = c*C_S(\mu)$$

where,

c-number of servers,

μ-average server intensity and

$C_S(\mu)$-expected cost per server and time unit as a function of μ.

4.4 DETERMINING μ AND C

Both the number of servers and their speed can be varied: Usually only a few alternatives are available

$$\underset{\mu \in A, c=0,1,...}{Min}\ TC = c \cdot C_s(\mu) + WC$$

where, A - set of available μ - options

Enumerate all interesting combinations of μ and c, compute TC and choose the cheapest alternative.

Table.1. Design Cost ($) of NoC

Implementation	M/D/1	M/D/1/B	Proposed M/D/1/B
Registers	0.8720	0.8770	0.8851
Mux	0.8602	0.8652	0.8732
Tristates	0.8748	0.8799	0.8880
Basic cells	0.8574	0.8624	0.8703
Flip flops	0.8637	0.8687	0.8767
Clock buffer	0.8346	0.8394	0.8472
LUT	0.8717	0.8767	0.8848
Slices	0.8331	0.8379	0.8456
FSM	0.8873	0.8925	0.9007
Adders	0.8597	0.8647	0.8726
Counters	0.8574	0.8624	0.8704
Equivalent gates	0.8511	0.8561	0.8639

Table.2. Waiting Cost of NoC

Implementation	M/D/1	M/D/1/B	Proposed M/D/1/B
Registers	0.8933	0.9048	1.0023
Mux	0.8813	0.9021	1.0010
Tristates	0.8963	0.8983	0.9758
Basic cells	0.8784	0.8612	0.9563
Flip flops	0.8849	0.8888	0.9949
Clock buffer	0.8551	0.9064	0.9594
LUT	0.8930	0.9079	1.0041
Slices	0.8535	0.8947	0.9694
FSM	0.9091	0.9008	0.9897
Adders	0.8808	0.9142	1.0004
Counters	0.8785	0.8662	0.9564
Equivalent gates	0.8720	0.8986	0.9804

Table.3. Service cost of NoC

Implementation	M/D/1	M/D/1/B	Proposed M/D/1/B
Registers	1.0081	1.0174	1.0269
Mux	1.0068	1.0161	1.0255
Tristates	0.9815	0.9905	0.9997
Basic cells	0.9619	0.9708	0.9798
Flip flops	1.0007	1.0099	1.0193
Clock buffer	0.9649	0.9738	0.9829
LUT	1.0099	1.0192	1.0287
Slices	0.9750	0.9840	0.9932
FSM	0.9955	1.0046	1.0140
Adders	1.0062	1.0154	1.0249
Counters	0.9619	0.9708	0.9798
Equivalent gates	0.9861	0.9952	1.0044

Table.4. Speed of NoC

Implementation	M/D/1	M/D/1/B	Proposed M/D/1/B
Registers	0.8632	0.8682	0.8762
Mux	0.8088	0.8135	0.8210
Tristates	0.8617	0.8667	0.8747
Basic cells	0.8973	0.9025	0.9109
Flip flops	0.8596	0.8646	0.8726
Clock buffer	0.8843	0.9323	0.9922
LUT	0.8286	0.8524	0.9317
Slices	0.8828	0.8878	0.9611
FSM	0.9193	0.9442	1.0009
Adders	0.8807	0.8975	0.9798
Counters	0.8843	0.9323	0.9922
Equivalent gates	0.9980	1.0072	1.0165

In this particular case study, the input/output parameters that are used for the router input/output ports are a value of 0.5 and a value of 12. Using eight different VHDL traffic sources and sinks modules, a star network architecture was created and implemented. While the traffic sinks observed and analysed the network traffic and produced the data necessary to assess the efficiency, the traffic sources in this experiment created fixed packets that were 16 bits in length. For the implementation of our concept, we relied on a Xilinx FPGA Vertix II Pro family of 100 K. After that, the study show the results of our research into how well our router FPGA synthesis works and explain what we've learned.

In our simulation experiments, we used uniform traffic model to express the predefined traffic arrival rate as an example of one of the traffic models used by researchers to evaluate the design performance.16 Different traffic models can be easily applied according to the target application. As a case study of the router performance, we assume the following parameters at the router input/output ports: $a = 0.125$, $n = 8$.

Eight traffic sources/sinks VHDL modules were designed to implement a star network topology. The traffic sources were used to generate fixed packets of 16-bit length, whereas the traffic sinks were used to monitor and analyze the network traffic and output the data required to calculate the efficiency.

5. CONCLUSION

This paper presents the Markov M/D/1/B queuing model that has been developed. Using a queue model similar to the one that we showed here is one way to analyse the performance of a router. However, there are multiple ways in which this may be done. The trials indicate that when the service rate is changed, this has a significant impact on both the throughput and the efficiency of the system. As a proof-of-concept, we constructed an output-queuing router on FPGA using our model. As a consequence of using this method, it is possible to see that queues and the control unit of the router take up an excessive amount of space on the silicon. There is only a two-percent difference between the efficiency of a theoretical model and that of an actual prototype.

REFERENCES

[1] Ikeguchi, T., & Aihara, K. (2008). Meta-Heuristic Algorithms with Chaotic Neuro-Dynamics for Solving Combinatorial Optimization Problems. *IEICE Proceedings Series*, *42*(A3L-F3).

[2] Alagarsamy, A., & Gopalakrishnan, L. (2016, May). SAT: A new application mapping method for power optimization in 2D—NoC. In *2016 20th International Symposium on VLSI Design and Test (VDAT)* (pp. 1-6). IEEE.

[3] Areibi, S., & Vannelli, A. (2000). Tabu search: A meta heuristic for netlist partitioning. *VLSI Design*, *11*(3), 259-283.

[4] Gonsalves, T., & Itoh, K. (2006). Simulated Annealing In The Optimization Of Collaborative Systems Operation. *Journal of Integrated Design and Process Science*, *10*(3), 87-95.

[5] Mao, F., Xu, N., & Ma, Y. (2009, November). Hybrid algorithm for floorplanning using B*-tree representation. In *2009 Third International Symposium on Intelligent Information Technology Application* (Vol. 3, pp. 228-231). IEEE.

[6] Emmert, J. M., Lodha, S., & Bhatia, D. K. (2003). On using tabu search for design automation of VLSI systems. *Journal of Heuristics*, *9*(1), 75-90.

[7] Priyadarshini, R., Barik, R. K., & Mishra, B. K. (2020). Meta-Heuristic and Non-Meta-Heuristic Energy-Efficient Load Balancing Algorithms in Cloud Computing. In *Modern principles, practices, and algorithms for cloud security* (pp. 203-222). IGI Global.

[8] Rajesh, K., & Pyne, S. (2021). Invasive weed optimization based scheduling for digital microfluidic biochip operations. *Integration*, *76*, 122-134.

[9] Wang, R., & Lu, J. (2021). QoS-Aware Service Discovery and Selection Management for Cloud-Edge Computing Using a Hybrid Meta-Heuristic Algorithm in IoT. *Wireless Personal Communications*, 1-14.

[10] Cao, K., Cui, Y., Liu, Z., Tan, W., & Weng, J. (2021). Edge intelligent joint optimization for lifetime and latency in

large-scale cyber-physical systems. *IEEE Internet of Things Journal*.

[11] Xie, G., Xiao, X., Peng, H., Li, R., & Li, K. (2021). A survey of low-energy parallel scheduling algorithms. *IEEE Transactions on Sustainable Computing*.

[12] Tariq, U. U., Ali, H., Liu, L., Panneerselvam, J., & Zhai, X. (2019). Energy-efficient static task scheduling on VFI-based NoC-HMPSoCs for intelligent edge devices in cyber-physical systems. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *10*(6), 1-22.

[13] Jayabalan, M., Srinivas, E., Shajin, F. H., & Rajesh, P. (2021). On Reducing Test Data Volume for Circular Scan Architecture Using Modified Shuffled Shepherd Optimization. *Journal of Electronic Testing*, 1-16.

[14] Cao, K., Zhou, J., Wei, T., Chen, M., Hu, S., & Li, K. (2019). A survey of optimization techniques for thermal-aware 3D processors. *Journal of Systems Architecture*, *97*, 397-415.

[15] Kumar, R., & Banerjee, N. (2011). Multiobjective network topology design. *Applied Soft Computing*, *11*(8), 5120-5128.

[16] Youssef, H., Sait, S. M., & Ali, H. (2003). Fuzzy simulated evolution algorithm for VLSI cell placement. *Computers & Industrial Engineering*, *44*(2), 227-247.

[17] Amiri-Zarandi, M., Safaei, F., & Roozikhar, M. (2015). Performance evaluation of generic multi-stage interconnection networks with blocking and back-pressure mechanism. *The Journal of Supercomputing*, *71*(3), 1038-1066.

[18] Rodrigues, D., Souza, A. N., & Papa, J. P. (2017, October). Pruning Optimum-Path Forest Classifiers Using Multi-Objective Optimization. In *2017 30th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)* (pp. 127-133). IEEE.

[19] Torkzadeh, S., Soltanizadeh, H., & Orouji, A. A. (2021). Energy-aware routing considering load balancing for SDN: a minimum graph-based Ant Colony Optimization. *Cluster Computing*, *24*(3), 2293-2312.

[20] Sadatdiynov, K., Cui, L., Zhang, L., Huang, J. Z., Salloum, S., & Mahmud, M. S. (2022). A review of optimization methods for computation offloading in edge computing networks. *Digital Communications and Networks*.

[21] Giagkos, A., & Wilson, M. S. (2014). BeeIP–A Swarm Intelligence based routing for wireless ad hoc networks. *Information Sciences*, *265*, 23-35.

[22] Youssef, H., Sait, S. M., & Khan, S. A. (2001, March). Fuzzy evolutionary hybrid metaheuristic for network topology design. In *International Conference on Evolutionary Multi-Criterion Optimization* (pp. 400-415). Springer, Berlin, Heidelberg.

[23] Lodha, S. K., & Bhatia, D. (1998, September). Bipartitioning circuits using TABU search. In *Proceedings Eleventh Annual IEEE International ASIC Conference (Cat. No. 98TH8372)* (pp. 223-227). IEEE.

[24] Ali, H., Tariq, U. U., Hardy, J., Zhai, X., Lu, L., Zheng, Y., ... & Antonopoulos, N. (2021). A survey on system level energy optimisation for MPSoCs in IoT and consumer electronics. *Computer Science Review*, *41*, 100416.

[25] Nguyen, V. T. N., & Kirner, R. (2015). Throughput-driven partitioning of stream programs on heterogeneous distributed systems. *IEEE Transactions on Parallel and Distributed Systems*, *27*(3), 913-926.

[26] Chaudhry, R., Tapaswi, S., & Kumar, N. (2019). A green multicast routing algorithm for smart sensor networks in disaster management. *IEEE Transactions on Green Communications and Networking*, *3*(1), 215-226.

[27] Chaudhry, R., Tapaswi, S., & Kumar, N. (2019). A green multicast routing algorithm for smart sensor networks in disaster management. *IEEE Transactions on Green Communications and Networking*, *3*(1), 215-226.

[28] Furuholmen, M., Glette, K., Hovin, M., & Torresen, J. (2010, July). A Coevolutionary, Hyper Heuristic approach to the optimization of Three-dimensional Process Plant Layouts—A comparative study. In *IEEE Congress on Evolutionary Computation* (pp. 1-8). IEEE.

[29] Siddavaatam, P. (2018). *A Delta Diagram Synthesis for IoT Optimization with Grey Wolf Driven Multi-Objective Automation* (Doctoral dissertation, Ryerson University).

[30] Sahu, P. K., & Chattopadhyay, S. (2013). A survey on application mapping strategies for network-on-chip design. *Journal of systems architecture*, *59*(1), 60-76.