International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Design of an Error Detection Fault Tolerant Arbiter for a Network on Chip

[1]Malathi Naddunoori, [2]Dr. Devanathan M.

Submitted: 30/12/2023 **Revised**: 04/02/2024 **Accepted**: 12/02/2024

Abstract: Device to Device communications and System-on-Chip (SoC) communications needs thehigh-speed data transfer with low hardware resource utilization. However, the conventional methods have resulted in higher area, high power consumption including time delay .This paper thus proposes the concept of Fault Tolerant Arbiter based Network on Chip (FTA-NoC)architecture with FIFO-Buffer, crossbar switching, route control, and arbiter modules. This work presents encoder and decoder-based route controlling using fault tolerant arbiter, which was introduced for high speed, error-free route calculation with less hardware resources in NOC.Initially, data generated from the different devices is stored into FIFO-Buffer logic, which allocates the data based on IP addresses. Then, route controller module controls the different routers in crossbar switching, which arecontrolled by priority-based scheduling. The data is then sent from the source to the destination using arbitrator in accordance with request levels. The simulation results show how well the FTA-NOC performs in terms of area, latency, and power when compared to the most advanced NoC architectures.

Keywords: *Route-Controller Network on Chip, System on Chip, FIFO-Buffer, Crossbar switching, route control, Arbiter.*

1. Introduction

Thehuman existence has been greatly impacted by the fascinating advancements of contemporary electronic technology, such as pervasive and ubiquitous computing, ambient intelligence [1-2], communication, and Internet. Today micro-electronic gadgets are affecting the means of communication, learning and enjoyment. The primary driving factor for the improvements throughout decades is the SoC technologies [3], where sophisticated packages are combined into single VLSI chips. Not only functionally enhanced, these items such as mobile phones, laptops and personal portable sets are growing quicker, smaller-in-size, larger-in-capacity, lighter in weight, lower-in-power-consumption and cheaper. One may pleasantly believe that this tendency would constantly continue. Following this trend, we may incorporate more and more complicated applications and even systems into a single chip [4].

However, our existing approaches for SoC design and integration do not uniformly progress because to the enormous hurdles encountered. Because the broad variety of IP modules in SoCs expands [5-6], bus-based fully connectivity topologies may also prohibit such structures to satisfy the overall performance necessary via many applications. For structures within the depth parallel verbal exchange requirements the buses do not tend to meet required metrics like latency, bandwidth and input power .The ultimate solution for overcoming this bottleneck issue focuses on an integrated switch connected network, dubbed NoC [7], which connects all the Intellectual Property units predominantly . This shows that the system area of the Network on chips is significantly greater in comparison to the conventional bus based approach because this ensures to implement various arbitration for routing algorithms[8]. Inclusively the NoCs employ a specific feature of inbuilt redundancy which handles tolerance errors simultaneously overcomes all the bottlenecks in communication infrastructure. This aims to offer more scope to the VLSI designer good insight to develop viable solutions for diverse system features and restrictions.

In the current day, there may be a desire for convergence of varied applications (video, conversation, computing and so on.) onto a single IC [9]. Normally, when any of these Packages are employed in a stand -alone way, they have got resources that are isolated and customized to them [10]. However, at this time, those apps are required to share a number of the previously separate assets that make it possible for them to harmoniously work as a unit after being integrated onto a SoC. One example of this would be from the point of view of quality of service. Moore's law [11], which applies to the logic and memory products of semiconductors, is driving the flaming scale of integration of many IP cores on a single chip. Additionally, as a result of Moore's law, a wide variety of diverse technologies are improving, although at a more gradual rate. According to the "More than Moore" (MTM) technique [12], many new capabilities are now

[1]*Research Scholar, School of ECE, Reva University, Bengaluru, Karnataka, India. Email: n.malathiraj@gmail.com*
[2]*Associate Professor, School of ECE, Reva University, Bengaluru, Karnataka, India. Email: devanathan.m@reva.edu.in*

emerging as a result of the fact that those technologies are becoming more accessible. These trends are also contributing to the phenomenal expansion ofSoCs.The significant findings in the proposed work are listed in the following order :

- Implementation of FTA-NoC with FIFO-Buffer, crossbar switching, route control, and arbiter modules.
- FIFO-Buffer logic is developed for storing of data generated from the different devices based on IP addresses.
- Route Controller logic is developed for assigning the route priorities between input and output devices using priority-based scheduling.
- In accordance with request levels, arbitrator logic is created to transfer data from the source to the destination.
- When compared to state-of-the-art NoC architectures, the simulation results of the proposed FTA-NoC clearly show that the FIFO-Buffer, crossbar switching, route control, and arbiter modules performed better in terms of area, power consumption, and delay.

The remaining portions of the suggested paper are organized as follows: section 2 provides literature and problem analysis; section 3 lists a detailed analysis of the novel concept of FTA-NoC; section 4 provides results and output analysis; and section 5 concludes with future possibilities.

2. Literature survey

In [13] authors implemented the low-complexity NoC with the growing complexity of structures and trends in technology, the pins and wires that manage interconnections among systems and components are scaling down at a slower rate than the components themselves. This is because the scaling of transistors has been helped by the trends within the silicon processes of chip fabrication [14], which help development in device architectures for SoC design. In [15] authors implemented the NoC with Globally Asynchronous and Locally Synchronous (GALS) method and formed GALS-NoC.In addition, synchronization of approaching chips with a single clock and a negligible amount of distortion may be very difficult to accomplish, which makes use of a number of different clocks. In [16] authors implemented the synchronization module concept for next processors. Instead of logic being the limiting element in modern systems, the transfer of data across resources is becoming the bottleneck for cost, performance, size, and power consumption. In [17] authors implemented the frequency based NoC (DoS-NoC) systems. In addition, the frequency at which components communicate with one another is far lower than the clock rates of modern CPUs. These elements quickly mix with SoCs, resulting in a shift toward a communication-focused orientation. However, scaling [18] wires at the same rate as transistors has proven problematic, and as a result, gates now cost far less than wires do, in terms of both the amount of space they take up and the amount of performance they provide, in comparison to a number of basic NoC approaches [19].

Since of this, the buses used in SoC designs [20], which have for a long time been the backbone of device interconnects, are becoming incapable of keeping up with the expanding system performance needs. This is a problem because busses in NoC [21] have long been the backbone of device interconnects. In the field of device interconnects, there were several emerging trends, including crossbars [22] and a great number of others. may provide a solution to the problem of inadequate communication.

In [23] authors implemented the hybrid crossbar switching based NoC. A report for the semiconductor industry is provided here, and it is derived from the ITR for Semiconductors [24]. Silicon is used in the production process of this technology, which is a step toward the growth of the IC industry. It is very evident [25], based on the roadmaps, that the reports for the new research projects focus on the internet of things. In [26] authors implemented the high speed NoC (HS-NoC). In addition to this, it places an emphasis on wireless technologies in order to deliver the finest solution. It is thus of the utmost importance that the device's overall energy usage be cut down to a far larger degree. In [27] authors implemented the cost effective NoC (Connect-NoC)design, which is effective in efficiently providing bandwidth and quality of service (QOS) in traffic flows. As a direct result of this, the need for model-wide synchronization is reduced. In [28] authors implemented the NoCusing parallelism,which minimizes the need of global communication cables, and reducing the overall power consumption of the chip. All of these benefits come as a result of the method's use of parallelism. Not only can the quantity of connections cables be reduced [29], but also the amount of network traffic may be monitored and controlled to significantly reduce the amount of power that is used. In addition, the convenient clock speed and device controller based NoC (DC-NoC) voltage may both be adjusted in line with the amount of bandwidth that is now accessible. The designers of the networks need to appropriately handle a large number of issues in order to provide better results from the chip.

3. Proposed NOC Design

The methodology of transmitting data and sharing of information among the sources in an NoC architecture is very crucial with two major objectives. The first one is its size which helps to increase the hardware resources individually, making then to become as standalone blocks, at the same time making the NoC structure interconnected with blocks inside the NoC. Second, the network is scalable and configurable makingit a flexible platform which makes its adaptable to a wide variety of workloads as it still maintains the generality of software development methodologies .As the network is reliable, scalable in bandwidth, energy efficient with distributed routing decision it becomes reusable and all these features make the the NoC as a viable solution to create routers with a large scope in NoC design. Ultimaltely all this is the result router development. Fig. 1 gives a block diagram of proposed FTA-NoC architecture. The FTA-NoC has four input and output ports along with four output/intermediate ports and all these are linked together with an intermediate crossbar. To make sure that effectively the transmission of information is done with the Processing Elements (PEs), the local input and output ports are connected directly to the datapackets .Initially the data that sent by the different devices will be saved to FIFO-Buffer logic, allocation of data is done depending on the corresponding IP addresses of devices . The route controller module controls various routers . All these routers are handled by priority-based scheduling. Traffic in the routes is managed using an allocator called static allocator (SSA).This static straight allocator usually synchronizes encoder-decoder modules .In the the next step switch allocation (SA) and virtual channel allocations (VCA) are grouped together to form virtual channel -switch allocation (VSA). The FTA arbiter thus controls the different routes along with checking error correction operation, occurring at the time of data transmission.Delays in the network are minimized by a module called look-ahead bypass route computation (LBRC).Hence ,after all the above processes the output data is retrieved after decoding of data successfully.

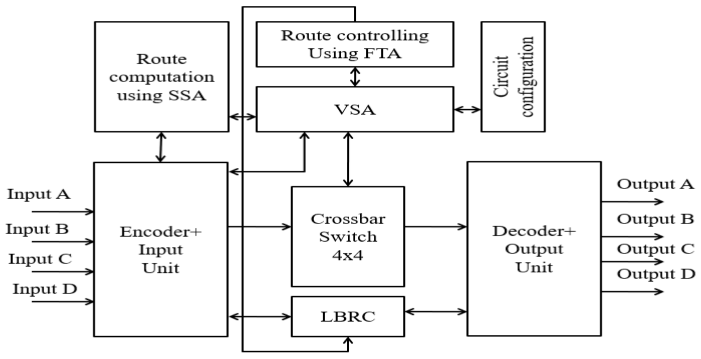


Figure 1. Proposed FTA-NoC architecture.

3.1FIFO-Buffer Architecture

A schematic representation of the buffer's architecture can be seen in Figure 2, and it can be broken down as follows: input/output data; processing circuit; clock operation; Read Access Memory (RAM); multiplexer; data ack action; state machine; and logical operation. The performance of the network is analyzed in order to retain data whenever a flit becomes congested with many priorities.

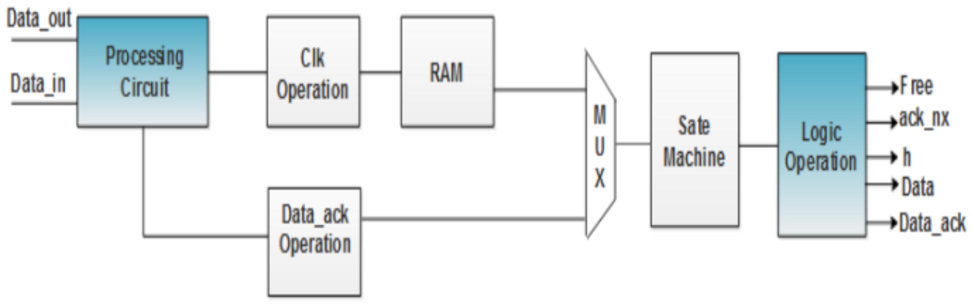


Fig 2. A diagram of the buffer architecture.

Figure 3 shows the input buffer architecture, which acts as the preprocessing circuit. This is because flits belong to the even packets, which are separated in another switch. As part of the process to restructure the structure, the buffer was included into each input switch port while simultaneously reducing the hierarchy of blocked flits. The newly implemented buffers functioned as First-in-First-out (FIFOs) queues in a circular fashion.

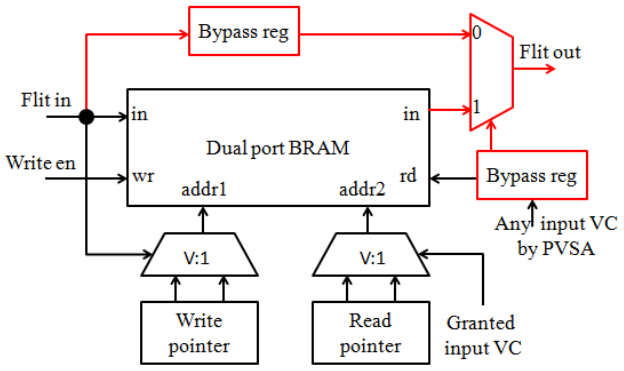


Fig 3. Input buffer architecture.

The O/P-port comprises of the following indications:

- The control signal –**T**x indicating availability of data .
- The output data is indicated by data out signal.
- Ack Tx- is a sort of control signal for confirming data function successfully.

The input/output port is made up of the following signals:

- Rx- control signal, which indicates that data is available
- Data-in - data to be received.
- Ack_Rx - control signal signifying successful data function.

3.2 Route computation using SSA

The network structure shown by the SSA model is made up of the same seven levels as the OSI reference model.Starting with an application layer, the layers are arranged hierarchically and include a presentation layer, session layer, transport layer, network layer, data layer, and physical layer (PL).Figure 4 depicts the SSA's construction in diagrammatic form. These layers define the requirements for communication between PEs. In most cases, the NoCs have implemented a subdivision of the lower layer, from transport layer down to physical layer and all these are described under for the context of the NoC. The PL offers definitions, both mechanical and electrical, for switching data with various entities at the bit level. In order to construct buffering systems, the address of routing resources and memory must be used to determine the width of the physical data bus. Routing and arbitration logic, as well as communication ports that are directly linked to other switches or cores, are typical components of a switch. The I/P and O/P channels make up the communication ports, and each one has a buffer for the short-term storing of any information that may pass across it (messages). In addition to routing logic, the switch has four bi-directional ports labeled A, B, C, D, and Local (L). The data flow via each and every port, which serves as a temporary storage area for information. Through the L port, communication may be established between the "Switch" and the local core of the device. The switch's other ports, including A, B, C, D, and L are connected to switches in the surrounding area.

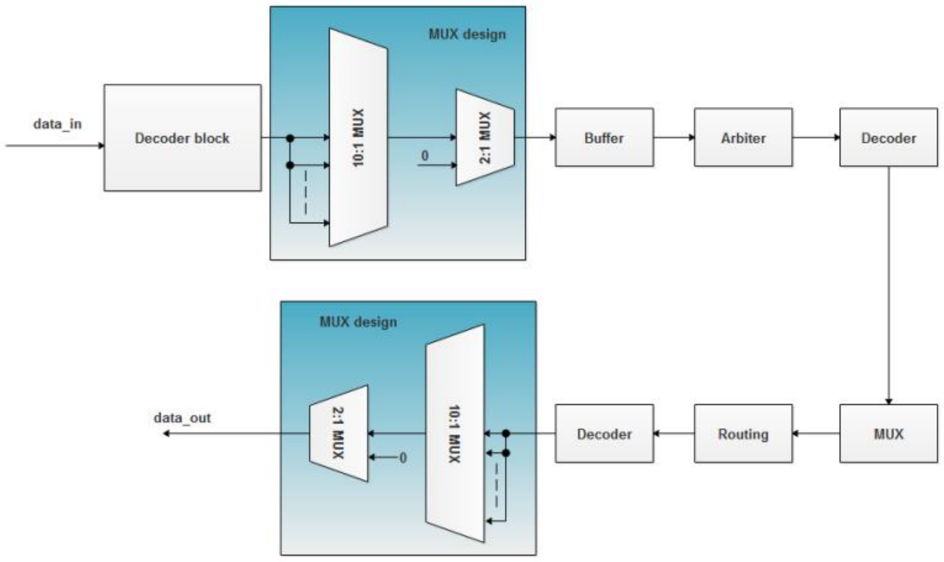


Fig 4. Route computation using SSA

Because of its low latency and relatively minimal amount of memory, the wormhole switch was chosen. The primary rationale for this decision is to simplify and reduce the overall cost of the switch by assigning one logical channel to each physical channel. The wormhole mode is what is used in order to cut the data packets up into flits. For the sake of this study, a flit size of 16 bits has been decided upon for the prototype and assessment procedure. The 1st and 2nd flits of the packet are the header information. This information performs the internal route by SSA switching components, the header flit, and a number of the flits that are solved are included inside the packet payload. In addition, every single switch request goes through the sorting process in the network. The address is provided in XY coordinates for the purpose of simplifying the routing algorithms that are intended to be used on the network. Here, X signifies the horizontal position, and y denotes the vertical location.

3.3 Parallel VSA

The non-speculative PVSA architecture is shown in Figure 6. Here, considering all requeststransmitted to the FTA arbiter's stage one , a single request is made per input port. Here, the FTA arbitrator also selects the output port depending on the requests that are available, which means that FTA chooses the VA or SA allocation, accordingly. In this case, the header flit request assigns both VA and SA, whereas the non-header flit request only assigns SA because the header flit has already assigned VA. For the duration of the packet's interval, the Virtual Allocation reservation is intended for the header flit. However, because switch requests are distributed on serial flit basis, each flit needs to take the SA allocation. In order to distinguish between requests for signals belonging to header and signals meant for non-header , the bits "0" and "1" are used to choose the requests for the VC and switch, respectively. The second step of the arbitrator receives the chosen output port at that point. The second step of the arbitrator chooses the various requests made via various input ports that are sent to the identical output port. The arbitration outcomes from the second one are retrieved to corresponding inputs. Access request are grantedonly after the completion of two stages of arbitration have been successfully. In the event in case of choosing switch "0,"type , the Switch Allocation of PVSA procedure gets completed. If type request for VC is chosen as"1," the PVSA allocation procedure for VA proceeds. With the help of the output channel arbitrator and amount of (V:1). The application monitoring logic has removed the input request for the output channel that does not have space available in the virtual machine. On the output channel, there can only be one grant signal sent. Ultimately the successful VC assignment is the end outcome. Since there is no available VA allocation on the crucial route, switch assignment takes precedence over VA allocation. Arbitration is required in the event that there are two allocator levels.

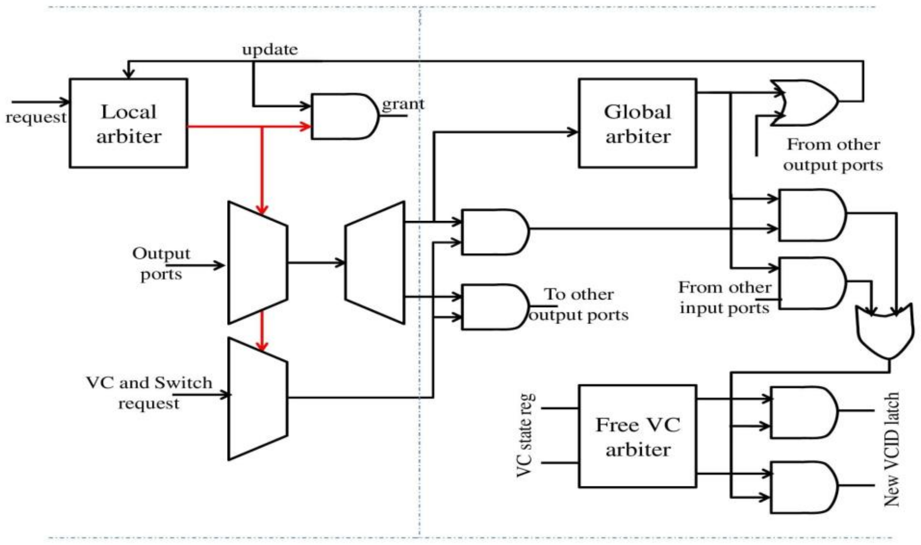


Fig 5. PVSA architecture

3.4LBRC module

The control logic generates a digital system that may be divided into two categories, such as routing and arbitration. Figure 4 presents an illustration of the arbiter architecture. When the switch gets a header flit, the arbitration receives a sequence of response to operate the program allowed, and an LBRC algorithm is considered for the operation of input-port data to the proper O/P port. The LBRC method creates a connection between the true switch address (xLyL) of the data packet and the target switch address (yTyL) of the packet, which is put in the header flit. When the xT<yT address of the packet is equal to the xL=yL address of the actual switch, the flits need to be offered in order to implement all of the

nodes. In the event that the criterion is not met, the xL address is contrasted with the xT address. When the condition of xLxT is met, the flits will be routed to the E-port. At addition, the header flit was horizontally oriented in the prior xL = xT position. In the event that the criterion is met, the yT address is weighed against the yL address. The flits will be sent to the S-port if the condition yLoyT is met, and they will be sent to the N-port whenever the condition yL>yT is met. In the event that the desired port is occupied, the header flit necessitates that the performance of routing data packets be obstructed. In order to set up connections in the switch schemes, this packet will need a Routing Request, abbreviated as RR.

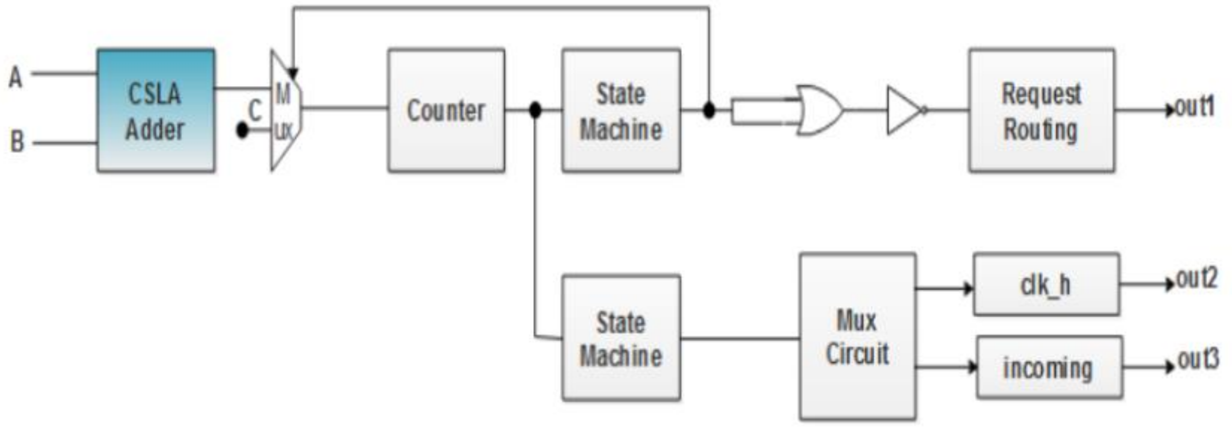


Fig 6. A diagram of the arbiter architecture.

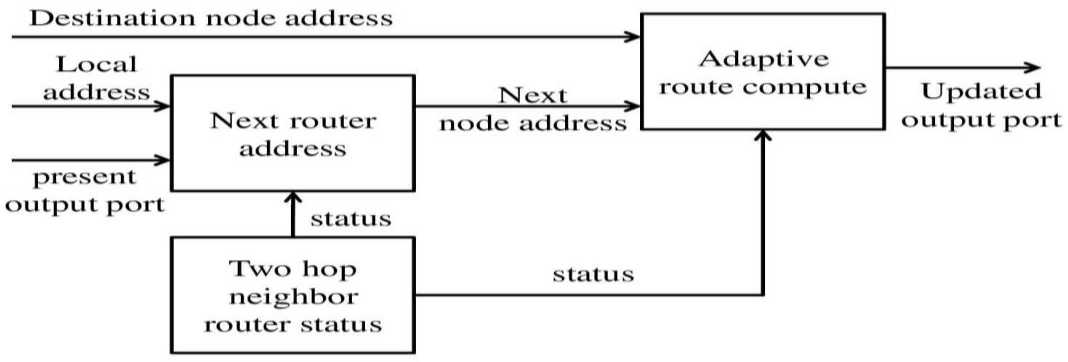


Fig 7. Proposed LBRCmodule.

Figure 7 depicts the proposed architectural arrangement of the LBRC module, which incorporates the likelihood of an inline router address for quick route computations to the output port, adaptive route computation, and a two-hop neighbor router. The local address and current output port information have an initial impact on the address of the subsequent router, and these factors forecast the addresses of the subsequent router nodes based on three pipeline bypass rules, respectively. The two-hop neighbor router then uses PVSA flits to generate the final status signal. There is no route congestion or overlapping situation if the status signal is genuine; otherwise, there is heavy traffic on that specific port. The status signal is once again stated to be a genuine enable with a decrease in traffic on that specific port alone. The adaptive route calculation module uses the status signal and the next node address to compute the possible routes depending on the destination node address and assignment of an optimized route to the successive output port, accordingly.

3.5Proposed FTA

The proposed FTA is described in the below section in which the faukt detection methodology is implemented.

The FTA implements the multiple bit detection and correction of errors.

The proposed Arbiter has a generator matrix which generates a coded output word. This coded output is result of matrix multiplication of generator matrix and input data bits. The resultant codeword is transmitted in the medium of space engineering. During codeword transmission in the channel there is a probability of the codeword getting corrupted because of errors and inclusion of channel noise. The FTA decoder also includes features such as Syndrome Detection, detecting and locating errors, error correction methods. The status of error , presence or absence of error in codeword are detected by Syndrome Detection. The error location and detection module in FTA is used to determine the quantity of error bits and where they are located inside the codeword. Error correction in the encoded result is the responsibility of the correction module.

B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15	B16	B17	B18	B19	B20	B21	B22
1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	1	1	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
1	1	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

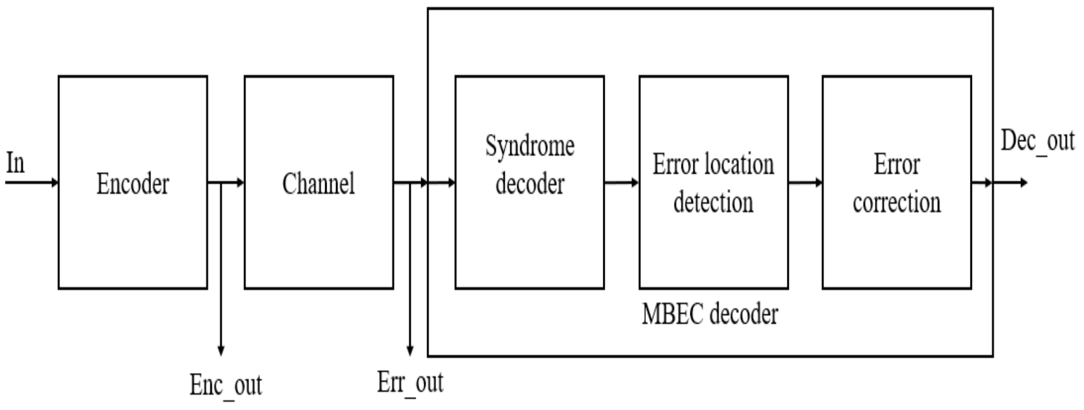


Fig 8. Proposed FTA flowchart

3.5.1FTA Encoding

Matrix multiplication of generation generator matrix and input data bits give the resultant FTA encoded code word .

Consider W is the word to be encoded , Generator Matrix is M and if N is the data input then W=M.N ----------(1)

The following table 1 illustrates Generator matrix values:

The generator matrix which is given below has identity matrix and parity bits. The size of parity bits is 16x7 placed in rows from columns C0-C6. The identity matrix size is 16x16 placed in columns C7-C22.

The data bit size is given as 16bits as :

N= [N1, N2, N3, N4, N5, N6, N7, N8, N9, N10, N11, N12, N13, N14, N15, N16] (3)

Generator matrix can be represented as :

M = [$P_{k\,x\,(n-k)}$. $I_{k\,.k}$] and Z = [P^T.$I_{(n-k)}$] ----------------------------(2)

As $I_{k\times k}$ is the identity matrix, P is the matrix with size k × (n − k), and P^T is the transpose of P.

The process of encoding is done with generator matrix ‘M’.

Encoding is got by check bit calculation of C1-C7 given below:

B1=($N1 \oplus N4 \oplus N6 \oplus N8 \oplus N9 \oplus N10 \oplus N14$) ------(4)

B2=$(N2 \oplus N4 \oplus N5 \oplus N7 \oplus N8 \oplus N11 \oplus N15)$ ------(5)

B3=$(N3 \oplus N7 \oplus N11 \oplus N13 \oplus N16 \oplus N10)$ ------(6)

B4 =$(N1 \oplus N4 \oplus N8 \oplus N10 \oplus N12 \oplus N13)$ -------(7)

B5 =$(N2 \oplus N5 \oplus N6 \oplus N7 \oplus N8 \oplus N13 \oplus N14)$ -------(9)

B6=$(N2 \oplus N6 \oplus N7 \oplus N11 \oplus N13 \oplus N16)$ -------(10)

B7 =$(N3 \oplus N6 \oplus N9 \oplus N11 \oplus N12 \oplus N13 \oplus N15 \oplus N16)$ --------(11)

Final expression of encoded word is shown as: W=B.N ---------------------(11)

$W =$ [B1,B2,B3,B4,B5,B6,B7,N1,N2,N3,N4,N5,N6,N7,N8,N9,N10,N11,N12,N13,N14,N15,N16]---------(12)

3.5.2 FTA Decoding

The process of Arbiter decoding includes error detection, correction and identification of error detection stages.

W *is the* output encoded word to which X=W+F is the representation which has noise added to it.

X=W+F --------- (13)

$XB = [B1 + B2 + B3 + B4 + B5 + B6 + B7] + [F1 + F2 + F3 + F4 + F5 + F6 + F7]$--------(14)

XN= $[N1, N2, N3, N4, N5, N6, N7, N8, N9, N10, N11, N12, N13, N14, N1$ $[F8, F9, F10, F11, F12, F13, F14, F15, F16, F17, F18, F19, F20, F2$ ---------------(15)

X= XB.XN ----------------(16)

If in the following equation the encoded word is written as W and Y is the syndrome where as F represents the error ,Z is the parity check matrix , then we have Y=X. Z transpose -------------------(17)

In table 2 the identity matrix is I with size 7x7 and parity bits are in rows from H1-H7 with 7x16 size all rows with H8-H23 columns. When syndrome Y=0 then zero errors are indicated whereas Y=1 indicated errors have occurred.

Parity check matrix is given in table 2 :

Z1	Z2	Z3	Z4	Z5	Z6	Z7	Z8	Z9	Z10	Z11	Z12	Z13	Z14	Z15	Z16	Z17	Z18	Z19	Z20	Z21	Z22	Z23
1	0	0	0	0	0	0	1	0	0	1	0	1	0	1	1	1	0	0	0	1	0	0
0	1	0	0	0	0	0	0	1	0	1	1	0	1	1	0	0	1	0	0	0	1	0
0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0	1	1	0	1	0	0	1
0	0	0	1	0	0	0	1	0	0	1	0	0	0	1	0	1	0	1	1	0	0	0
0	0	0	0	1	0	0	0	1	0	0	1	1	1	1	0	0	0	0	1	1	0	0
0	0	0	0	0	1	0	0	1	0	0	0	1	1	0	0	0	1	0	1	0	0	1
0	0	0	0	0	0	1	0	0	1	0	0	1	0	0	1	0	1	1	1	0	1	1
XB1	XB2	XB3	XB4	XB5	XB6	XB7	XN1	XN2	XN3	XN4	XN5	XN6	XN7	XN8	XN9	XN10	XN11	XN12	XN13	XN14	XN15	XN16

Simplification of syndrome bit values is done in the following format:

Y1=XDI ⊕ XNI ⊕ XN4 ⊕ XN6 ⊕ XN8 ⊕ XN9 ⊕ X10 ⊕ X14 ---------------(18)

Y2= XD2 ⊕ XN2 ⊕ XN4 ⊕ XN5 ⊕ XN9 ⊕ XN11 ⊕ XN15 ---------------(19)

Y3= $XD3 \oplus XN3 \oplus XN7 \oplus XN11 \oplus XN13 \oplus XN16 \oplus XN10$---------------(20)

Y4= $XD4 \oplus XN1 \oplus XN4 \oplus XN8 \oplus XN10 \oplus XN12 \oplus XN13$ ---------------(21)

Y5= $XD5 \oplus XN2 \oplus XN5 \oplus XN6 \oplus XN7 \oplus XN8 \oplus XN13 \oplus XN14$------------(22)

Y6= $XN6 \oplus XN2 \oplus XN6 \oplus XN7 \oplus XN11 \oplus XN13 \oplus XN16$-------------(23)

Y7= $XD7 \oplus XD3 \oplus XD6 \oplus XN11 \oplus XN12 \oplus XN13 \oplus XN15 \oplus XN16$ ---------(24)

The error values are identified based on the syndrome .

The following example gives the values of bit pattern as N=[1,1,0,1,1,1,0,0,1,1,0,0,1,1,1]

Check bits are calculated based equations (4)- (11) and the result is given as B= [0,0,1,0,1,0,1]

Finally the encoded resultant codeword is expressed as W=[0,0,1,0,1,0,1,1,1,0,1,1,1,0,0,1,1,0,0,1,1,1,1]

If 4 bits are to be assumed corrupted causes error in the positions of N2,N3,N4,N5 .

Then the value X =[0,0,1,0,1,0,1,1,0,0,1,1,0,0,1,1,1].

The XOR combinations in S-matrix columns are performed by error locations, and the XOR outputs must match the values of syndrome for each combination.

The above process is continued for syndrome matching continuously.

The below Table 3 gives the location of error identification:

Table 3. Error location identification.

Z8	Z9	Z10	Z11	XOR (Z8,Z9,Z10,Z11)	Y
0	0	1	0	1	1
1	0	1	1	1	1
0	1	0	0	1	1
0	0	1	0	1	1
1	0	0	1	1	0
1	0	0	0	0	1
0	1	0	0	1	1

Syndrome matching is done with XOR (Z8,Z9,Z10,Z11) and error locations are given as XN1,XN2,XN3,XN4.

For the vector X the equivalent Z vector positions include..Z8,Z9,Z10,Z11.The output id obtained as OUT which is a result of compliment of error corrected bits given as OUT =[0,0,1,0,1,0,1,1,1,0,1,1,1,0,0,1,1,1,1] .

4. Results and Discussions.

The implementation of proposed FTA-NoC is dealt in the below section and the metrics of a NoC are calculated . The metrics which carry significance for proposed arbiter are delay, power consumption and area of the chip.

4.1 Simulation Environment

The Xilinx® Versal® programmable NoC is an AXI-interconnecting network designed to mimic IP endpoint data exchange in integrated blocks such as processor elements (PE), programmable logic (PL), and others. This infrastructure is present throughout the entire device and consists of a dedicated high-speed channel with switching. Through the use of several horizontal and vertical channels along with a flexible set of architectural elements, the NoC may be rationally arranged to illustrate intricate topologies. Scalability was a primary design goal for the NoC. It is made up of many programmable hardware-implemented horizontal (HNoC) and vertical (VNoC) routes that are coupled together in various ways to fulfill the timing, speed, and logic consumption requirements of the design.

4.2 Performance evaluation

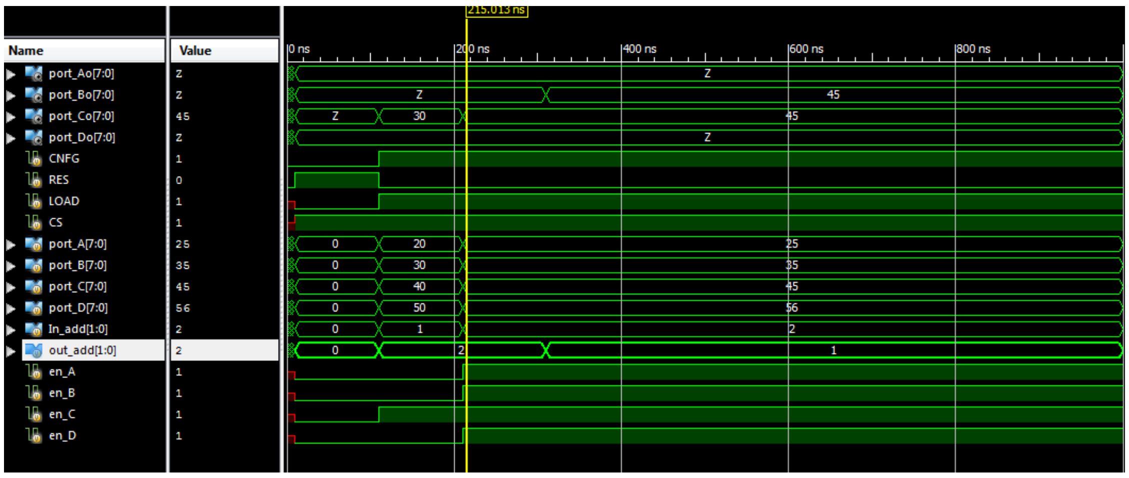
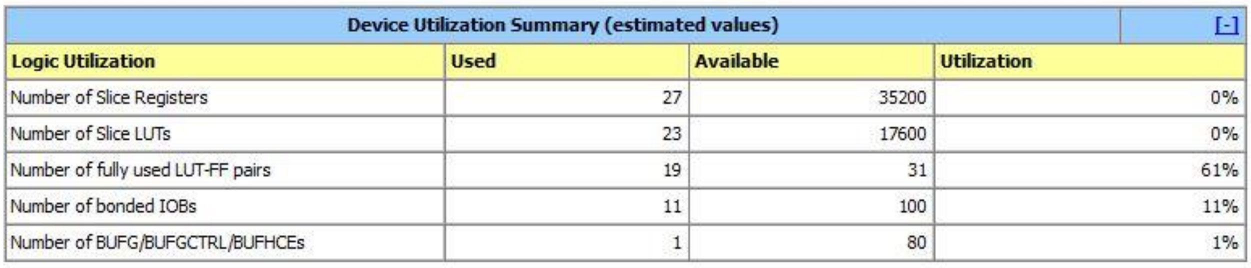


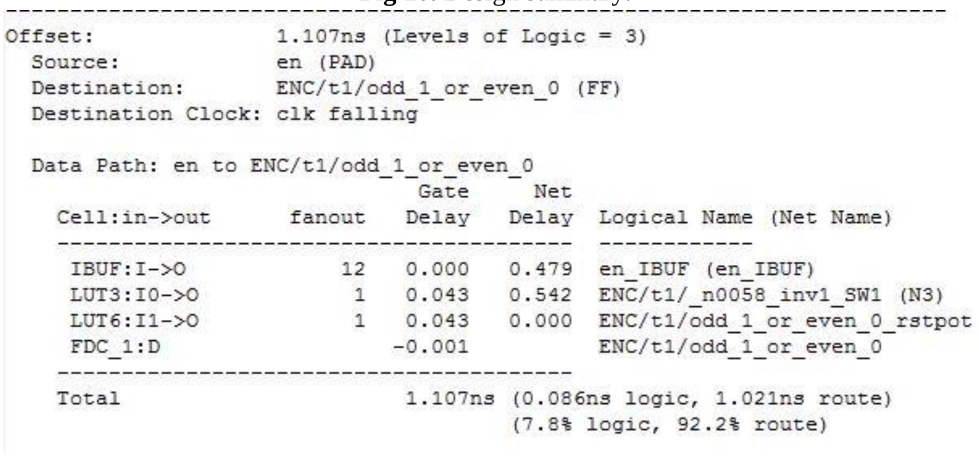
Fig 9. Simulation outcome.

The next part uses a range of performance indicators, such as slice registers, path delays, power consumption, data rate, Look-Up-Tables (LUTs), and Look-Up-Table-Flip-Flops (LUT-FFs) to compare the performance of the proposed technique with the state-of-the-art NoC methods. Figure 9 shows the simulation outcome of FTA-NoC, which shows the inputs and outputs. Figure 10presents the design summary of propose FTA-NoC. Figure 11presents the proposed FTA-NoCtime summary.



Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	27	35200	0%
Number of Slice LUTs	23	17600	0%
Number of fully used LUT-FF pairs	19	31	61%
Number of bonded IOBs	11	100	11%
Number of BUFG/BUFGCTRL/BUFHCEs	1	80	1%

Fig 10. Design summary.



```
-------------------------------------------------------------------------
Offset:              1.107ns (Levels of Logic = 3)
  Source:            en (PAD)
  Destination:       ENC/t1/odd_1_or_even_0 (FF)
  Destination Clock: clk falling

  Data Path: en to ENC/t1/odd_1_or_even_0
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IBUF:I->O            12   0.000   0.479  en_IBUF (en_IBUF)
     LUT3:I0->O            1   0.043   0.542  ENC/t1/_n0058_inv1_SW1 (N3)
     LUT6:I1->O            1   0.043   0.000  ENC/t1/odd_1_or_even_0_rstpot
     FDC_1:D                  -0.001          ENC/t1/odd_1_or_even_0
    ----------------------------------------
    Total                      1.107ns (0.086ns logic, 1.021ns route)
                                       (7.8% logic, 92.2% route)
```

Fig 11. Time Summary.

Table 4. Performance comparison of NoC approaches.

Method	Slice registers	LUTs	LUT-FF	Delay(nS)	Power(W)
Stanford-NoC [15]	89	96	28	3.92	1.93
DoS-NoC [17]	76	89	19	2.28	1.84
Conetntion-NoC [27]	72	74	23	1.82	1.29
Proposed FTA-NoC	**24**	**43**	**8**	**0.793**	**0.065**

The performance of the suggested FTA-NoC is contrasted in Table 4 with that of traditional methods like Stanford-NoC [15], DoS-NoC [17], and Contention-NoC [27]. In this case, using traditional methods to choose the high-speed routing channel results in failure and higher hardware resource consumption. Additionally, because of parallelism, the suggested approach's data rate also increased.

5. Conclusion

The primary emphasis of this paper is placed on the development of the FTA-NoC architecture using FIFO-Buffer, crossbar switching, route control, and arbiter modules. At first, the data that is created by the various devices is saved into FIFO-Buffer logic, which allocates the data depending on the IP addresses of the devices. After then, the route controller module will take control of the various routers in crossbar switching. These routers will be controlled by priority-based scheduling. The arbiter, which is chosen based on the request levels, then transmits the data from the source to the destination. The simulation results showed that the recommended FTA-NoC architecture performed better in terms of area, latency, and power when compared to other state-of-the-art NoC designs. Additionally, the suggested increased data rates by 11.3% as compared to traditional NoC methods. For better performance, this work can be expanded using NoC architectures based on optimization.

References

[1] Mehmood, Farrukh, et al. "An efficient and cost-effective application mapping for network-on-chip using Andean condor algorithm." *Journal of Network and Computer Applications* 200 (2022): 103319.

[2] Parepalli, Ramanamma, and Mohan Kumar Naik. "Design alternatives of Network-on-Chip (NoC) Router microarchitecture for future Communication System." *2022 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI)*. IEEE, 2022.

[3] Chen, Yuan-Ho, et al. "A VLSI Chip for the Abnormal Heart Beat Detection Using Convolutional Neural Network." *Sensors* 22.3 (2022): 796.

[4] Manzoor, Misbah, and Roohie Naaz Mir. "PAAD (Partially adaptive and deterministic routing): A Deadlock Free Congestion Aware Hybrid Routing for 2D Mesh Network-on-chips." *Microprocessors and Microsystems* (2022): 104551.

[5] Sharma, Ashish, et al. "Pre-Silicon NBTI Delay-Aware Modeling of Network-on-Chip Router Microarchitecture." *Microprocessors and Microsystems* 91 (2022): 104526.

[6] Gogoi, Ankur, et al. "Application driven routing for mesh based network-on-chip architectures." *Integration* 84 (2022): 26-36.

[7] Yazdanpanah, Fahimeh, and Raheel Afsharmazayejani. "A systematic analysis of power saving techniques for wireless network-on-chip architectures." *Journal of Systems Architecture* 126 (2022): 102485.

[8] Jadhav, Nathrao B., and Bharat S. Chaudhari. "Efficient Non-Blocking Optical Router for 3D Optical Network-on-Chip." *Optik* (2022): 169563.

[9] Biswas, Arnab Kumar. "Using Pattern of On-off Routers and Links and Router Delays to Protect Network-on-Chip Intellectual Property." *ACM Transactions on Computer Systems (TOCS)* (2022).

[10] Al-Azzwai, Waleed K., Aqeel A. Al-Hilali, and Laith F. Jumma. "Design and implementation 4x4 Network on Chip (NoC) using FPGA." *Periodicals of Engineering and Natural Sciences (PEN)* 10.3 (2022): 341-349.

[11] Seetharaman, Gopalakrishnan, and Debadatta Pati. "Design and Area Performance Energy Consumption Comparison of Secured Network-on-Chip with PTP and Bus Interconnections." *Journal of The Institution of Engineers (India): Series B* (2022): 1-13.

[12] Xia, Yuhao, et al. "Strict non-blocking four-port optical router for mesh photonic network-on-chip." *Journal of Semiconductors* 43.9 (2022): 092301-1.

[13] Velangi, Radha, and S. S. Kerur. "Hardware Implementation and Comparison of OE Routing Algorithm with Extended XY Routing Algorithm for 2D Mesh on Network on Chip." *Micro-Electronics and Telecommunication Engineering*. Springer, Singapore, 2022. 159-171.

[14] Kashi, Somayeh, et al. "A multi-application approach for synthesizing custom network-on-chips." *The Journal of Supercomputing* (2022): 1-23.

[15] Florida, L. Mary, S. Brilly Sangeetha, and K. Krishna Prasad. "OPTIMISED META-HEURISTIC QUEUING MODEL IN VLSI PHYSICAL DESIGN."

[16] Amin, Waqar, Fawad Hussain, and Sheraz Anjum. "iHPSA: An improved bio-inspired hybrid optimization algorithm for task mapping in Network on Chip." *Microprocessors and Microsystems* 90 (2022): 104493.

[17] Fan, Weibei, et al. "Communication and performance evaluation of 3-ary n-cubes onto

network-on-chips." *Science China Information Sciences* 65.7 (2022): 1-3.

[18] Gupta, Ruchika, et al. "Securing On-chip Interconnect against Delay Trojan using Dynamic Adaptive Caging." *Proceedings of the Great Lakes Symposium on VLSI 2022*. 2022.

[19] Kaleem, Muhammad, and Ismail Fauzi Bin Isnin. "Interval Based Transaction Record Keeping Mechanism for Adaptive 3D Network-on-Chip Routing."

[20] F. Imani, Mohammadreza, Sergi Abadal, and Philipp Del Hougne. "Metasurface-Programmable Wireless Network-On-Chip." *Advanced Science* (2022): 2201458.

[21] Thakkar, Ishan G., et al. "Hardware Security in Emerging Photonic Network-on-Chip Architectures." *Emerging Computing: From Devices to Systems*. Springer, Singapore, 2023. 291-313.

[22] Bhamidipati, Padmaja, and Avinash Karanth. "HREN: A Hybrid Reliable and Energy-Efficient Network-on-Chip Architecture." IEEE Transactions on Emerging Topics in Computing 10.2 (2022): 537-548.

[23] Kunthara, Rose George, et al. "DAReS: Deflection Aware Rerouting between Subnetworks in Bufferless On-Chip Networks." *Proceedings of the Great Lakes Symposium on VLSI 2022*. 2022.

[24] Firuzan, Arash, Mehdi Modarressi, and Midia Reshadi. "Reconfigurable network-on-chip based Convolutional Neural Network accelerator." *Journal of Systems Architecture* (2022): 102567.

[25] Patil, Trupti, et al. "A Minimal Buffer Router with Level Encoded Dual Rail-Based Design of Network-on-Chip Architecture." *Wireless Communications and Mobile Computing* 2022 (2022).

[26] Khan, Kamil, Sudeep Pasricha, and Ryan Gary Kim. "RACE: A Reinforcement Learning Framework for Improved Adaptive Control of NoC Channel Buffers." *Proceedings of the Great Lakes Symposium on VLSI 2022*. 2022.

[27] Salehnamadi, M. O. H. A. M. M. A. D. R. E. Z. A. "A Novel 3D Mesh-Based NoC Architecture for Performance Improvement." *Majlesi Journal of Electrical Engineering* 16.2 (2022).

[28] Bhaskar, Adusumilli Vijaya. "A Detailed Power Analysis of Network-on-Chip." *2022 IEEE Delhi Section Conference (DELCON)*. IEEE, 2022.

[29] Bhaskar, Adusumilli Vijaya. "Estimation of Power Consumption in a Network-on-Chip Router." *2022 IEEE Delhi Section Conference (DELCON)*. IEEE, 2022.

[30] Singh, Sangeeta, J. V. R. Ravindra, and B. Rajendra Naik. "Design and implementation of network-on-chip router using multi-priority based iterative round-robin matching with slip." *Transactions on Emerging Telecommunications Technologies* (2022): e4514.