ISSN: 0975-8585

Research Journal of Pharmaceutical, Biological and Chemical Sciences

A Genetic Algorithm Based Error Detection Mechanism in Dynamic Network on Chip.

Venkateswarlu B, and Vijayakumar V*.

Department of Electronics and Communication Engineering, Sathyabama University, Chennai-600119, India

ABSTRACT

A new error detection mechanism for Genetic algorithm in dynamic Network on Chip has to be utilize for find parent node data for router arrangement and it is also describe the shortest path and fault analysis in NOC to transmit the data efficiently with the help of genetic operators such as crossover, mutation and fitness function. The simulation of the proposed router has been verified through VHDL codes using XILINX VIVADO. The damaged router in Network on Chip can be rejected the path for data transmission through faultless path. Comparatively, power consumption will be reduced to 20% with adaptive router architecture.

Keywords: VHDL, XILINX VIVADO, Network on Chip, System on Chip

Corresponding author

INTRODUCTION

Scheduling is one of the biggest issues in the managing and planning of manufacturing process. Especially, the difficulty of searching the optimal schedule is mainly depends on the process constraints, shop environment and the performance indicator. Qing-dao-er-ji and Wang et al. [1] discussed about the job shop scheduling problem (JSSP). The descriptions of JSSP as follows: there are m machines utilized for processing n jobs. A single machine can process maximum one operation at the instance and once an operation begins processing on a selected machine it should be complete processing without any interruption. A schedule contains a complete set of operations which can be denoted as n x m. These set of operations can be processed on various machines, in a given order. The main difficulty is to find a schedule of short time to complete all jobs.

Lawer et al. [2] reported that the JSSP is mainly faced combinatorial optimization problem. The prominent methods have been suggested by many authors for solving small instances [3-7]. In the present situation 15x15 dimensions of problems are still deliberated to be beyond the reach of present exact methods. Different methods have been utilized in the last few decades to solve the JSSP, such as simulated annealing [8], Tabu search [9], Evolutionary algorithm techniques, genetic algorithms for instance [10], particle swarm optimization (PSO) [6], Greedy randomized adsptive search procedure (GRASP) [11] and shifting bottleneck procedure (SBP) [12]. The most popular type evolutionary algorithm is genetic algorithms.

In electronic, more importance is on how to reduce the chip cost, area and how to develop a system in short interval of time. Integrating a complete product on a single chip is known as system on chip (SOC). Many works have been performed in this field of SOC. Owing to regular increase in the count of transistors on a single chip has revealed new fields leads to enhance more complex products on SOC. Due to the increasing of complexity, alternate method must be introduce which makes it simple to integrate new component in SOC. Network-on-chip (NOC) is a prominent platform to reduce the complexity of SOC, in addition it is mainly concentrated on research challenge of scalability of interconnection network [13]. In the present work, a variable hardware router code is designed with the help of Verilog and same to implement for the NOC router and enhance the power consumption.

ALGORITHM DESCRIPTION

The basic of genetic algorithm, start with a large “population” of randomly generated “attempted solution” to a problem. Then repeatedly do the following are evaluate each of the attempted solutions, probabilistically keep a subset of the best solution, and use these solution to generate a new population. Routing on NOC is quite similar to routing on any network.

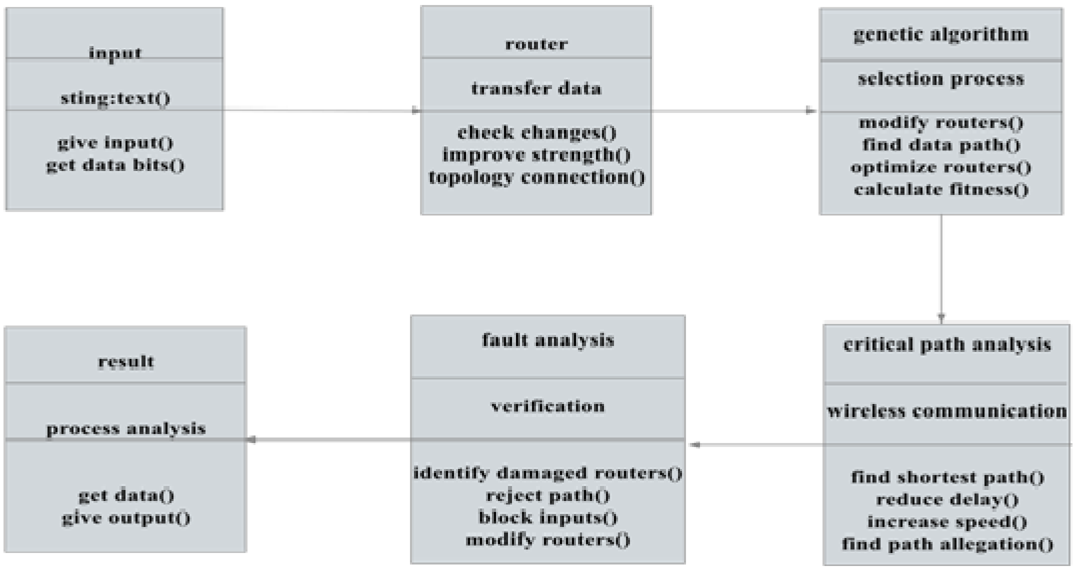
A routing algorithm determines how the data is routed from sender to receiver. Routing algorithms are divided into two groups, oblivious and adaptive algorithms. Oblivious algorithms are also divided into two subgroups: deterministic and stochastic algorithms. Oblivious algorithms route packets without any information about traffic amounts and conditions of the network, deterministic algorithms route packets always along a same route and stochastic routing is based on randomness.

Finally, quit when you have a satisfactory solution. The algorithm for fitness function,

```
1. Choose initial population
2. Evaluate the fitness of each individual in the population
3. while <terminating condition> do
4. Select best-ranking individuals to reproduce
5. Breed new generation through crossover and mutation
6. (Genetic operations) and give birth to offspring
7. Evaluate the individual finesses of the offspring
8. Replace worst ranked part of population with offspring
9. end while
```

Genetic Algorithms are adaptive heuristic search algorithm based on the evolutionary ideas of natural selection and genetics [14]. As such they represent an intelligent exploitation of a random search used to solve optimization problems. Although randomised, GAs are by no means random, instead they exploit historical information to direct the search into the region of better performance within the search space. It is better than conventional AI in that it is more robust. Unlike older AI systems, they do not break easily even if the inputs changed slightly, or in the presence of reasonable noise. Also, in searching a large state-space, multi-modal state-space, or n-dimensional surface, a genetic algorithm may offer significant benefits over more typical search of optimization techniques.

In the computer science field of artificial intelligence, a Genetic Algorithm is a search heuristic that mimics the process of natural selection. This heuristic is also sometimes called a meta heuristic which is routinely used to generate useful solutions to optimization and search problems. Genetic algorithms belong to the larger class of Evolutionary Algorithms, which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover.



CLASS DIAGRAM

Fig 1. Class Diagram to Transmit Data

Figure 1 shows that, representation of class diagram to transmit data. The input blocks is used to get data bits. Data are transfer with help of router from one position to other position, it connected with mesh topology and also check the changes in the router then improve its strength.

Router process is selected by genetic algorithm to find the data, optimize router and to calculate the fitness value with the help of mutation results. The critical path analysis are used to find shortest path thereby it reduce delay and increase its speed and also it find path allegation.

The fault analysis process is used to verify faulty block if it identify any damaged router means then reject path or block the input. So the faulty block will be removed data will transmitted through fault free block.

SOFTWARE DESCRIPTION

The ISE® Design Suite is the Xilinx® design environment, which allows you to take your design from design entry to Xilinx device programming. With specific editions for logic, embedded processor the ISE Design Suite provides an environment tailored to meet your specific design needs.

Xilinx ISE (Integrated Software Environment) is a software tool produced by Xilinx for synthesis and analysis of HDL designs, enabling the developer to synthesize ("compile") their designs, perform timing analysis, examine RTL diagrams, simulate a design's reaction to different stimuli, and configure the target device with the programmer.

ISE Design Suite: Logic Edition

The ISE Design Suite: Logic Edition allows you to go from design entry, through implementation and verification, to device programming from within the unified environment of the ISE Project Navigator or from the command line. This edition includes exclusive tools and technologies to help achieve optimal design results, including the following:

- **PlanAhead™ software -** allows you to do advanced FPGA floorplanning. The PlanAhead software includes PinAhead, an environment designed to help you to import or create the initial I/O Port list, group the related ports into separate folders called "Interfaces" and assign them to package pins. PinAhead supports fully automatic pin placement or semi-automated interactive modes to allow controlled I/O Port assignment.
- **CORE Generator™ software -** provides an extensive library of Xilinx LogiCORE™ IP from basic elements to complex system level IP cores.
- **SmartGuide™ technology -** allows you to use results from a previous implementation to guide the next implementation for faster incremental implementation.
- **ChipScope™ Pro tool -** assists with in-circuit verification.

ISE Design Suite: Embedded Edition

The ISE Design Suite: Embedded Edition includes all the tools and capabilities of the Logic Edition with the added capabilities of the Embedded Development Kit (EDK). This pre-configured kit is an integrated software solution for designing embedded processing systems, which includes the Platform Studio tool suite as well as all the documentation and IP required for designing Xilinx Platform FPGAs with embedded PowerPC® hard processor cores and MicroBlaze™ soft processor cores. This edition provides an integrated development environment of embedded processing tools, processor cores, IP, software libraries, and design generators, including the following:

- **Xilinx Platform Studio (XPS) -** provides an integrated environment for creating software and hardware specification flows for embedded processor systems based on MicroBlaze and PowerPC processors. It also provides an editor and a project management interface to create and edit source code. XPS allows you to customize tool flow configuration options and provides a graphical system editor for connection of processors, peripherals, and buses.
- **Hardware Platform Generation Tool (PlatGen) -** customizes and generates the embedded processor system through the use of hardware netlist Hardware Description Language (HDL) files. By default, PlatGen synthesizes each processor IP core instance found in your embedded hardware design using Xilinx Synthesis Technology (XST). PlatGen also generates the system-level HDL file that interconnects all the IP cores, which can then be synthesized as part of the overall design flow.
- **Base System Builder Wizard (BSB) -** allows you to quickly create a working embedded design, using any features of a supported development board or using basic functionality common to most embedded systems. After you create a basic system, you can then customize it using the XPS and ISE software tools.
- **Simulation Model Generation Tool (SimGen) -** generates simulation models of your embedded hardware system, based either on your original, behavioral embedded hardware design or your finished, timing-accurate device implementation. SimGen can also incorporate your embedded software to run on the model.
- **Create and Import Peripheral Wizard -** helps you create your own peripherals and import them into EDK-compliant repositories or XPS projects. The wizard can create an HDL template for your custom logic and provides an interface to one of the supported IBM CoreConnect or Xilinx FSL buses.
- **Software Development Kit (SDK) -** provides a C/C++ development environment for software application projects. SDK is based on the Eclipse open source standard. SDK provides tool software project management and access to the GNU toolchain for code compilation and debug. It is also available for purchase as a standalone product.

- **GNU Software Development Tools -** - assist with compiling and debugging. Embedded software applications written in C, C++, or assembly are compiled using the GNU compiler tool chain. The GNU tool chain is part of the SDK and customized to target the PowerPC and MicroBlaze processors. For detailed information about the GNU tools, including compilers and debuggers, see the "GNU Compiler Tools" and "GNU Debugger (GDB)" chapters in the *Embedded System Tools Reference Manual*.
- **Xilinx Microprocessor Debugger (XMD) and GNU Software Debugging Tools -** allows you to debug your embedded application; either on the host development system, using an instruction set simulator, or on a board that has a Xilinx device loaded with your hardware bit stream. For more information on XMD, see the "Xilinx Microprocessor Debugger (XMD
- **Library Generation Tool (LibGen) -** - configures libraries, device drivers, file systems, and interrupt handlers for the embedded processor system to create a software platform.
- **Bitstream Initializer (BitInit) -** - updates a device configuration bitstream to initialize the on-chip instruction memory with the software executable. For more information, see the "Bitstream Initializer (BitInit)" chapter of the *Embedded System Tools Reference Manual* and the “Initializing Software Overview” topic in the XPS Help.

ISE Design Suite: DSP Edition

The ISE Design Suite: DSP Edition includes all the tools and capabilities of the Logic Edition with the added capabilities of the System Generator for DSP and the AccelDSP™ Synthesis Tool. This edition provides an integrated environment with tools to help you achieve optimal design results for your DSP design in less time, including the following:

- **System Generator for DSP -** allows you to define and verify complete DSP systems using industry-standard tools from The MathWorks. When using System Generator, previous experience with Xilinx devices or RTL design methodologies is *not* required. Designs are captured in the DSP-friendly Simulink® modeling environment using a Xilinx-specific blockset. All of the downstream synthesis and implementation steps are automatically performed to generate a device programming file.
- **AccelDSP Synthesis Tool -** allows you to transform a MATLAB® floating-point design into a hardware module that can be implemented in a Xilinx device. The AccelDSP Synthesis Tool features an easy-to-use graphical interface that controls an integrated environment with other design tools such as MATLAB tools, ISE software, and other industry- standard HDL simulators and logic synthesizers. AccelDSP Synthesis provides the following capabilities:

  - Reads and analyzes a MATLAB floating-point design.
  - Automatically creates an equivalent MATLAB fixed-point design.
  - Invokes a MATLAB simulation to verify the fixed-point design.
  - Provides you with the power to quickly explore design trade-offs of algorithms that are optimized for the target device architectures.
  - Creates a synthesizable RTL HDL model and a test bench to ensure bit-true, cycle-accurate design verification.
  - Provides scripts that invoke and control down-stream tools such as HDL simulators, RTL logic synthesizers, and ISE implementation tools.

ISE Design Suite: System Edition

The ISE Design Suite: System Edition includes all of the tools and capabilities of the Logic Edition, Embedded Edition, and DSP Edition. You can use the ISim standalone flow to simulate your design without setting up a project in ISE® Project Navigator. In this flow, you:

- Prepare the simulation project by manually creating an ISim project file to create a simulation executable using the fuse command.
- Start the ISim Graphical User Interface (GUI) by running the simulation executable generated by the fuse command.

ISSN: 0975-8585

TESTING OF PRODUCT

System testing

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

Types of tests

Unit test

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centred on the following items:

Valid Input	:	identified classes of valid input must be accepted.
Invalid Input	:	invalid input must be rejected.
Functions	:	identified functions must be exercised.
Output	:	identified classes of application outputs.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirement. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document.

Integration testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.

The entry screen, messages and responses must not be delayed

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

TOOLS REQUIRED

XILINX Version: 14.2

Families: Xilinx Power Estimator (XPE) are Artix-7, Kintex-7, Virtex-7

ISE:	Placement and Routing (P&R) runtime -13 hrs
	Memory usage-16 GB
	Wire length and congestion-high
VIVADO:	Placement and Routing (P&R) runtime -5 hrs
	Memory usage-9 GB
	Wire length and congestion-reduced

RESULTS AND DISCUSSION

SIMULATION RESULTS

Roulette wheel selection

The input of roulette wheel are clock, reset, spin and the output are digit0, digit1. Fitness proportionate selection is also known as roulette wheel selection. The solution with a higher fitness will be less likely to be eliminated.

After the optimization and technology targeting phase of synthesis process, it can display in schematic representation this shows a design in terms of logic element optimized to the target Xilinx. The selected source file is displayed in schematic form in the workspace.

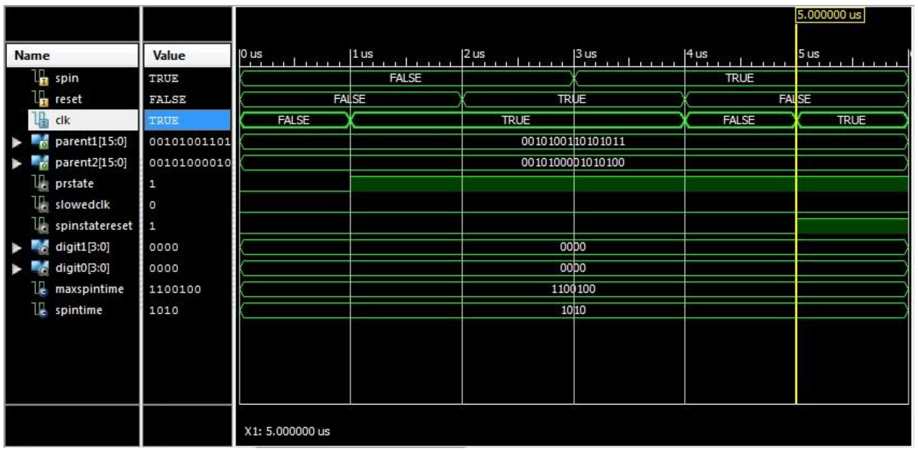


Fig 2. Waveform of Roulette Wheel Selection for Router Arrangement

From Figure 2, output waveform of roulette wheel selection performs router arrangement. The number of times the roulette wheel is spin is equal to size of the population. The individual consist of 16 bit chromosomes and are being used to optimize a simple mathematical function.

Crossover

The input of crossover are parent1, parent2, clk, reset and the output are child1, child2. So here the child node data are get from parent node. It having 16 bit data which is process of making more than one parent node to produce a child node data.

After the optimization and technology targeting phase of synthesis process, it can display in schematic representation this shows a design in terms of logic element optimized to the target Xilinx. The selected source file is displayed in schematic form in the workspace.

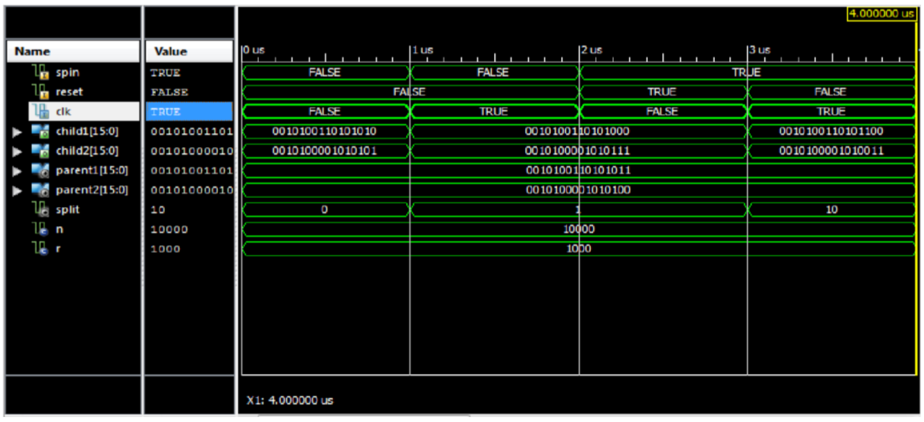


Fig 3. Waveform of Crossover to find Child Node Data

From fig 4.2, it shows the output waveform of crossover to find child node from parent node. By using two parent node data, find a child node.

Parent 1+ parent 2= child node
0010100110101011+0010100001010100=001010000101011

Mutation

The encoder will converts information from one format or code to another for the purposes of standardization, speed, secrecy. Mutation alters one or more gene values in a chromosome from its initial state. This probability should be set low. If it is set too high, the search will turn into a primitive random search.

After the optimization and technology targeting phase of synthesis process, it can display in schematic representation this shows a design in terms of logic element optimized to the target Xilinx. The selected source file is displayed in schematic form in the workspace.

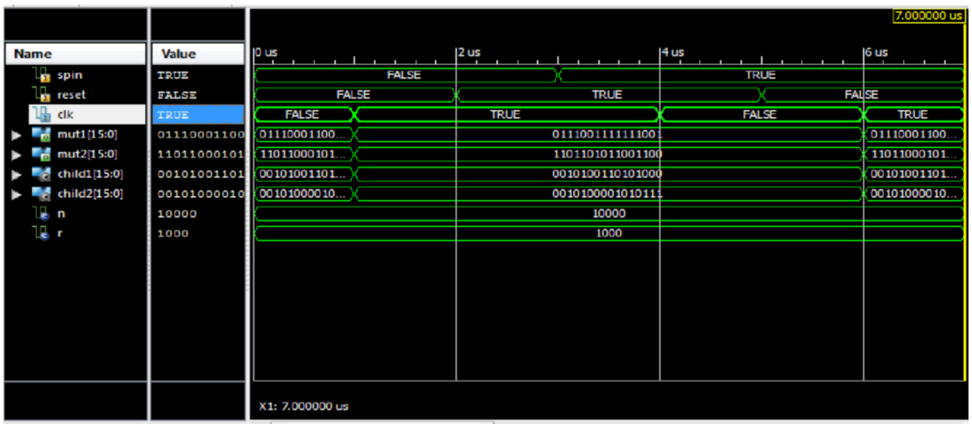


Fig 4. Waveform of Mutation to find Modify Bit Position

From Figure 4, it shows the output waveform of mutation to find modify bit position with help of crossover and encoder. Mutation should allow the algorithm to avoid local minima by preventing of chromosome from becoming too similar to each other. To find modify bit by,

Child node 1+ child node 2= modify bit
00101001101000+001010000101011=1101101011001100

Fitness function

The input of fitness are clk, reset, spin and the output of fitness are fit1, fit2. Here spin look like clock, which is a bit by bit analysis. It produce 4 bit data value when clock and enable become high otherwise it does not produce a output. After the optimization and technology targeting phase of synthesis process, it can display in schematic representation this shows a design in terms of logic element optimized to the target Xilinx. The selected source file is displayed in schematic form in the workspace.

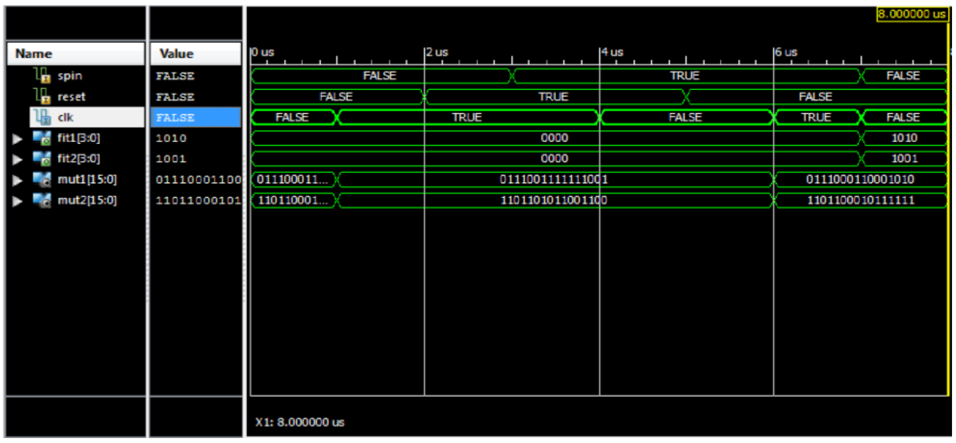


Fig 5. Waveform of Fitness Function to find Shortest Path

From Figure 5, it shows the output waveform of fitness function to find shortest path analysis with help of crossover, encoder, mutation. The fitness function must not only correlate closely with the particular value, it must be computed quickly. When clock and enable become high and reset as 0 it produce a resulted data as output.

Router arrangement

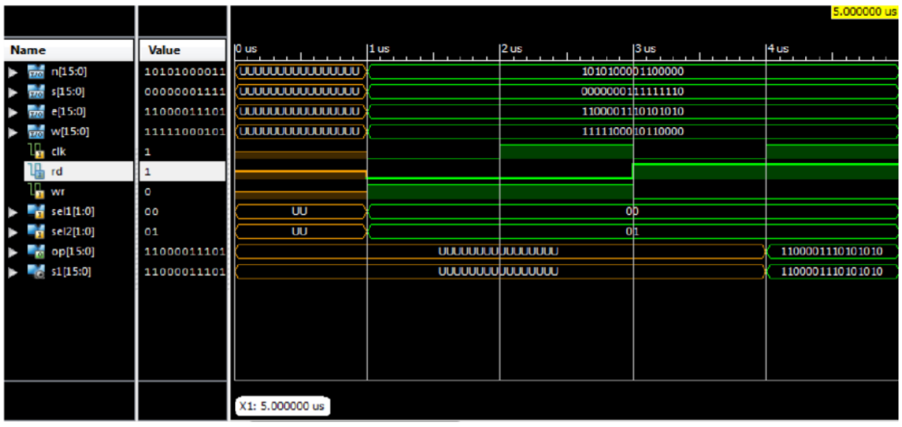


Fig 6. Waveform of Router Arrangement to find Parent Node Data

From Figure 6, it shows the output waveform of router arrangement to find parent node data. Here the four direction of router become input n, s, w, e and two selection lines with clock, rd, wr. Then the output will depend upon the selected direction. Hence the output node from e direction is: Parent1=> 1100001110101010

Stuck at fault

The input for stuck block is clock, spin, reset and the output is fault enable. The output is produced with help of clock and enable value.

After the optimization and technology targeting phase of synthesis process, it can display in schematic representation this shows a design in terms of logic element optimized to the target Xilinx. The selected source file is displayed in schematic form in the workspace.

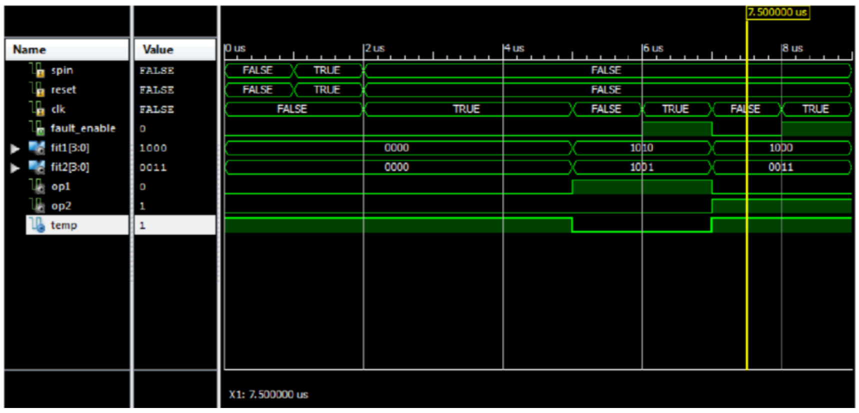


Fig 7. Waveform of Stuck at Fault to find Shortest Data Path

From Figure 7, it shows the output waveform of stuck at fault to find shortest data path with help of fitness function. Here the input are spin, reset, clock, enable, fitness1, fitness2 and the output are op1, op2, temp. When clock and enable become high and reset, spin are low then it produce data which is shortest path data.

ISSN: 0975-8585

Table 1: Specific NoC Experimental Results

NoC	Number of Switches	Average Latency (Clock Cycles)	Average Power Consumption(mW)
Static NoC	6	5.96	278.021
NoC 1	4	3.9	211.789
NoC 2	4	4	204.308
NoC 3	4	4.07	216.519

Fault analysis

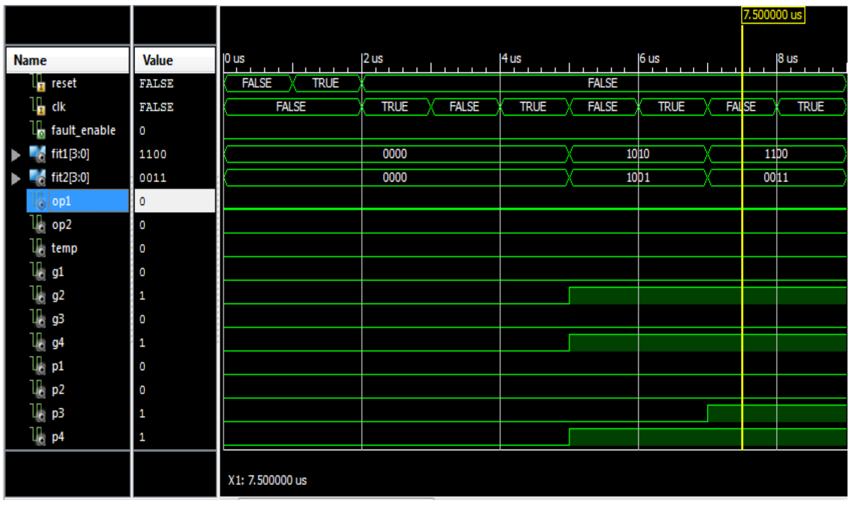


Fig 8. Waveform of Fault Analysis with the help of Fitness Function

The specific experimental results of NOC are tabulated in Table 1. From Figure 8, the fault analysis process is used to identify the damaged router in network on chip with the help of genetic operator which is fitness function. So the output of fitness function can be produced by crossover and mutation output. The internal signal are taken as previous output value, here the previous fitness results can be taken for internal input value for fault analysis.

Table 2: Parameter Comparisons Results

Metrics	**Existing System**	**Proposed System**
Slice	1436	53
Slice-Flip Flop	1866	140
LUT's	1888	127
BIO's	86	11
Delay(Ns)	4.277	2.322
Frequency(MHz)	322.781	640.820
Latency(Ns)	5.9	5.241
Power (mW)	5.3338	4

From the above Table 2, it shows the power comparison results. These comparisons and simulation for regular route and fitness function to find shortest path analysis and fault analysis output waveform can be simulated with the help of Xilinx.

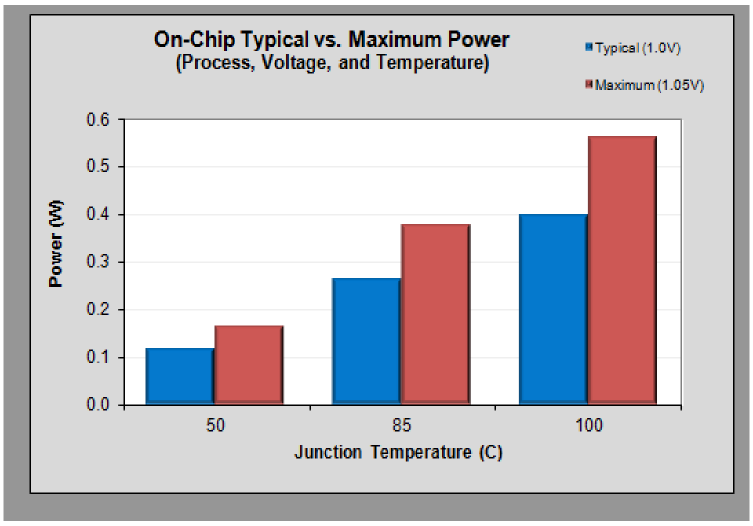


Fig 9. Power Analysis Output

Two processes take places in analyzing NOC router based on GA optimization algorithm: creating a router architecture and implementation of GA optimization algorithm. This results in identification of routing path from source to destination and creating a node for further transmission. Optimum output for search heuristics, minimal node selection, finding effective data path, improved data bit strength level with minimum power consumptions is obtained as shown in Figure 9.

CONCLUSION

The simulation of the proposed router is verified through VHDL codes using XILINX VIVADO. A new error detection mechanism for dynamic NOCs is Genetic algorithm, which is used to find parent node data for router arrangement and it also analyzes the shortest path and fault analysis in the NOC to transmit the data efficiently with the help of genetic operators (crossover, mutation, fitness function). Damaged router in NOC chip can be rejected the path for data transmission through faultless path. The power value obtained from results is 0.04mW. So that power consumption gets reduced 20% compared to adaptive router architecture and the results shows the graph for On-chip typical Vs maximum power.

REFERENCES

[1] Qing-doa-er-ji R, Wang Y, Computers & Operations Research, 2012; 39(10): 2291-2299.
[2] Adams J, Balas E, Zawack Z, Management Science, 1988; 34: 391-401.
[3] Giffler B, Thompson G.L, Operations Research, 1960; 8(4): 487-503.
[4] Carlier J, Pinson E, Management Science, 1989; 35(29): 164-176.
[5] Carlier J, Pinson E, Annals of Operations Research, 1990; 26: 269-287.
[6] Brucker P, Jurisch B, Sievers B, Discrete Applied Mathematics, 1994; 49:105-127.
[7] Williamson DP, Hall LA, Hoogeveen JA, Hurkens CAJ, Lenstra J K, Sevastjanov S V, Shmoys DB, Operations Research, 1997; 45(2): 288-294.
[8] Lourenço HR, European Journal of Operational Research, 1995; 83:347-364.
[9] Zhang C Y, Li P, Guan Z, Computers & Operations Research, 2008; 35: 282-294.
[10] Aarts EHL, Van Laarhoven PJM, Lenstra JK, Ulder NLJ, ORSA Journal on Computing, 1994; 6:118-125.
[11] Aiex RM, Binato S, Resende MGC, Parallel Computing, 2003; 29: 393 - 430.
[12] Pardalos P, Shylo O, Vazacopoulos A, Computational Optimization and Applications, 2010; 47(1):1-16.
[13] Dijkstra E W, Numerische Mathematlk 1959; l:269 – 271.
[14] Tabassum M, Mathew K, International Journal of Digital Information and Wireless Communications (IJDIWC) 2014; 4(1): 124-142.