ABSTRACT

Network-on-Chip (NoC) has been proposed as a solution for addressing the design challenges of future nanoscale architecture. This paper contribute in the enhancement of NOC architecture, a switch/router size 4x4 of 2-D mesh topology is targeted, the scheduler under consideration and Input module with efficient buffer to store the data packet while waiting for the scheduling are optimized at architecture level. The physical implementation of scheduler and the input module replacing D-Flip Flop based array by SRAM is carried out for area and power optimization.

For further optimization of area and power, this paper also presents a low power design methodology – Gate Level Power Optimization to optimize dynamics & leakage power of the design and It has been observed that approximately 28% of Area and 68% of Power are optimized when compared with generic architecture of 2-D Mesh NoC D-Flip Flop based router.

KEYWORDS: Network-on-Chip, Router, SRAM, RTL Gate Level Power Optimization

INTRODUCTION

Ever-increasing requirements on electronic systems are one of the key factors for evolution of the integrated circuit technology. [1] [2]. Network-on-Chip (NoC) is a general purpose on-chip communication concept that offers high throughput, which is the basic requirement to deal with complexity of modern systems, as shown in Fig 1. All links in NoC can be simultaneously used for data transmission, which provides a high level of parallelism and makes it attractive to replace the typical communication architectures like shared buses or point-to-point dedicated wires.

![Figure 1: NoC Architecture](image)

Apart from throughput, NoC platform is scalable and has the potential to keep up with the pace of technology advances [3]. But all these enhancements come at the expense of area and power. In the RAW multiprocessor system, interconnection network consumes 36% of the total chip power [4]. A typical NoC system consists of processing elements (PEs), network interfaces (NIs), routers and channels. The router further contains scheduler, switch and buffers. Buffers consume the 64% of the total node (router + link) leakage power for all process technologies, which makes it the largest power consumer in any NoC system. [5]. Moreover, buffers are dominant for dynamic energy consumption [6].
NoC ARCHITECTURE

Network-on-Chip has been proposed on various topologies [7] - [10]. A simple NoC architecture consists of three components: the routing nodes, the links, and network interfaces (or network adapters in some literature), as shown in Fig. 2. Routers direct data over several links (hops). Topology defines their logical lay-out (connections) whereas floorplan defines the physical layout. The function of a network interface (adapter) is to decouple computation (the resources) from communication (the network). Routing decides the path taken from source to the destination whereas switching and flow control policies define the timing of transfers. Task scheduling refers to the order in which the application tasks are executed and task mapping defines which processing element (PE) executes certain task. IP mapping, on the other hand, defines how PEs and other resources are connected to the NoC [11].

![Figure 2: NoC Overview](image)

The three critical challenges for NoC are: power, area, latency and CAD compatibility. [12]. The key research areas in network-on-chip design [13] [14], are as:

- Communication infrastructure: topology and link optimization, buffer sizing, floorplanning, clock domains, power.
- Communication paradigm: routing, switching, flow control, quality-of-service, network interfaces
- Application mapping: task mapping/scheduling and IP component mapping.

All of these challenges result in area, power, and performance tradeoffs [13]. Area and power can be estimated from hardware requirements. Performance is generally estimated using analytical model.

This paper proposes the area and power efficient design of the router as it is the most redundant component which is equal to the no. of PEs on one kind of NoC, as shown in Fig. 2.

PROBLEM STATEMENT

The implementation of network-on-chip presents certain challenges. Two of the most critical design metrics for networks-on-chip are area requirements and power consumption. Due to the fact that die area per wafer of silicon is limited, the NoC implementation should be carried out using an approach that minimizes area requirement. Also due to likelihood of most SoCs being implemented in battery powered devices, power consumption of the NoC should also be as low as possible. Usually, reduction in area results in a saving in power requirements due to the fact a smaller area is achieved using fewer components on-chip. Fewer components on-chip will consume less power compared to architecture requiring more components on-chip. Standard-cell based ASIC design methodology is the fastest approach available in the design of complex digital circuits.

DESIGN AND IMPLEMENTATION OF PROPOSED TASK

Given an existing implementation of a routing node for on-chip networks, it is the goal of this work to present a modified implementation of the routing node to minimize the area requirements and as a result lower the power
requirement. The routing node consists of four basic components: the input blocks, the scheduler, the output blocks, and the crossbar switch as shown in Fig 3.

![Figure 3: Router Components](image)

The primary function of the input block is to store incoming packets before they can be routed to their respective output ports. Hence, the majority of the area of the input blocks is used by memory elements. The existing design employs DFF (D flip-flop) elements for memory storage. The modified input block will be based on SRAM memory cells. SRAM memory cells provide the fastest and most compact means of on-chip storage. Always in high performance CPU architectures, memory is implemented as on-chip SRAM to achieve the best possible performance.

The function of the scheduler is to arbitrate between conflicting requests for access to the crossbar switch shared medium. The existing scheduler architecture is based on a symmetric implementation of round-robin-like algorithm requiring one set of grant arbiters and one set of accept arbiters to perform arbitration. The modified design uses the concept of folding to reduce the area of the scheduler by removing one set of arbiters and using the remaining set of arbiters to perform both grant and accept arbitration in a time multiplexed fashion.

The design of the modified routing node is implemented using standard cell based VLSI flow with provision for custom IP core inclusion. The Synopsis tool chain is used to implement the design from RTL coding to synthesis and place and route. Design verification is carried out using hierarchical functional simulation at each level of the design flow. Also, static timing analysis is used to verify timing closure in the final design layout.

Thus, the aim of this work is to present a modified architecture of the routing node to achieve higher area and power efficiency using changes at the RTL architecture level and use of custom IP to boost the performance of standard-cell based ASIC design.

THE PROPOSED ROUTER ARCHITECTURE

The routing node configuration shown in Fig.4 is 4x4. It is based on a 2D mesh NoC topology where each routing node is connected to four other routing nodes. The NOC infrastructure includes components responsible for packetization, transmission, and de-packetization of data. These components, respectively, are the NI, the VC router, and the links. These components are repeated for every grid element in NOC.

However, it must be noted that serial packet-based communication will still remain an optimum solution as compared to a bus-based system in terms of the power consumption and will reduce the cost of system design in the longer run due to the potential for reuse.
AREA OF FOCUS

The design of the proposed router has been carried out as follows:

**Proposed Switching Technique**

Switching techniques can be classified based on network characteristics. Circuit switched networks reserve a physical path before transmitting the data packets, while packet switched networks transmit the packets without reserving the entire path. Packet switched networks can further be classified as Wormhole, Store and Forward (S&F), and Virtual Cut through Switching (VCT) networks as shown in Fig. 5. Here with, we proposed S&F switching forwards a packet only when there is enough space available in the receiving buffer to hold the entire packet. Thus, there is no need for dividing a packet into flits. This reduces the overhead, as it does not require circuits such as a flit builder, a flit decoder, a flit stripper and a flit sequencer. Store and forward is the easiest policy in terms of implementation complexity. So this implementation is based on store and forward switching.

**Proposed Flow Control Mechanism**

Flow control determines how network resources, such as channel bandwidth, buffer capacity, and control state, are allocated to a packet traversing the network. The flow control may be buffered or buffer less as shown in Fig. 6. The Buffer less Flow Control has more latency and fewer throughputs than the Buffered Flow Control. The Buffered Flow Control can be classified further as:
In Handshaking Signal Based Flow Control, a VALID signal is sent whenever a sender transmits any flit. The receiver acknowledges by asserting a VALID signal after consuming the data flit and used in SoCIN NOC implementation [17]. In Credit Based Flow Control, an upstream node keeps count of data transfers, and thus the available free slots are termed as credits. Once the transmitted data packet is either consumed or further transmitted, a credit is sent back and used [15] [16]. To minimize the chances of dropped packets at the receiving end, the credit based flow control mechanism has been incorporated wherein only those output IP blocks take part in the scheduling that has some credit. In addition to this, every input block maintains packet array and the linked list array to maintain the proper flow so as to avoid the out of order delivery.

Proposed Buffer Implementation in the Design of Router

A higher buffer capacity and a larger number of virtual channels in the buffer will reduce network contention, thereby reducing latency. However, buffers are area hungry, and their use needs to be carefully directed [18] [19] therefore proposed a simple implementation of a buffer architecture for NOC buffers using 0.18 µm technology to estimate the cost and area of buffers needed for NOC. Also proposed the trade-off between buffer size and channel bandwidth to secure constant latency and concluded that increasing the channel bandwidth is preferable to reducing the latency in NOC.

The input block consists of six major components: the packet array, the linked list array, the destination head array, the destination tail array, the free-list FIFO, and a shift register. Four of these six components are conventional memory elements. In a standard cell based design, memory elements are realized using D flip flops in the standard SYNOPSYS Library. If we consider a NAND gate implementation of a D flip flop with no RESET or SET inputs it requires 28 MOS transistors to realize one D flip flop [20]. A more area efficient implementation of memory is through the use of SRAM cells. Each SRAM cell is implemented using 6 transistors. Therefore, memory realization using SRAM is more efficient compared to D flip flops. However, standard cell based approach to ASIC design does provide SRAM standard cells because of the many possible configurations of width and depth. SRAM design is carried out using full custom approach to ASIC design as shown in Fig. 7. By combining standard cell based and full custom ASIC design, D flip flops can be replaced by SRAM, improving the area efficiency of the input block. Full custom design of SRAMs has been carried out by MILKYWAY of SYNOPSYS, while physical implementation of the input module with SRAMs has been carried out by IC Compiler of SYNOPSYS.

![Figure 7: Input Module with SRAM Based Arrays](image)

Proposed Scheduler in the Design

The scheduler was modified using a folding approach due to the regular structure and placement of the arbiters. The modified scheduler is as shown in Fig. 8. Each arbiter in the modified scheduler now has to generate both grant and accept signals in a time multiplexed fashion. The arbiter is modified to hold both grant and accept pointers for successive time slots.
The proposed scheduler belongs to a Router in 2D Mesh NOC design. So here the value of N is 4.

**Figure 8: Modified Scheduler**

**INTRODUCTION TO GATE LEVEL POWER OPTIMIZATION**

Power Compiler optimizes designs for power. During an optimization session, Power Compiler performs additional steps to optimize the design for dynamic and leakage power. Gate Level Power Optimization Contains:-

- Leakage Power Optimization
- Dynamic Power Optimization

The speed of the transistor continues to improve. The most common technique used to achieve the high performance is to reduce the geometry of the transistor as well as the voltage to operate it. To maintain the speed and noise margin of the smaller transistor, the threshold voltage needs to be lowered too. Since the threshold voltage has exponential impact on the transistor leakage power, low threshold voltage transistors have high leakage power. Minimizing the leakage power is one of the major challenges to be resolved, especially in lower geometries. In any design, there are critical and non-critical timing paths. Using a lower speed cell on non-critical path does not affect the performance of a design. A slower cell allows higher threshold voltage, which reduces leakage power dramatically. Optimizing the high speed and low speed cells on different timing paths leads to a balanced design with high performance and low leakage power.

**Input and Output of Power Optimization**

The inputs for gate-level power optimization are:

- RTL or gate-level netlist and floor plan (optional)

*This netlist is not power optimized.*

- Power options

*Power options enable the power optimization.*

- Libraries

**Figure 9: I/O Flow for Power Optimization**
Power Compiler selects different library cells to rebuild the netlist with the optimized power. Multivoltage threshold libraries are highly recommended for leakage optimization.

- Switching activity

This is required for dynamic and total power optimization, and is used for high accuracy in leakage optimization.

The output of gate-level power optimization is a new gate-level netlist that has optimized power. The optimization is implemented with the compile or compile_ultra commands.

Power Optimization in Synthesis Flow

Figure shows the steps involved in power optimization in the synthesis flow.

Table 1: Comparative Results of Proposed Router with Existing Router Design

<table>
<thead>
<tr>
<th>Network</th>
<th>Topology</th>
<th>Flit Size in Bits</th>
<th>Ports</th>
<th>Buf Size in Flits</th>
<th>Tech in nm</th>
<th>L in Clk</th>
<th>A in Sq.mm</th>
<th>F in MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teraflops</td>
<td>Mesh</td>
<td>32</td>
<td>4</td>
<td>16</td>
<td>65</td>
<td>5</td>
<td>0.34</td>
<td>4270</td>
</tr>
<tr>
<td>Xpipes</td>
<td>Custom</td>
<td>32</td>
<td>4</td>
<td>--</td>
<td>100</td>
<td>7</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>Dally</td>
<td>Torus</td>
<td>256</td>
<td>5</td>
<td>4</td>
<td>100</td>
<td>3</td>
<td>--</td>
<td>200 - 2000</td>
</tr>
<tr>
<td>HIBI</td>
<td>Bus</td>
<td>32</td>
<td>2</td>
<td>2.8</td>
<td>130</td>
<td>4</td>
<td>0.03 - 0.05</td>
<td>435</td>
</tr>
<tr>
<td>Octagon</td>
<td>Ext. Ring</td>
<td>32</td>
<td>4</td>
<td>2.8</td>
<td>130</td>
<td>4</td>
<td>0.04 - 0.09</td>
<td>435</td>
</tr>
<tr>
<td>SPIN</td>
<td>Fat. T</td>
<td>16</td>
<td>8</td>
<td>8</td>
<td>130</td>
<td></td>
<td>0.24</td>
<td>200</td>
</tr>
<tr>
<td>Aethereal</td>
<td>Mesh</td>
<td>96</td>
<td>5</td>
<td>8</td>
<td>120</td>
<td>0.26</td>
<td>500</td>
<td></td>
</tr>
<tr>
<td>ANoC</td>
<td>Mesh</td>
<td>32</td>
<td>5</td>
<td>1</td>
<td>130</td>
<td>0.25</td>
<td>500</td>
<td></td>
</tr>
<tr>
<td>Mango</td>
<td>Mesh</td>
<td>32</td>
<td>5</td>
<td>2.8</td>
<td>130</td>
<td>10</td>
<td>0.19</td>
<td>795</td>
</tr>
<tr>
<td>Hermes</td>
<td>Mesh</td>
<td>32</td>
<td>5</td>
<td>1</td>
<td>130</td>
<td>10</td>
<td>0.05 - 0.11</td>
<td>435</td>
</tr>
<tr>
<td>SoCBus</td>
<td>Custom</td>
<td>16</td>
<td>3</td>
<td>1</td>
<td>180</td>
<td></td>
<td></td>
<td>180</td>
</tr>
<tr>
<td>ASoC</td>
<td>Mesh</td>
<td>32</td>
<td>4</td>
<td>2</td>
<td>180</td>
<td>0.04 - 0.08</td>
<td>400</td>
<td></td>
</tr>
<tr>
<td>Avg.</td>
<td></td>
<td>50.1</td>
<td>4.8</td>
<td>6.4</td>
<td>170</td>
<td>5.2</td>
<td>0.14 - 0.22</td>
<td>328-596</td>
</tr>
<tr>
<td>Present Work</td>
<td>Mesh</td>
<td>32</td>
<td>4</td>
<td>8</td>
<td>90</td>
<td>4</td>
<td>0.15</td>
<td>500</td>
</tr>
</tbody>
</table>
Legends used in above Table:-, Buf.-Buffer, Tech-Technology, L-Latency, A-Area, F-Frequency, Ext-Extended, R-Ring, T-Tree Cus- Custom, Avg-Average, Pre-Present.

EXPERIMENTAL RESULTS 1: PHYSICAL IMPLEMENTATION

4x4 Routing Nodes D Flip Flop (DFF) _Physical Implementation

Table 2: Comparisons Showing Result of Area, Power of 4x4 Routing Node at Post Synthesis
# Clock Period: 16ns.

<table>
<thead>
<tr>
<th>Post Synthesis</th>
<th>DFF based</th>
<th>SRAM based</th>
<th>% Reduction in SRAM based Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area</td>
<td>202704 sq um</td>
<td>150314 sq um</td>
<td>25.85%</td>
</tr>
<tr>
<td>Power</td>
<td>10.7 mW</td>
<td>3.9 mW</td>
<td>63.55%</td>
</tr>
</tbody>
</table>

Table 3: Comparisons Showing Results of Area, Power of 4x4 Routing Node at Post Layout
# Clock Period: 16ns.

<table>
<thead>
<tr>
<th>Post Layout</th>
<th>DFF based</th>
<th>SRAM based</th>
<th>% Reduction in SRAM based Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area</td>
<td>205298 sq um</td>
<td>150756 sq um</td>
<td>26.567%</td>
</tr>
<tr>
<td>Power</td>
<td>14.29 mW</td>
<td>4.98 mW</td>
<td>65.15%</td>
</tr>
</tbody>
</table>
Dynamic and Leakage Power Optimization Using Low Power Design Methodology – Gate Level Power Optimization for Implementation of 2-D Mesh Network-on-Chip Router

Figure 11: Comparison of DFF & SRAM Based Designs at Post Synthesis

Figure 12: Comparison of DFF & SRAM Based Designs at Post Layout

INSERTION OF GATE LEVEL POWER OPTIMIZATION

Choosing the Leakage Power Calculation Model

To choose the model that the tool should use to calculate the leakage power of the design, use the `set_leakage_power_model` command.

The syntax of this command is as follows:

```
set_leakage_power_model [-type leakage | channel_width] \\
[-mvth_weights leakage | channel_width] [-reset]
```

The following script uses the default usage model for multivoltage threshold leakage optimization.

```
# Specify all multivoltage threshold libraries in one place
set target_library “hvt.db nvt.db lvt.db”
set link_library “* $target_library”

# Read the design
read_verilog rtl toplevel1(Interconnect).v
link

# Enable leakage power optimization
set_leakage_optimization true
compile_ultra
report_power
```
Running Dynamic Power Optimization

# The set_dynamic_optimization command sets the dynamic power option and enables dynamic power optimization.

The syntax of this command is

set_dynamic_optimization [true | false]

set_dynamic_optimization true

The default setting has a well-balanced runtime and quality of result.

# setup general environment #

set target_library "lib.db"

set link_library "+ $target_library"

read_verilog design Toplevel1(Interconnect).v

link compile_ultra

# dynamic power optimization constraint

set_dynamic_optimization true

read_saif -input my.saif -instance_ Toplevel1 (Interconnect) tb/top_inst

compile Ultra -incremental

report_power

EXPERIMENTAL RESULTS 2: GATE LEVEL POWER OPTIMIZATION

Power Analysis at Gate Level

Comparisons Chart Showing Result of Area & Power

# Clock Period: 16 ns

<table>
<thead>
<tr>
<th></th>
<th>DFF based</th>
<th>Gate-Level Power Optimization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area</td>
<td>202704 Sq um</td>
<td>146254 Sq um</td>
</tr>
<tr>
<td>Total Power</td>
<td>10.7 mW</td>
<td>3.50 mW</td>
</tr>
</tbody>
</table>

Total Power = (Dynamic + Leakage Power)
CONCLUSIONS

Network-on-Chip (NoC) has been proposed as a solution for addressing the design challenges of future nanoscale architecture. We contribute in the enhancement of NOC architecture, a switch/router size 4x4 of 2-D mesh topology is targeted, the scheduler under consideration and Input module with efficient buffer to store the data packet while waiting for the scheduling are optimized at architecture level. The physical implementation of scheduler and the input module replacing D-Flip Flop based array by SRAM is carried out for area and power optimization.

For further optimization of area and power, this paper also presents a low power design methodology – Gate Level Power Optimization to optimize dynamics & leakage power of the design and it has been observed that approximately 28% of Area and 68% of Power are optimized when compared with generic architecture of 2-D Mesh NoC D-Flip Flop based router.

FUTURE SCOPE

The existing design has been synthesized using MUX as a crossbar switch more Area efficient crossbar like tristate and sense amps can further make the design efficient. We foresee that Area & Power of existing design can be further optimized by Low Power Design Methodology Likes:

1. Multi VDD - Since dynamic power is proportional to VDD$^2$ lowering VDD on selected blocks helps reduce power significantly. Unfortunately, lowering the voltage also increases the delay of the gates in the design.

2. Multi Threshold Logic- As geometries have shrunk to 130nm, 90nm, and below, using libraries with multiple $V_T$ has become a more efficient way of reducing leakage voltage.

REFERENCES


