CHAPTER 4

FPGA IMPLEMENTATION OF ROUTER AND
NETWORK COMPONENT

This chapter describes the FPGA implementation of the configurable DXbar router design. The DXbar of each block or component was modeled in Verilog HDL and was synthesized in cyclone V SoC Field Programmable Gate Array (FPGA) device. The synthesis results of DXbar implementation and its static & dynamic power dissipation are compared with generic router implementation. Functionality of each Modules are described in implementation perspective.

4.1 DXBar ROUTER DESIGN

The DXBar router contains primary & secondary crossbar, processing unit, routing unit, switch allocator, and mux and de-mux in input and output ports. The functional component for design the each unit was described as follow.

4.1.1 Crossbar

The crossbar switch of a router is the heart of the router data path. It switches the data from the input port to the output port doing the essence of the router function. The internal structure of a crossbar consists of an array of multiplexers. In this architecture it consists of five 5:1 multiplexers.
The five multiplexers have five 32 bit data as input. All the five inputs are connected to all the five multiplexers. The data to be forwarded to the output depends on the select lines. The select lines are generated by the arbiter depending on the request signals. There are five select lines of three bit width to select one of the five ports. So the output of each multiplexer depends on the select line of that multiplexer which was control by processing unit.

4.1.2 Processing Unit

Processing unit is the control unit of the router architecture. It steer the sequence of operation and generate the required control signal to each integrated cores. Main responsibility of Processing unit is to control the communication between different input ports to different output ports. It also resolves the contention problem between the inputs ports which desire the same output port by giving them priority. It selects the output port by sending signal to cross Bar. Locking and unlocking of path is also its
This confirms the reception of packet. When the CI receives the packet whose ACK_I is active, it sends back ACK signal to Core PE indicating that the transfer is complete. This process even though is quite useful for safe data transmission, keeps the CI waiting until acknowledge is received. This can be avoided by disabling the ACK Mode. When CI acts as a receiver/destination node, it reads in the incoming packet (noc_in) to an input register (Register), unpack the message and sends the data information (DAT_O) to the PE.

4.2.1 The slave Network Interface

The slave Network Adapter/Interface (NI) and CI are fundamentally the same but they differ in some aspects. The NI waits for the network, where the master network adapter waits on the IP core.

Figure 4.12

responsibility. It locks the path after passing the Head flit and unlocks the path after sending the End flit. The state diagram representation processing unit is shown in figure 4.2. It receive the three bit control signal to select the network topology. The control input determines the specific router ports that are used and the algorithm for deciding the path of packets through the router, as appropriate to the topology. The three-bit control input is intended to be constantly driven with the selected topology-dependent setting. The appropriate setting is assigned to the configurable DXBar routers by high-level integration software during system creation.

Figure 4.2 State diagram representation DXBar processing unit
4.1.3 Routing Unit

The routing unit implements hardware support for the routing algorithms for all supported topologies, and it also includes the receiver portion of the link-level flow control. Figure 4.3 shows the hardware architecture of FSM based Routing Unit. The control logic of the routing unit is a finite-state machine (FSM) for deterministic and distributed routing. It processes the packet header to compute an appropriate output channel and generates requests for that output channel accordingly. Moreover, the routing unit provides the handshaking response signals (ACK or NACK) to the sender in order to indicate packet acceptance status.

![Figure 4.3 State diagram representation DXBar Routing unit](image-url)

Figure 4.3 State diagram representation DXBar Routing unit
4.1.4 Memory Buffer

There are many different implementations possible for storing packets (flits) in the buffer. In this work, secondary cross bar input buffer has been implemented using memory buffer. The reason is that memory Buffer takes less hardware as compared to a set of registers. Although register based buffer are faster than memory based buffer but they are costly. Secondly memory buffers are easily scalable as compared to register based buffers. Figure 4.4 shows the buffer hardware architecture. It consist of 32 bit data width and FSM based control logic. In this architecture the packets in secondary cross bar transit are stored in a buffer. The FSM controller receives the packet from the primary port when traffic arise and stores them in buffer for manages the flow control between adjacent routers. The depth of input buffer used in this work is 8 and each location is 32 bits wide.
4.1.5 Input Port

Input Port performs the following functionality.

- To transfer the flits between two routers or resources by using handshaking protocol.
- To receive the flits and store them in input buffers
- To send the flits to the next blocks
- To send signal to the other routers or to the resource about the status of the input buffer (empty or full).
- To manage the input buffer

Figure 4.5 Input port

Figure 4.5 shows hardware architecture of input port. It has five inputs and three outputs. Inputs are clock, Rst, Put, Get, and Data_in while
outputs are RTR, Req, and Data_out. Data_in and Data_out both are 32 bits wide which are used for both sending and receiving data transferring.

### 4.1.6 Switch Allocator

The switch allocator is for controlling the arbitration of ports and to resolve the contention issues. It knows the current status of all the ports, which ports are free, which ports are communicating with each other and in which ports the data contention can occur.

![Switch Allocator Diagram](image)

**Figure 4.6 Switch allocator**

Packets of same priority and destined for the same port is scheduled by a round robin algorithm. The arbiter can release the output port which is connected to the crossbar once it finishes the data transmission in that particular port. Then the port will be assigned to the next awaiting port in the
queue. The arbiter generates an output signal of three bit which is given to the select line of the crossbar for selecting the corresponding port. It is these three bits that determines the data that comes in the output port of the router.

The internal structure of the arbiter was shown in figure 4.6. It consist of D flip flop, ring counter, five priority logic blocks and five input OR gates. In logic designs flip flops are used to create simple finite state machines. The ring counter is a counter where simple shift registers are connected in cascade to each other. The output of the last flip flop is connected as the input to the first flip flop. The data pattern will circulate as long as the clock pulses are applied. In the arbiter it is the ring counter that selects the incoming requests in round robin way and gives it to the priority logic block. The functionality of the priority logic block is similar to a priority encoder without output encoding.

Since each block has different order of inputs the priority of chosen signals varies with the chosen block. With a round-robin arbiter, the last request to be serviced will have the lowest priority in the next round of arbitration.

4.1.7 Synthesis Results

The configurable DXbar router hardware architecture was developed using IEEE standard Verilog HDL and was tested on Altera Cyclone V System-on-Chip FPGA. Its compilation report is listed in Table 4.1. The design consumes 2.8k logic element, 33.7k embedded memory bits and 4 embedded multiplier of a cyclone V FPGA. The end results for RTL schematic and chip planner has been shown in Figure 4.7 & 4.8.
The arbiter generates an output signal of three bits which is given to the select line of the crossbar for selecting the corresponding port. It is these three bits that determine the data that comes in the output port of the router.

The internal structure of the arbiter was shown in Figure 4.6. It consists of D flip flops, a ring counter, five priority logic blocks, and five input OR gates. In logic designs, flip flops are used to create simple finite state machines. The ring counter is a counter where simple shift registers are connected in cascade to each other. The output of the last flip flop is connected as the input to the first flip flop. The data pattern will circulate as long as the clock pulses are applied. In the arbiter, it is the ring counter that selects the incoming requests in a round-robin way and gives it to the priority logic block. The functionality of the priority logic block is similar to a priority encoder without output encoding.

Since each block has a different order of inputs, the priority of chosen signals varies with the chosen block. With a round-robin arbiter, the last request to be serviced will have the lowest priority in the next round of arbitration.

4.1.7 Synthesis Results

The configurable DXbar router hardware architecture was developed using IEEE standard Verilog HDL and was tested on Altera Cyclone V System-on-Chip FPGA. Its compilation report is listed in Table 4.1. The design consumes 2.8k logic elements, 33.7k embedded memory bits, and 4 embedded multipliers of a Cyclone V FPGA. The end results for RTL schematic and chip planner have been shown in Figure 4.7 & 4.8.

Figure 4.7 Chip planner

Figure 4.8 RTL view
Table 4.1 Resource utilization summary

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Total logic elements</td>
<td>2832</td>
</tr>
<tr>
<td>Combinational function</td>
<td>2700</td>
</tr>
<tr>
<td>Memory bits</td>
<td>33792</td>
</tr>
<tr>
<td>Total power dissipation</td>
<td>28.3mV</td>
</tr>
</tbody>
</table>

4.2 NETWORK INTERFACE

The network interface provide the communication interface between router and processing element. Its main function is to generate and process packets. Network Interface component implements a Core Network Interface (CNI) at the core side and a Slave network interface (SNI) at the slave side. Core Network Interface When the master PE wishes to make a request to the slave, the Core Network Interface wraps the request into a packet containing the necessary data and the route to the slave and sends the packet. Hence, CNI functions as a slave for the master PE.
4.1.5 Input Port

Input Port performs the following functionality:
- To transfer the flits between two routers or resources by using handshaking protocol.
- To receive the flits and store them in input buffers.
- To send the flits to the next blocks.
- To send signal to the other routers or to the resource about the status of the input buffer (empty or full).
- To manage the input buffer.

Figure 4.5 shows hardware architecture of input port. It has five inputs and three outputs. Inputs are clock, Rst, Put, Get, and Data_in while outputs are DAT_O and ACK_O. The noc_out Packet to Network is connected to the output of the module.

Figure 4.9 Network Interface Module

Figure 4.9 shows functional block diagram of Network interface. It is implemented as a Moore Machine which is triggered by the control signals generated from the PE and generates the packet. In the Store and forward switching, every CI consists of a Routing Table (RT). RT is basically a lookup table which contains minimal routing path between every source node to every possible destination node in the NoC. When the Master PE initiates a request/transfer, the CI FSM is triggered. In SF switching, CI reads in the destination address (ADR_I), looks up the RT and maps the address into routes to the intended destinations (host intelligent routing). This information is stored in the packet. The size and contents of the packet can also be parameterized by modifying the corresponding parameters in a single configuration file. The packet format is shown in Figure 4.10.
Each Flit consists of 2 bit packet ID, to determine the type of packet.

00 – Header

01 – Data

10 – Address

11 – Tail.

The data and address flits use three bits to determine the order or packet. Every packet has 2 bits of Wishbone control signals namely STB_I and CYC_I. Each data or address packet includes 16 bits of data or address information.

In Store and Forward switching, the packet header contains the entire routing information in the Route/Destphit. To reduce the length of packet, we use two bits to represent each direction; 00 for north, 01 for South; 10 for East and 11 for West. IP or end of route is represented by last two bits in the route in opposite direction. For example a packet in the direction {East, East, North, North} will have the route as {10,10,00,00,01}. The last two bits represent the end of route will be in opposite direction to the final route.
The tail bit contains four parity bits, one for each data and address flits. A parity bit is a bit that is added to ensure that the number of bits with the value one in a set of bits is even or odd. Parity bits are used as the simplest form of error detecting code. Even parity is used where, the parity bit is set to 1 if the number of ones in the corresponding data bits is odd. Even parity is a special case of a cyclic redundancy check (CRC), where the 1-bit CRC is generated by the polynomial $x+1$.

The State Machine used in CI module is shown in Figure 4.11. Each state represents the generation of single FLIT. CI has 2 modes of operation, ACK mode and NON-ACK mode. In ACK mode, the CI after sending packet to the network waits for slave to send back a packet with acknowledge signal.

![Figure 4.11 Core Interface State Machine](image)

**Figure 4.11 Core Interface State Machine**
This confirms the reception of packet. When the CI receives the packet whose ACK_I is active, it sends back ACK signal to Core PE indicating that the transfer is complete. This process even though is quite useful for safe data transmission, keeps the CI waiting until acknowledge is received. This can be avoided by disabling the ACK Mode. When CI acts as a receiver/destination node, it reads in the incoming packet (noc_in) to an input register (Register), unpack the message and sends the data information (DAT_O) to the PE.

4.2.1 The slave Network Interface

The slave Network Adapter/Interface (NI) and CI are fundamentally the same but they differ in some aspects. The NI waits for the network, where the master network adapter waits on the IP core.

Figure 4.12 slave Network Interface
The implementation details of Slave Network Interface module is shown in Figure 4.12. It consists of a FIFO to store the incoming packets and a FSM that generates control signals. It sends out data and control signals to Slave PE after unpacking the received packets (FIFO_OUT). When the Slave PE has to respond to the Master request, the FSM packs the message and sends it to interface.

In SF switching, as the packet travels from node to node through the network the route is updated with a return route. Hence the slave network adapter does not have to find the route back, and can follow same route followed by the incoming packet from Master. The route however is in the reverse order, so the order of the route has to be corrected. This further reduces the complexity of slave network adapter. In other words a slave network adapter does not have any routing table.

The State Machine used in the implementation of NI is shown in Figure 4.13. The FSM is initiated when a packet is available in FIFO. It then enables strobe signals indicating a read to the address specified in the incoming packet. It then determines the route/address from the incoming packet. In ACK Mode, NI would wait until it receives a ACK_* signal from Master, indicating completion of transfer, whereas in normal mode, NI continues the FSM with the next packet in FIFO.
4.2.2 Synthesis Results

Figure 4.14, 4.15 & 4.16 shows the synthesis result timing diagram of master and slave and its resource utilization is given in table. The synthesis result provides a summary and analysis of netlist generation of the design. The device used for implementation is Cyclone V SoC FPGA with a speed grade of 7 and the results given are obtained after the FPGA implementation.
Figure 4.14 Chip planner View
Deterministic and distributed routing algorithms have been presented for star, ring and mesh network topologies. The packet format is versatile, and it consists of a header that contains node identification for distributed routing, and a free-form payload that can be specified to meet any systems' needs. The handshaking flow-control protocol is employed to ensure proper inter-router communication control. FPGA implementation details of the router components and network interface are presented in the next chapter.

Figure 4.15 RTL view

Figure 4.16 Timing diagram
Table 4.2 Resource utilization table

<table>
<thead>
<tr>
<th>Component</th>
<th>Total Logic Elements</th>
<th>Total comb functions</th>
<th>Total Registers</th>
<th>Total Memory Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Master core interface</td>
<td>2543</td>
<td>1234</td>
<td>234</td>
<td>420</td>
</tr>
<tr>
<td>Slave Network interface</td>
<td>1800</td>
<td>892</td>
<td>220</td>
<td>360</td>
</tr>
</tbody>
</table>

4.3 SUMMARY

Configurable DXbar router and Network interface for both Master and slave processing unit was modeled in Verilog hardware description language and was prototype in Cyclone V SoC FPGA to find the resource requirement of our design. The Altera Quartus II v 14.1 was used to compile each component. The synthesis results consume less resource utilization, consume low power and more suitable for real time MPSoC design.
As implementing an NoC with age-based arbitration might be difficult, the arbitration could be based on the number of hops the packets have traversed. In this way, only several bits in the look-ahead signal are required to represent the number of hops of a packet. If an incoming packet wins in the arbitration, that packet could traverse the primary crossbar and go to its designated output port without being buffered. Otherwise, it will be sent to the secondary crossbar and be buffered. On the other hand, a buffered packet waits in the buffer slot or PE until it wins in arbitration. The de-multiplexers and multiplexers are set to select the primary crossbar inputs / outputs by default. Switch allocator uses the arbitration result to set the two crossbars correctly, control all of the multiplexers to link each output port of the router to the correct crossbar, and control all the de-multiplexers to direct incoming packets between the primary and the secondary crossbar.

3.10 FAIRNESS MAINTENANCE

Because the arbitration is age-based, packets coming from the nodes on the edges of the mesh network would have higher priority when they pass through the nodes in the center. As a result, the packets injected by center nodes are more likely to lose in arbitration and traverse the secondary crossbar. More importantly, this scenario could limit the injection of packets of center nodes, as injection ports and buffers have lower priority than those of input ports. A fairness issue thereby arises, since packets could be stalled in the injection ports or buffers for numerous cycles, when the desired output ports are taken by other packets from the input ports.

In order to maintain fairness between the primary crossbar and the secondary crossbar, a fairness counter is maintained in each router to count the number of times packets from the primary crossbar win consecutively in arbitration.