```
module NoC (
  input clock,
  input reset,
  input [3:0] src_id,
  input [3:0] dest_id,
  input [7:0] data_in,
  output [7:0] data_out,
  output busy
);
  // Number of nodes in the NoC
  parameter NODE_COUNT = 16;
  // Number of virtual channels per link
  parameter VC_COUNT = 4;
  // Maximum packet size
  parameter MAX_PACKET_SIZE = 128;
  // Router address width
  parameter ADDR_WIDTH = 4;
  // Router ID width
  parameter ROUTER_ID_WIDTH = 4;
  // Router input port count
  parameter INPUT_PORT_COUNT = 5;
  // Router output port count
  parameter OUTPUT_PORT_COUNT = 5;
  // Router buffer depth
  parameter BUFFER_DEPTH = 8;

  // Router module
  module router (
    input clock,
    input reset,
    input [ROUTER_ID_WIDTH-1:0] id,
    input [INPUT_PORT_COUNT-1:0] rx_valid,
    input [INPUT_PORT_COUNT-1:0] [ADDR_WIDTH-1:0] rx_dest,
    input [INPUT_PORT_COUNT-1:0] [VC_COUNT-1:0] rx_vc,
    input [INPUT_PORT_COUNT-1:0] [MAX_PACKET_SIZE-1:0] rx_data,
    output [OUTPUT_PORT_COUNT-1:0] tx_valid,
    output [OUTPUT_PORT_COUNT-1:0] [ADDR_WIDTH-1:0] tx_dest,
    output [OUTPUT_PORT_COUNT-1:0] [VC_COUNT-1:0] tx_vc,
    output [OUTPUT_PORT_COUNT-1:0] [MAX_PACKET_SIZE-1:0] tx_data
  );
    // Router control logic
    // ...
  endmodule
```

```
// Router instance array
router routers [NODE_COUNT];

// Link module
module link (
  input clock,
  input reset,
  input [ROUTER_ID_WIDTH-1:0] src_id,
  input [ROUTER_ID_WIDTH-1:0] dest_id,
  input [VC_COUNT-1:0] src_vc,
  input [VC_COUNT-1:0] dest_vc,
  input rx_valid,
  input [MAX_PACKET_SIZE-1:0] rx_data,
  output tx_valid,
  output [MAX_PACKET_SIZE-1:0] tx_data
);
  // Link control logic
  // ...
endmodule

// Link instance array
link links [NODE_COUNT][NODE_COUNT];

// Node module
module node (
  input clock,
  input reset,
  input [ROUTER_ID_WIDTH-1:0] id,
  input [VC_COUNT-1:0] rx_vc,
  input rx_valid,
  input [MAX_PACKET_SIZE-1:0] rx_data,
  output tx_valid,
  output [VC_COUNT-1:0] tx_vc,
  output [MAX_PACKET_SIZE-1:0] tx_data
);
  // Node control logic
  // ...
endmodule

// Node instance array
node nodes [NODE_COUNT];

// Connect routers to links
for (i = 0; i < NODE_COUNT; i = i + 1) begin
```

```
    for (j = 0; j < NODE_COUNT; j = j + 1) begin
      if (i != j) begin
        routers[i].tx_valid[j] = links[i][j].tx_valid;
        routers[i].tx_dest[j] = j;
        routers[i].tx_vc[j] = links[i][j].tx_vc;
        routers[i].tx_data[j] = links[i][j].tx_data;
        links[i][j].rx_valid = routers[i].rx_valid[j];
        links[i][j].rx_data = routers[i].rx_data[j];
        links[i][j].src_id = i;
        links[i][j].dest_id = j;
        links[i][j].src_vc = routers[i].rx_vc[j];
        links[i][j].dest_vc = routers[j].tx_vc[i];
      end
    end
  end

  // Connect nodes to routers
  for (i = 0; i < NODE_COUNT; i = i + 1) begin
    nodes[i].tx_valid = routers[i].tx_valid[i];
    nodes[i].tx_vc = routers[i].tx_vc[i];
    nodes[i].tx_data = routers[i].tx_data[i];
    routers[i].rx_valid[i] = nodes[i].rx_valid;
    routers[i].rx_vc[i] = nodes[i].rx_vc;
    routers[i].rx_data[i] = nodes[i].rx_data;
  end

  // Connect input and output ports to nodes
  busy = nodes[dest_id].tx_valid;
  data_out = nodes[dest_id].tx_data;
  nodes[src_id].rx_valid = 1'b1;
  nodes[src_id].rx_vc = 0;
  nodes[src_id].rx_data = data_in;
endmodule
```