The Physical link Module Between routers:

Introduction: The physical link module implements a partial serialization of input data presented and also receives partially serialized data from counterpart which is constructed back as output data. This physical links module in its current state is designed for 24 bit data serialization/deserialization, specifically designed for partitioned connect generated NOC with a flit width of 18 bits. With a few modifications this module can be made for any bit input data to 8 bit data stream and 8 bit data stream to reconstructed useful data. The physical link module maintains two buffers one for input router data and another for output data to router. Currently the buffer size is kept as 90 but an optimized design can reduce the buffer sizes.

Features:

* Any bit to 8-bit serialization
* Data reconstruction with validity check
* Buffered Input and output data
* Simple GPIO to GPIO interface
* Full duplex communication
* 2 x {n bit *(input/output data)* } / 8 clock cycles for data transfer
* Buffer full indication

Pin Description:

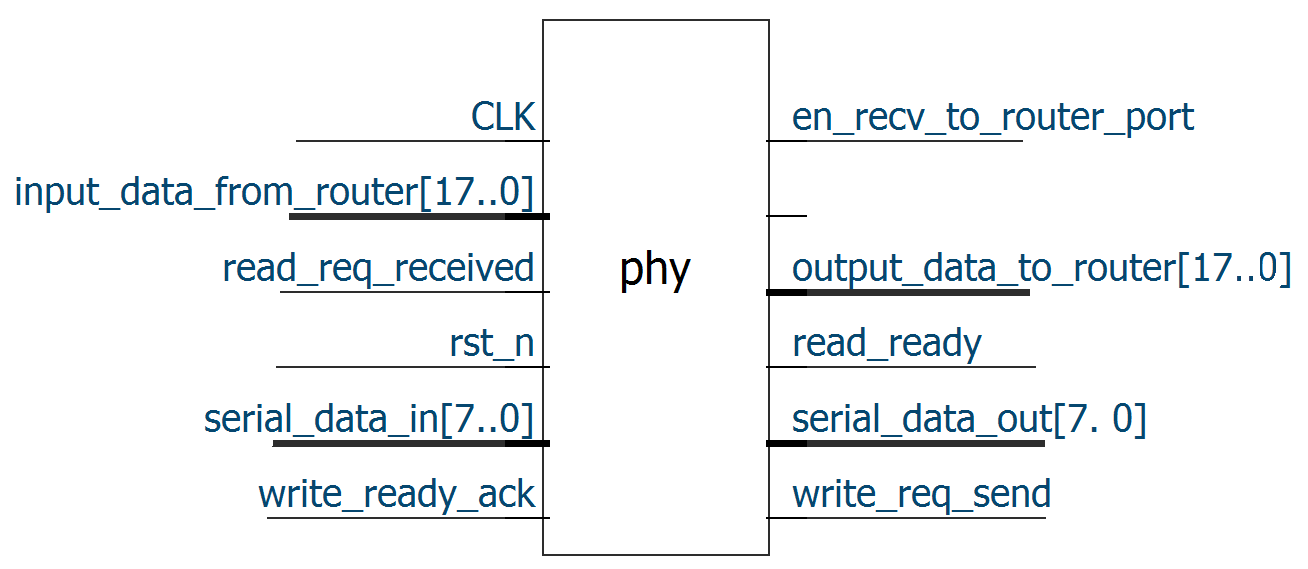


Figure: Physical Link Module

|  |  |
| --- | --- |
| **PIN** | **Description** |
| clk | Input Clock |
| rst\_n | Global Reset (Logic zero is reset state) |
| input\_data\_from\_router[17:0] | 18 bit input data(can be modified if needed) |
| serial\_data\_in[7:0] | 8 bit data input (MSB first) |
| serial\_data\_out[7:0] | 8 bit data output (MSB first) |
| output\_data\_to\_router[17:0] | Reconstructed data received from 8 bit input |

Description:

This physical module implements a simple protocol which is whenever a valid data in presented as input from router keep in buffer and start sending 8 bits at a time with MSB first. Similarly for output data to router reconstruction. Whenever a valid 8 bit MSB presented construct the data back and send out from output port. The Physical link module recognizes a valid flit from router by checking its valid bit. Rest of the bits is treated as data and no processing or alteration is done to it.

Key Block in Verilog HDL:

Preparing the input data buffer when a valid data in received.

if (input\_data\_from\_router [17])

begin

input\_buffer\_stack[a] <= input\_data\_from\_router;

en\_recv\_to\_router\_port <= 1;

a = a +1;

input\_buffer\_ready <= 1;

end

Sending the input data stored in buffer 8 bits at a time.

Case (e) 1:

begin

serial\_data\_out <= {6'b0,output\_buffer[17:16]}; // Zero Padding

e <= 2;

end

2:

begin

serial\_data\_out <= output\_buffer[15:8];

e <= 3;

end

3:

begin

serial\_data\_out <= output\_buffer[7:0];

e = 0;

output\_buffer <= 0;

Similarly two more blocks for receiving 8 bits data and sending 18 bits output data to router.