1. Features

of version 1x, we only support following features:

* support configured by core processor
  + configure refresh interval
  + router initial control
  + router reset control (can upgrade to a net reset)
* net control
  + device ID management
  + timely network refreshment
  + support domain searching
* route dataflow packets
  + route uploading request/response packets
  + route downloading request/response packets
  1. configuration feature

configuration is set from a core processor by APB bus. router can only receive 1 address with series of write requests, to execute configure commands one by one. Once the router is not available, then the PREADY signal will be asserted to invalid state.

For every write data from APB bus, the [31:16] bits are treated as command, and [15:0] are treated as corresponding value.

* + 1. router initial command

initial command is a start of router semaphore in current design, it'll do nothing but just info the router to issue a initial done broadcast to all the net.

* + 1. router reset command

software reset, clear the domain-device table,

* + 1. configurable refresh interval

controls the time interval between two refresh broadcasts.

* 1. net control feature
     1. device ID management

each time the device trying to attach, it should send out its domain, and then router will give back a valid device ID, and will record the association table about the domain-device pairs.

* + - 1. attach request format

total packet size, 22bits

* packet type: ATTACH, bit[5:0]
* domain ID of the requester, bit[15:0]
  + - 1. attach response format

total packet size, 38bits

* packet type: ATTACH-CPL, bit[5:0]
* domain ID of the requester, bit[15:0]
* device ID assigned by the router, bit[15:0]
  + 1. timely network refreshment

router will send out a broadcast refresh request to all physical connected devices, each device should give out a response within a specific timeout limit.

* + - 1. refresh request format

total packet size,

* packet type: REFRESH, bit[5:0]
* target address bits, bit[15:0] (0 value)
* expected response time limit, bit[15:0] (seconds)
  + - 1. refresh response format

total packet size,

* packet type: REFRESH-CPL, bit[5:0]
* response status message, bit[2:0], default is 0(OK)
  + 1. domain name search

can receive a domain search request and give back the corresponding device ID according to the domain name.

* + - 1. DNS request format

total packet size,

* packet type: DNS, bit[5:0]
* target address, the searching target: bit[15:0]
* source address, bit[15:0]
  + - 1. DNS OKAY response format

total packet size,

* packet type: DNS-OKAY, bit[5:0]
* target address, the address of the requester, bit[15:0]
* response message, the domain ID, bit[15:0]
  + - 1. DNS name not exists response format

total packet size,

* packet type: DNS-NNE, bit[5:0]
* target address, the address of the requester, bit[15:0]
* response message, RSV(0), bit[15:0]
  1. route dataflow communications
     1. route uploading requests

receive requests from client devices, and convert the request to target server. There will be following situations may appear:

* server is reachable, then the router will send the request and start counting the time, if timeout reached and no response received, it will send back a timeout response to the client.
* server is unreachable, which is not found by router, then will directly send back an unreached response.
  + - 1. dataflow diagram

<in the graffle>

* + 1. route downloading request
       1. dataflow diagram

TBD

1. interface

two kinds of interfaces supported by PRU: APB and NCP (rhNet Communication Port).

* 1. APB

using standardized APB4 interface, refer to the protocol docs

* 1. NCP

the rhNet Communication Port, of version 1.00, will be defined in this document.

Aiming of this interface is to transmit and receive rhNet packets by this interface, all devices among this network will use the unified interface to communication with each other.

* + 1. clock and reset

|  |  |  |  |
| --- | --- | --- | --- |
| signalName | width | I/O | description |
| ncp\_clock | 1 | I |  |
| ncp\_resetN | 1 | I |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

* + 1. control signals

packet type, addresses

the maximum Rx/Tx outstanding packet is 1, which means for request/response, the issuer can send maximum 2 packets if ACK is always not available.

|  |  |  |  |
| --- | --- | --- | --- |
| signalName | width | I/O | description |
| ncp\_pktRxReq | 1 | I | request valid signal from client/server |
| ncp\_pktRxReqAck | 1 | O | request valid signal, indicates pru has admitted this req |
| ncp\_pktTxRsp | 1 | O | response valid signal from PRU |
| ncp\_pktTxRspAck | 1 | I |  |
| ncp\_pktTxReq |  |  |  |
| ncp\_pktTxAck |  |  |  |
| ncp\_pktRxRsp |  |  |  |
| ncp\_pktRxRspAck |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

* + 1. data and packet message signals

packet data

|  |  |  |  |
| --- | --- | --- | --- |
| signalName | width | I/O | description |
| ncp\_pktRxData | 128 | I |  |
| ncp\_pktTxData | 128 | O |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

* + 1. timing diagram

TBD

1. PRU topology



(TBD), sketch

1. design structure
   1. top level hierarchy



(TBD) sketch

* 1. packet editor component

PEC, is a packet process and NCP conversion component. Assemble/Dissemble packet and send/receive on NCP port.

* + 1. NCP converter
    2. packet assembler
    3. packet dissembler

By receiving a packet, will dissemble it and trigger different signals to next component.

* + - 1. connection with DAMC

attach packet, will trigger signals to DAMC, connections with DAMC:

|  |  |  |  |
| --- | --- | --- | --- |
| signalName | width | I/O | description |
| DAMC\_PEC\_pktData | 32 | O | data, for attach request, will store the domain of device |
| DAMC\_PEC\_pktValid | 1 | O | indicates valid of a receiving packet |
| DAMC\_PEC\_pktType | 6 | O | indicates the packet type |

**available packet type for DAMC:**

* attach request (to packet dissembler)
* attach response (to packet assembler)
  + - 1. connection with DRC

similar signals as with DAMC established for connecting with DRC.

|  |  |  |  |
| --- | --- | --- | --- |
| signalName | width | I/O | description |
| DRC\_PEC\_pktDisData | 128 | O | data, for attach request, will store the domain of device |
| DRC\_PEC\_pktDisValid | 1 | O | indicates valid of a receiving packet |
| DRC\_PEC\_pktDisType | 6 | O | indicates the packet type |
| DRC\_PEC\_pktDisAddr | 16 | O | target device address this packet aimed for |

available packet type for DRC

* uploading request from client side
* uploading request to server side
* uploading response from server side
* uploading response to client side
* server unreachable response to client
* response timeout
  1. device address management component

DAMC, a component to manage all the device addresses

* 1. dataflow routine component

DRC, is a component to route upload/download packets from requester and responder.

* + 1. connection with PEC

|  |  |  |  |
| --- | --- | --- | --- |
| signalName | width | I/O | description |
| DRC\_PEC\_pktDisData | 128 | I | data, for attach request, will store the domain of device |
| DRC\_PEC\_pktDisValid | 1 | I | indicates valid of a receiving packet |
| DRC\_PEC\_pktDisType | 6 | I | indicates the packet type |
| DRC\_PEC\_pktDisAddr | 16 | I | target device address this packet aimed for |
| DRC\_PEC\_pktAssData | 128 | O |  |
| DRC\_PEC\_pktAssType | 6 | O |  |
| DRC\_PEC\_pktAssValid | 1 | O |  |
| DRC\_PEC\_pktAssAddr | 16 | O |  |
| DRC\_PEC\_pktDisPort | 16 | I | port ID indicates the packet comes from which physical port |
| DRC\_PEC\_pktAssPort | 16 | O | indicates this packet will sent to which physical port. |

* + 1. features of this component
* receive uploading/downloading requests,
  + convert and route to corresponding server port
  + if no available server found, then send back an unreachable response
* for request that sent to server, will start to count time, if time's up, drop this outstanding packet and send back a timeout response to requester
* receive uploading/downloading response
  + if outstanding is already dropped, then discard this response
  + if has corresponding outstanding request waiting, then convert and send back the response to corresponding requester.
    1. design structure

using meta flow mechanism to describe the DRC design, as an example for defining the flow.

* + - 1. assembly

selectionLogic :pktTypeSel do

in[0] = 'DRC\_PEC\_pktDisType'

out[0] = 'uploadRequestValid'

sel[0] = "6'd1"

end

* + - 1. feature
         1. feature selectionLogic definition

def feature selectionLogic , &config;

config.call

out.each do |item,index|

genv "assign #{item} = #{in[0]}==#{sel[index]}? 1'b1:1'b0"

end

end

* 1. net control component

NCC

* 1. router configuration component

RCC