Тестовое задание на должность разработчика ПЛИС

L2 Network Encryptor

Exported on 2024-02-09 15:05:47

Table of Contents

1 Функциональные требования: 4

2 Требования к выполнению: 8

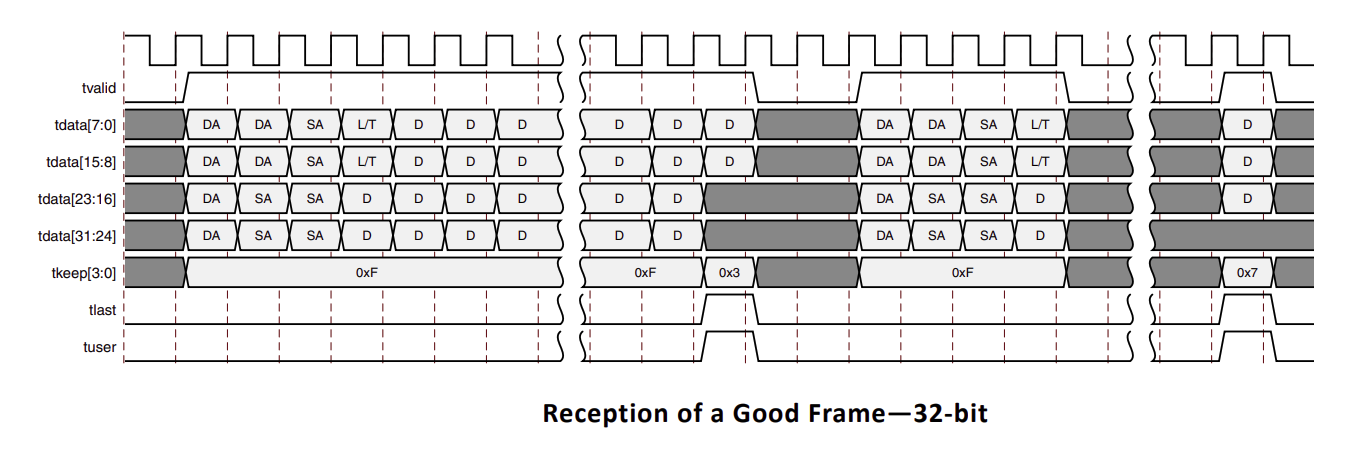
Необходимо разработать блок фильтрации IPv4 трафика и тестовое окружение  
для его проверки.

# Функциональные требования:

Блок получает от 10G Ethernet MAC v15.x (подробнее указано тут:  
<https://www.xilinx.com/support/documentation/ip_documentation/ten_gig_eth_mac/v15_0/pg072-ten-gig-eth-mac.pdf>)  
ядра трафик:

* без underrun фреймов
* с заголовками канального уровня
* без VLAN тэгов
* без MPLS меток
* без поля FCS
* минимальный интервал между пакетами 2 такта

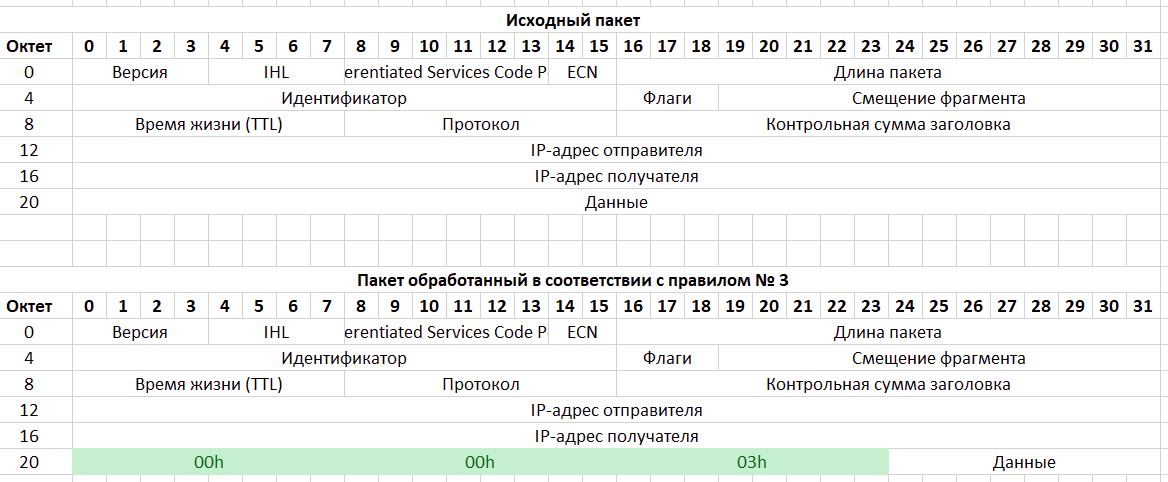
Пример приема пакета интерфейса MAC ядра представлен на рисунке:



С каждым полученным пакетом он производит следующие операции:

* Отбрасывает пакеты, для которых отсутствует запись во внутренней таблице
* Пропускает пакеты, для которых нашлась пара IP-IP, и в ставляет в начале поля данных  
  n октетов, в последнем из которых записано число n (где n - номер пары IP IP)

Пример обработки IP пакета показан на рисунке ниже (поле необязательных параметров считать  
всегда пустым):



Требования к интерфейсам:

Интерфейс AXI Stream Slave для получения трафика от MAC ядра.  
Ширина шины данных 32 бита. Частота 312.5 МГц.

Интерфейс AXI Stream Master для передачи отфильтрованного трафика  
в MAC ядро. Ширина шины данных 64 бит. Частота 156.25 МГц (независимая и  
не производная от 312.5 МГц).

Интерфейс AXI Lite Slave для записи конфигурации и управления блоком.   
Ширина шины данных 32 бита. Частота 125 МГц.  
Адресное пространство конфигурационного интерфейса представлено в таблице:

|  |  |  |
| --- | --- | --- |
| Offset | Назначение регистра | Правило |
| 0x00 | Source IP | Правило № 1 |
| 0x04 | Destination IP |
| ... | | |
| 0x38 | Source IP | Правило № 8 |
| 0x3C | Destination IP |

Описание интерфейса блока представлено ниже:

|  |
| --- |
| library ieee;  use ieee.std\_logic\_1164.all;  use ieee.numeric\_std.all;    entity test\_ip\_v1\_0 is  generic (  -- Parameters of Axi Slave Bus Interface S00\_AXI  C\_S00\_AXI\_DATA\_WIDTH : integer := 32;  C\_S00\_AXI\_ADDR\_WIDTH : integer := 6;    -- Parameters of Axi Slave Bus Interface S00\_AXIS  C\_S00\_AXIS\_TDATA\_WIDTH : integer := 32;    -- Parameters of Axi Master Bus Interface M00\_AXIS  C\_M00\_AXIS\_TDATA\_WIDTH : integer := 64  );  port (  -- Ports of Axi Slave Bus Interface S00\_AXI  s00\_axi\_aclk : in std\_logic;  s00\_axi\_aresetn : in std\_logic;  s00\_axi\_awaddr : in std\_logic\_vector(C\_S00\_AXI\_ADDR\_WIDTH-1 downto 0);  s00\_axi\_awprot : in std\_logic\_vector(2 downto 0);  s00\_axi\_awvalid : in std\_logic;  s00\_axi\_awready : out std\_logic;  s00\_axi\_wdata : in std\_logic\_vector(C\_S00\_AXI\_DATA\_WIDTH-1 downto 0);  s00\_axi\_wstrb : in std\_logic\_vector((C\_S00\_AXI\_DATA\_WIDTH/8)-1 downto 0);  s00\_axi\_wvalid : in std\_logic;  s00\_axi\_wready : out std\_logic;  s00\_axi\_bresp : out std\_logic\_vector(1 downto 0);  s00\_axi\_bvalid : out std\_logic;  s00\_axi\_bready : in std\_logic;  s00\_axi\_araddr : in std\_logic\_vector(C\_S00\_AXI\_ADDR\_WIDTH-1 downto 0);  s00\_axi\_arprot : in std\_logic\_vector(2 downto 0);  s00\_axi\_arvalid : in std\_logic;  s00\_axi\_arready : out std\_logic;  s00\_axi\_rdata : out std\_logic\_vector(C\_S00\_AXI\_DATA\_WIDTH-1 downto 0);  s00\_axi\_rresp : out std\_logic\_vector(1 downto 0);  s00\_axi\_rvalid : out std\_logic;  s00\_axi\_rready : in std\_logic;    -- Ports of Axi Slave Bus Interface S00\_AXIS  s00\_axis\_aclk : in std\_logic;  s00\_axis\_aresetn : in std\_logic;  s00\_axis\_tready : out std\_logic;  s00\_axis\_tdata : in std\_logic\_vector(C\_S00\_AXIS\_TDATA\_WIDTH-1 downto 0);  s00\_axis\_tkeep : in std\_logic\_vector((C\_S00\_AXIS\_TDATA\_WIDTH/8)-1 downto 0);  s00\_axis\_tlast : in std\_logic;  s00\_axis\_tvalid : in std\_logic;    -- Ports of Axi Master Bus Interface M00\_AXIS  m00\_axis\_aclk : in std\_logic;  m00\_axis\_aresetn : in std\_logic;  m00\_axis\_tvalid : out std\_logic;  m00\_axis\_tdata : out std\_logic\_vector(C\_M00\_AXIS\_TDATA\_WIDTH-1 downto 0);  m00\_axis\_tkeep : out std\_logic\_vector((C\_M00\_AXIS\_TDATA\_WIDTH/8)-1 downto 0);  m00\_axis\_tlast : out std\_logic;  m00\_axis\_tready : in std\_logic  );  end test\_ip\_v1\_0; |

Code Block 1 Интерфейс модуля

# Требования к выполнению:

1. Тест должен инициализировать конфигурационную таблицу блока фильтрации.
2. Автоматически завершаться и выводить статистику по количеству отброшенных  
   пакетов после успешной отправки на вход блока 1000 пакетов.
3. Нельзя использовать готовые ip ядра из библиотеки Xilinx