TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik, Inst. für Technische Informatik, Prof. für VLSI-Entwurfssysteme, Diagnostik und Architektur

Implementierung eines universellen IPv6 Protokollstapels

Kolloquium zum Masterpraktikum

Patrick Lehmann
Patrick.Lehmann@tu-dresden.de

Dresden, 04.03.2014

DRESDEN concept
Exzellenz aus Wissenschaft und Kultur

Agenda

1 Aufgabenstellung

2 Entwurf

3 Implementierung

4 Simulation

5 Auswertung

AUFGABENSTELLUNG

1 Aufgabenstellung

Zielstellung des Masterpraktikums

- Verwendung des Ethernet Controllers der PoC-Library
- Implementierung mehrerer Ethernet-Adapter pro physikalischem Port
- Implementierung eines IPv6-Protokollstapels
  - IPv6-Pakete senden und empfangen
  - UDP-Pakete senden und empfangen
  - Adressen Auflösen per NDP
- Generische Hardwarebeschreibung
- Optimierung für Streaming-Verarbeitung

1 Aufgabenstellung

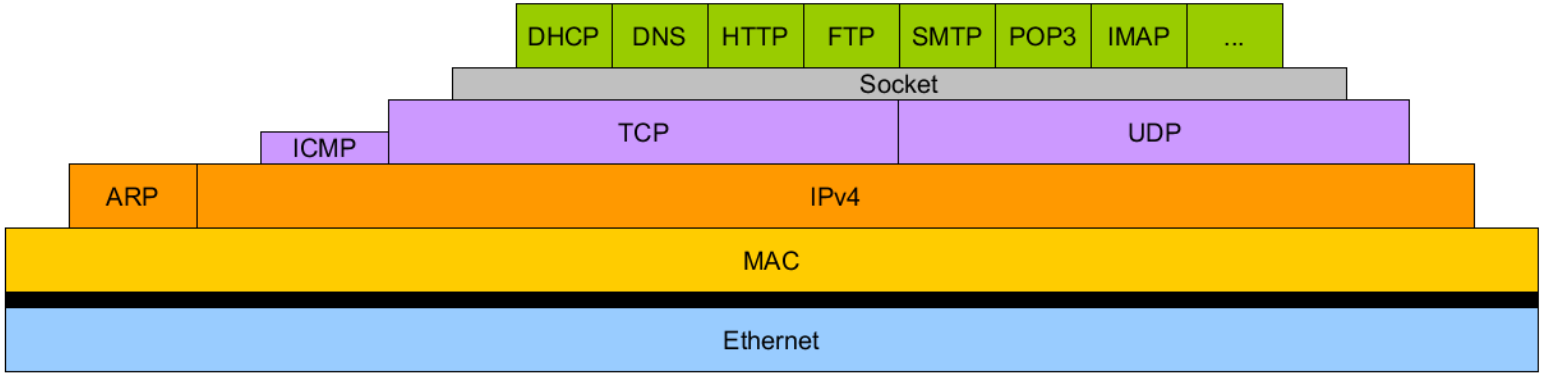
Besonderheiten

- IPv6 benutzt 128-Bit Adressen
  - Register-Ressourcen
  - Verdrahtungs-Ressourcen
  - Multiplexer-Ressourcen (LUTs)
- Universelles Protokoll für den Datenaustausch zwischen den Protokollschichten => PoC.Stream
- Caches
  - Assoziativität
  - Anzahl der Cache-Zeilen
  - Ersetzungsstrategie
- Automatisierte Simulation der Schichten und Module

ENTWURF

2 Entwurf

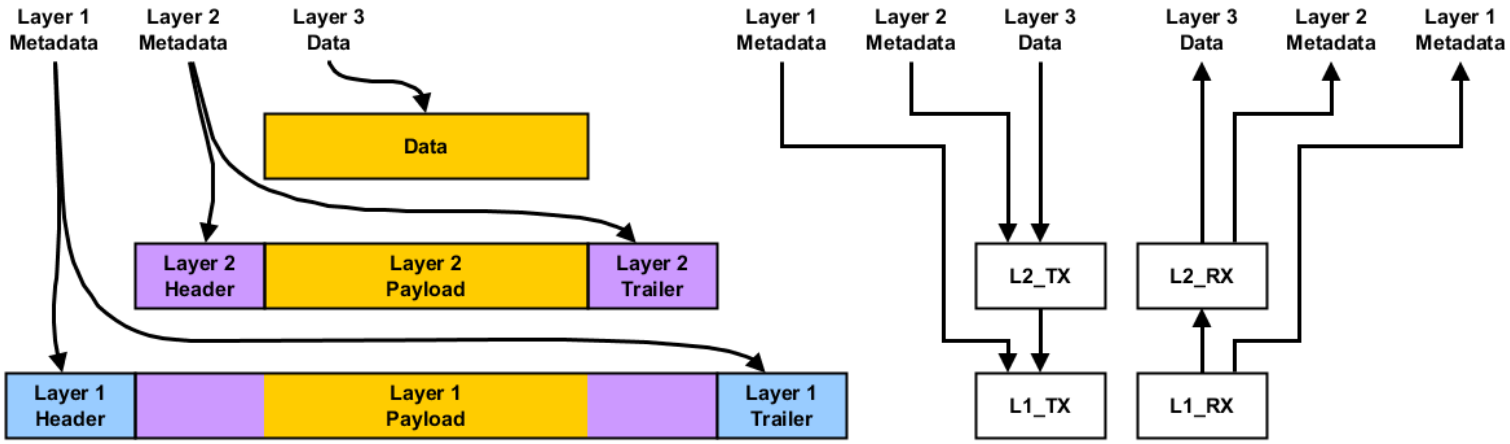
Protokollstapel



TCP/IP-Protokollstapel inkl. Anwendungsprotokollen [1].

2 Entwurf

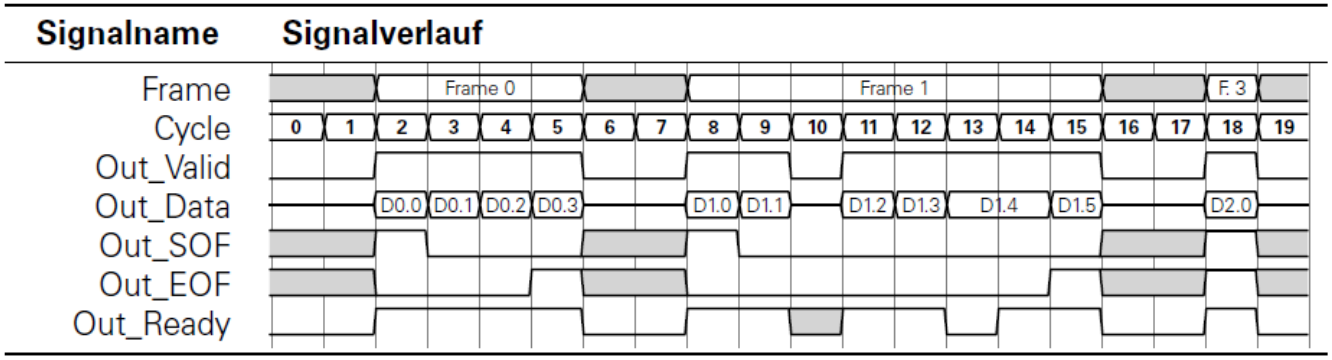
Prinzip der Rahmenbildung (framing, encapsulation)



Beispielhafte 3-Schichtenarchitektur; Abbildung der Umrahmung auf HDL-Module.

2 Entwurf

Das Streaming-Protokoll „PoC.Stream“

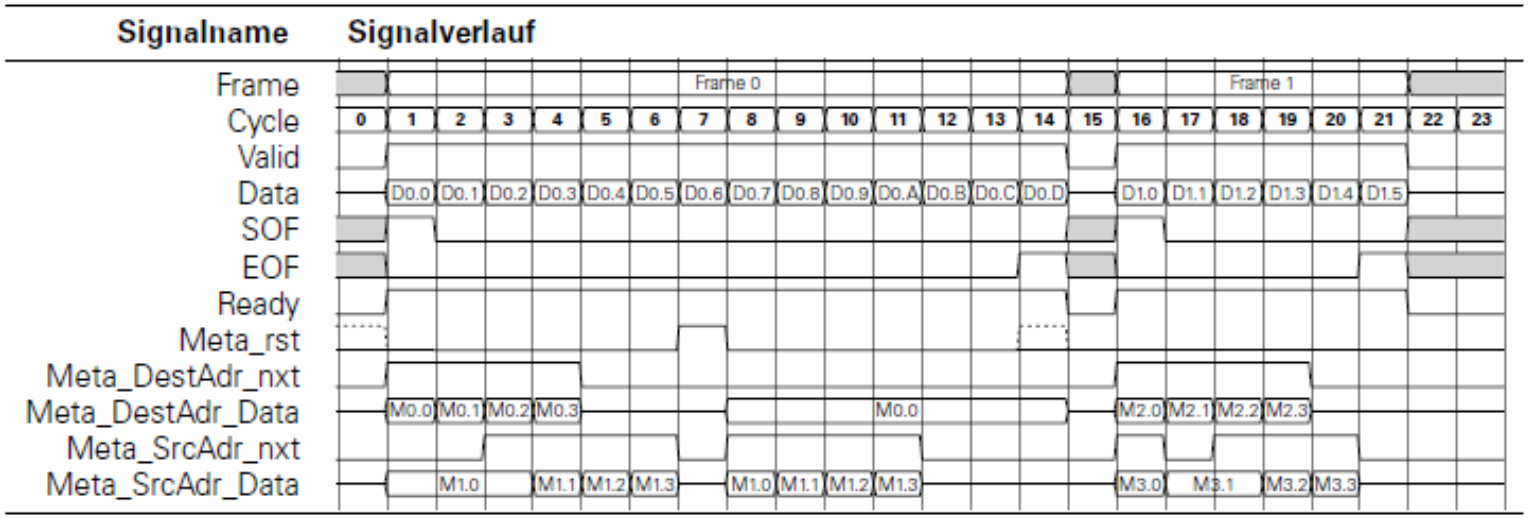


Signalverlauf dreier Frames über einen PoC.Stream Kanal.

- Wie überträgt man Metadaten?

2 Entwurf

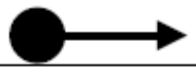
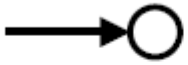
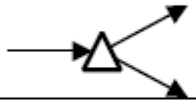
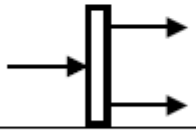
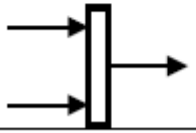
Das erweiterte „PoC.Stream“ Protokoll



Signalverlauf zweier Frames über einen PoC.Stream Kanal, welcher mit zwei Metadatenströmen annotiert ist.

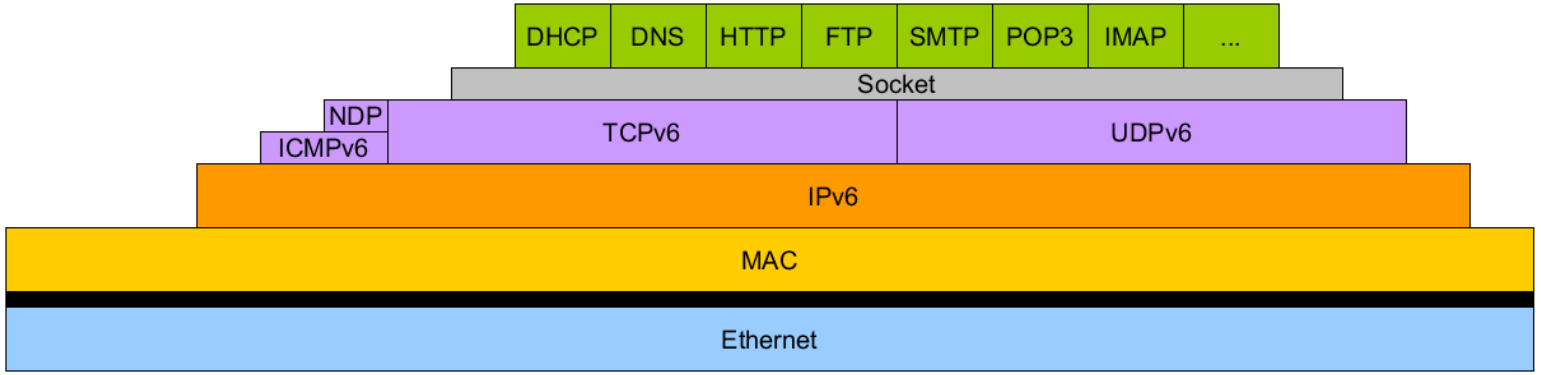
2 Entwurf

Datenflussoperatoren

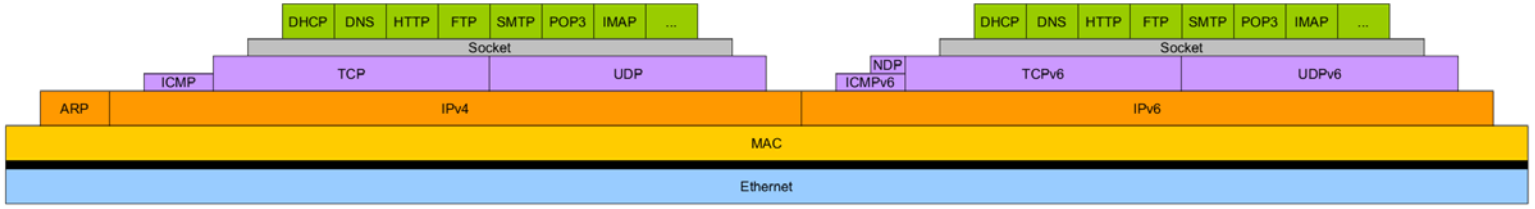
Operator	Symbol	Beschreibung
Datenquelle		Die Datenquelle stellt einen PoC.Stream Datenstrom bereit.
Datensenke		Die Datensenke nimmt einen PoC.Stream entgegen und kann diesen gegen einen Referenz-Stream vergleichen.
Datenspiegel		Der Datenspiegel vervielfältigt einen PoC.Stream unter Einhaltung des Flusskontrollprotokolls.
Demultiplexer		Der Demultiplexer gibt einen PoC.Stream an nur einen Ausgangsport weiter.
Multiplexer		Der Multiplexer schaltet einen PoC.Stream Datenstrom eines Eingangs durch.
Puffer		Der Puffer puffert einen PoC.Stream ähnlich einer Standard-FIFO.

2 Entwurf

Dualer Protokollstapel (IP dual stack)



IPv6-Protokollstapel inkl. Anwendungsprotokollen [1].



2 Entwurf

Abstraktion der Inter-Schichtenkommunikation als Bus

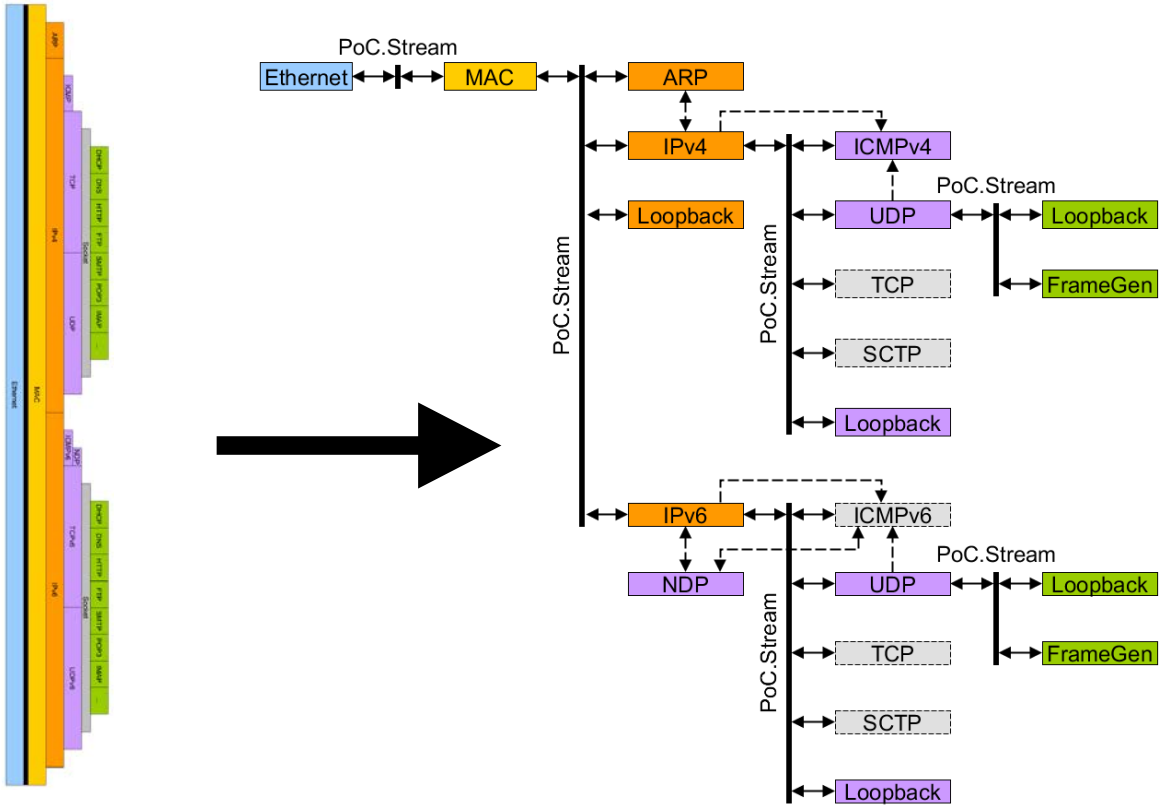
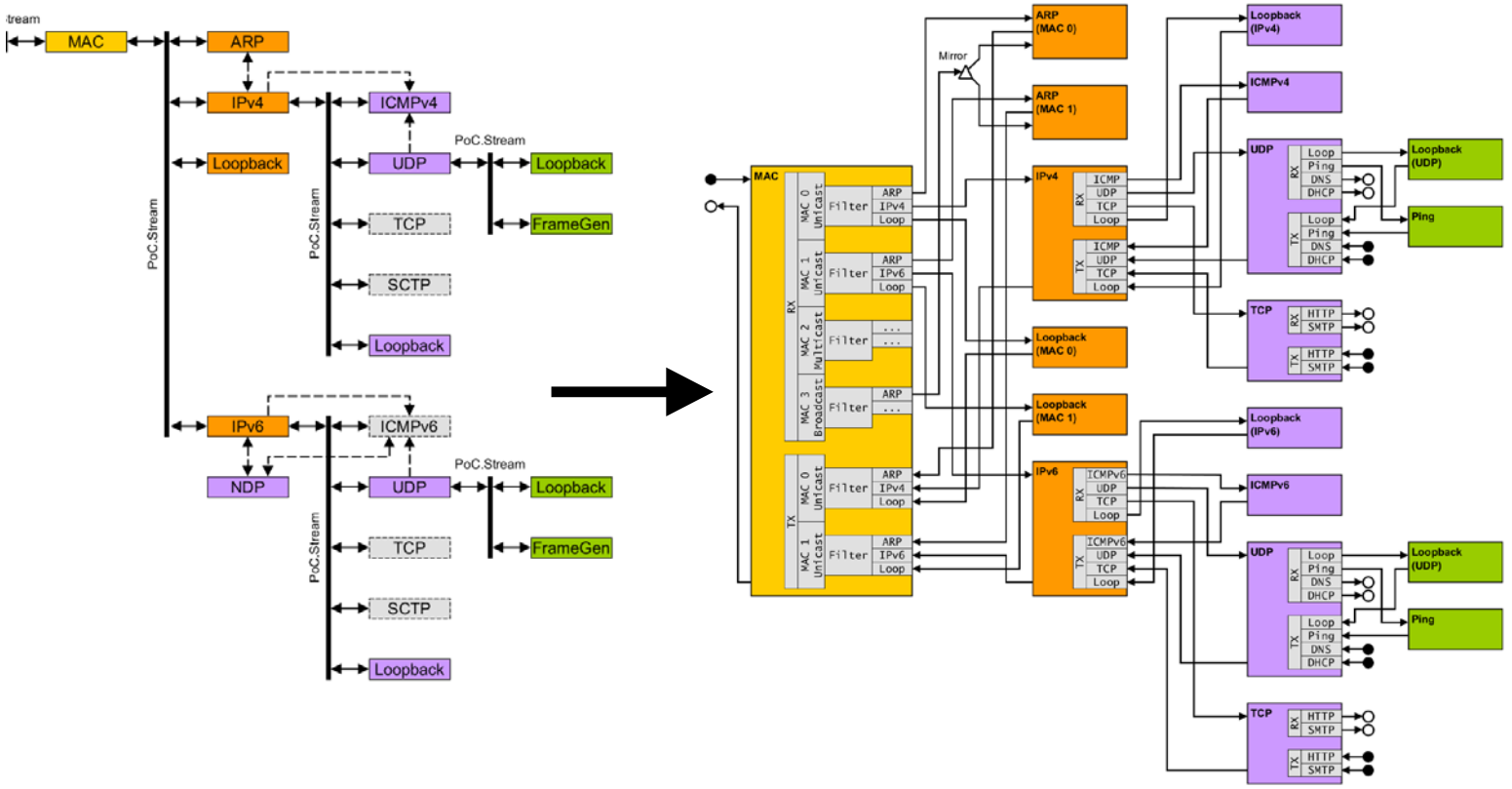


Abbildung der Schichten und Protokolle auf Module und Busse.

2 Entwurf

Transformation in eine Datenflussarchitektur

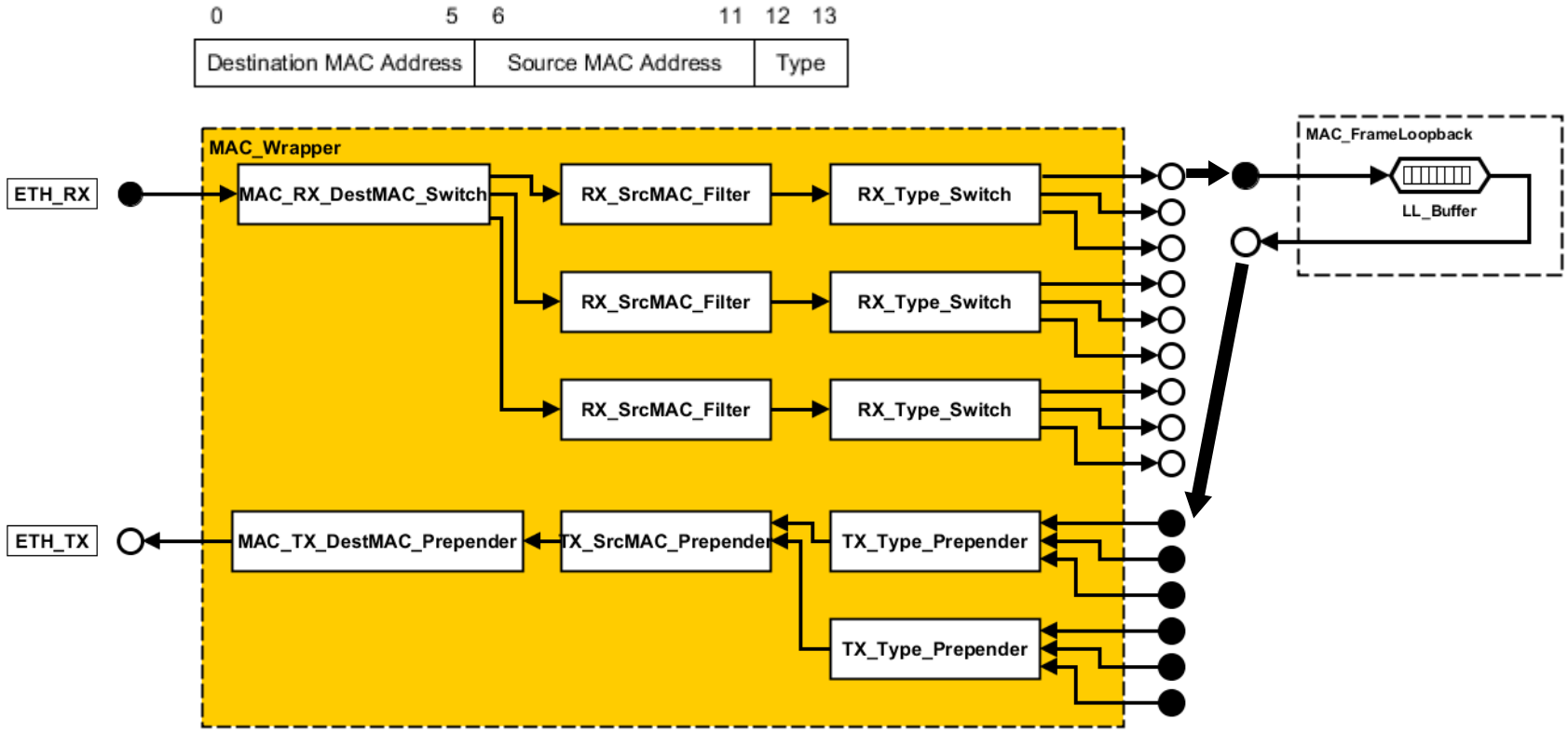


Transformation der Busse in unidirektionale PoC.Stream Kanäle.

IMPLEMENTIERUNG

3 Implementierung

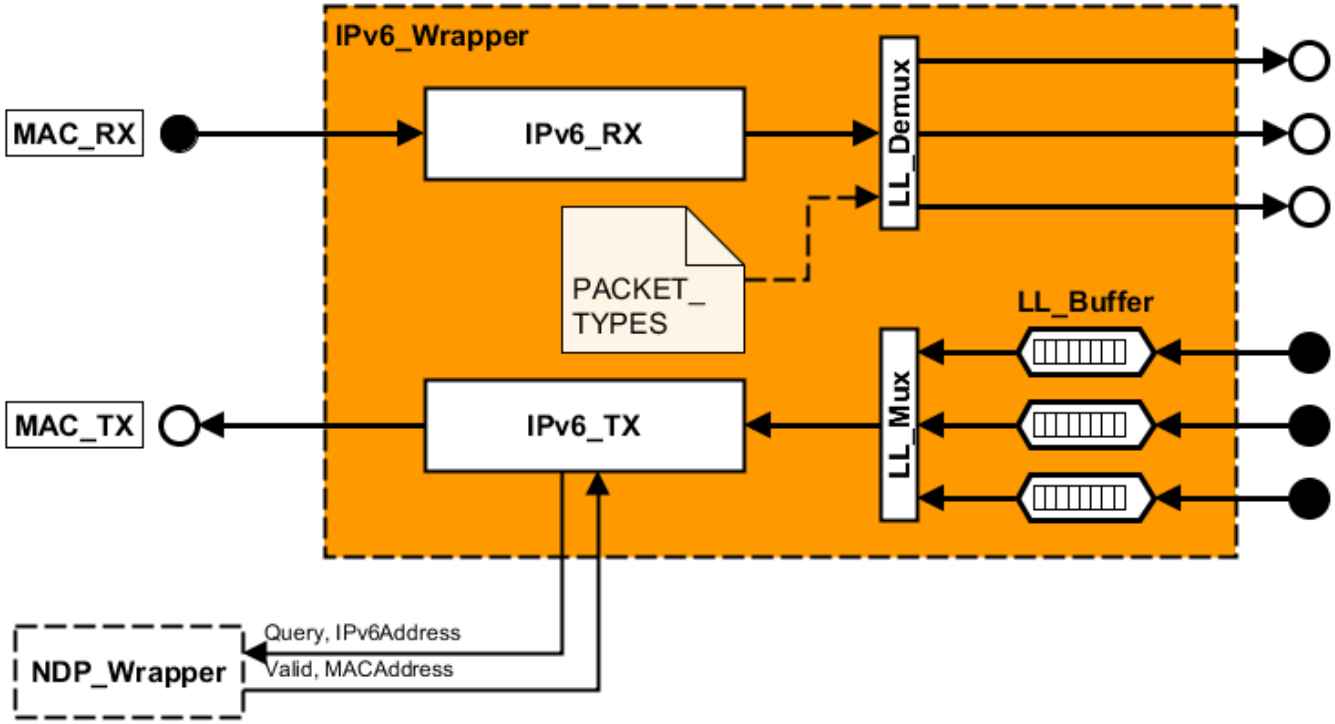
Media Access Control (MAC) - Layer



Aufbau eines Type 3.1 a) Basic-Ethernet-Frames und dessen Verarbeitung im MAC-Layer [2].

3 Implementierung

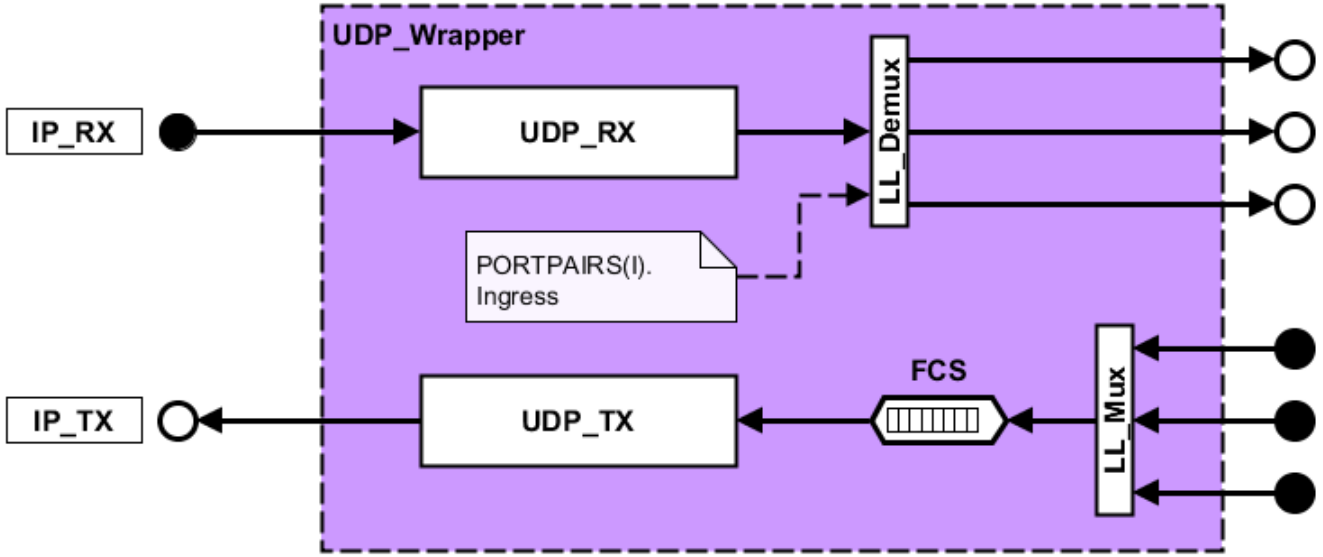
Internet Protocol version 6 (IPv6) - Layer



Verarbeitung von IPv6 Paketen im IPv6-Layer.

3 Implementierung

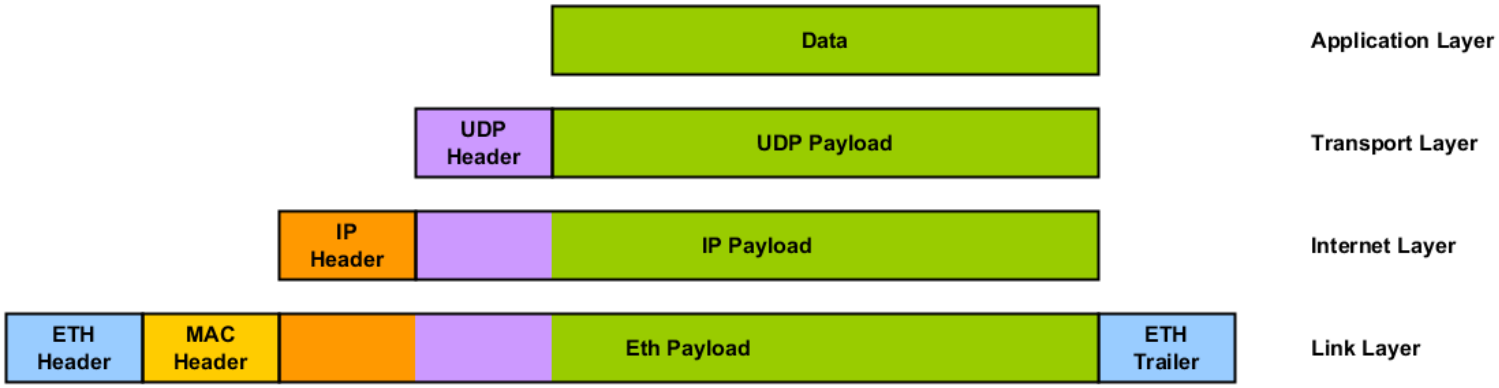
User Datagram Protocol (UDP) - Layer



Verarbeitung von UDP-Paketen im UDP-Layer. Das Modul FCS berechnet die 16-Bit 1er-Komplement Prüfsumme über den Nutzdatenanteil des UDP-Paketes.

3 Implementierung

Rahmenbildung am Beispiel von UDP

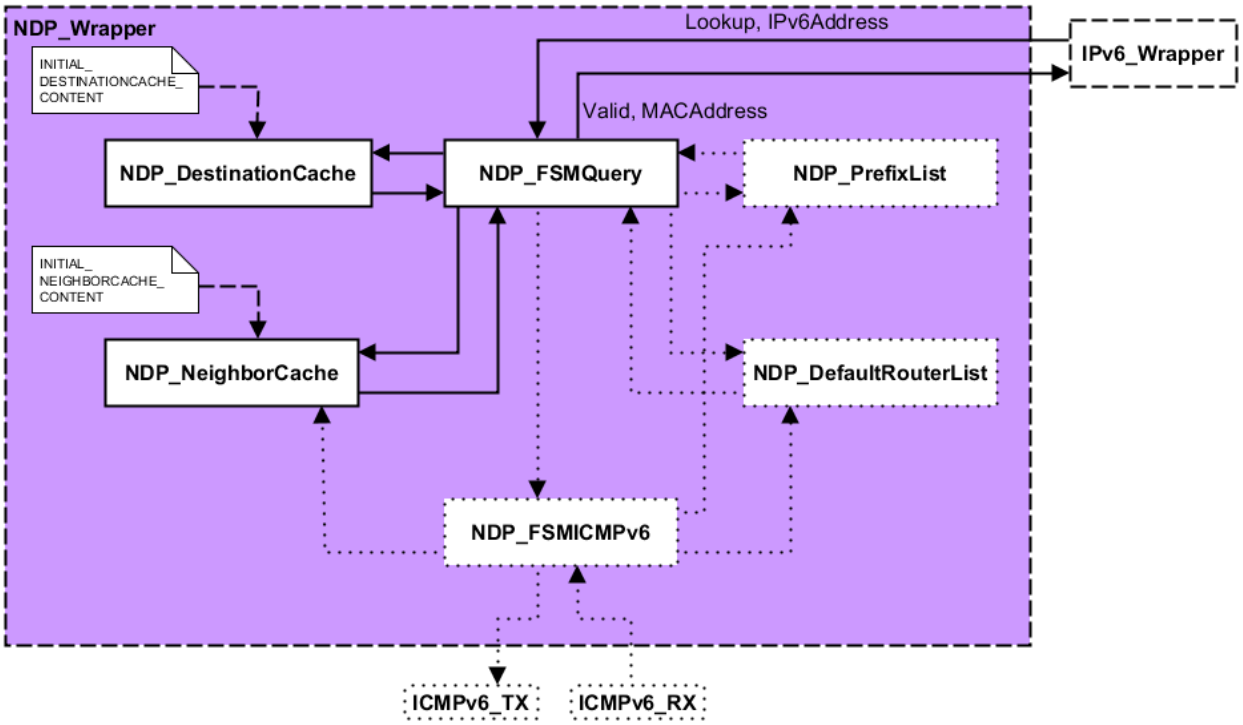


Nutzdatenanteil: 94,41% (bei einem Type 3.1 a) Basic-Frame)

Nettodatenrate: 112,55 MiByte/s

3 Implementierung

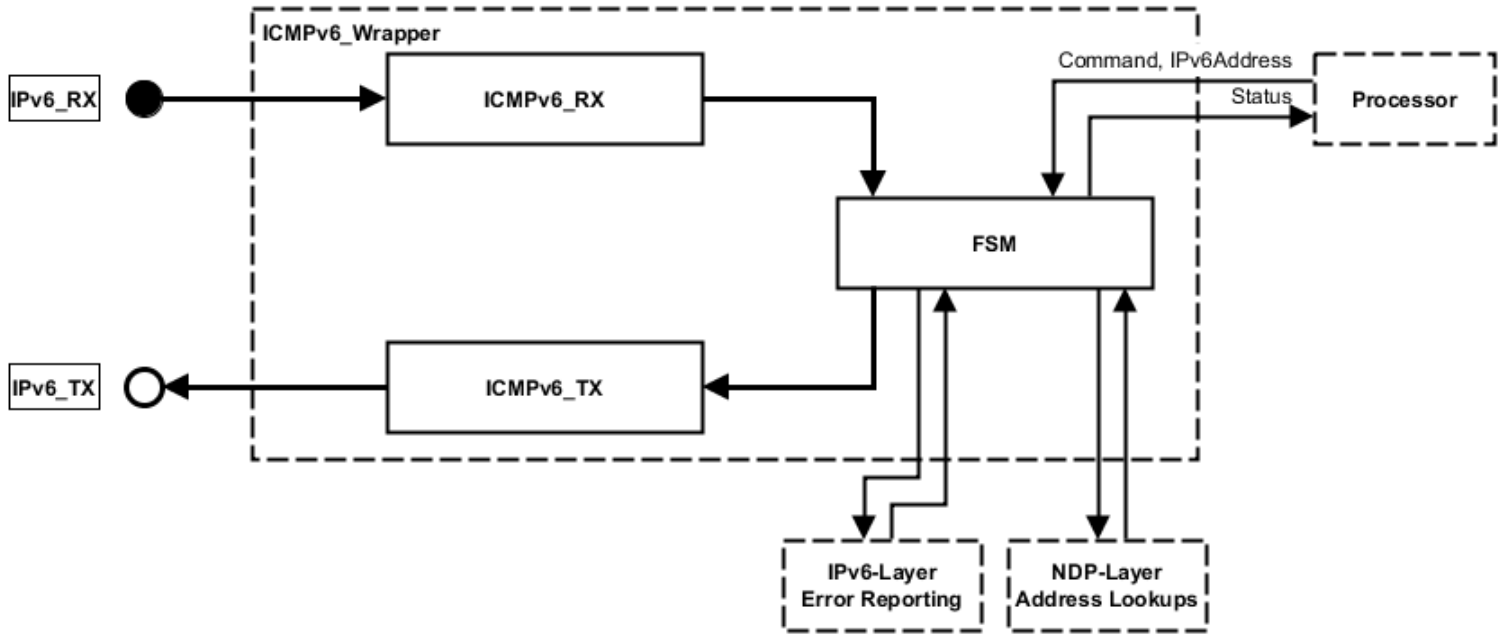
Neighbor Discovery Protocol (NDP) - Layer



Teilweise implementiertes NDP-Layer: aktuell nur 2-stufiger Cache-Lookup-Prozess.

3 Implementierung

Internet Control Message Protocol for IPv6 (ICMPv6) - Layer



Vorbereitetes ICMPv6-Layer.

SIMULATION

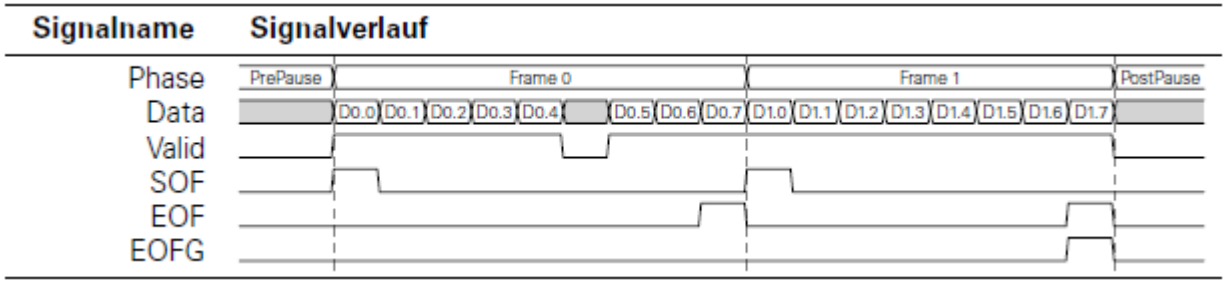
4 Simulation

Datenstrukturen

```
1 TYPE T_SIM_LL_WORD_8 IS RECORD
2   Valid      : STD_LOGIC;
3   Data       : T_SLV_8;
4   SOF        : STD_LOGIC;
5   EOF        : STD_LOGIC;
6   Ready      : STD_LOGIC;
7   EOFG       : BOOLEAN;
8 END RECORD;
```

```
1 TYPE T_SIM_LL_WORD_32 IS RECORD
2   Valid      : STD_LOGIC;
3   Data       : T_SLV_32;
4   SOF        : STD_LOGIC;
5   EOF        : STD_LOGIC;
6   Ready      : STD_LOGIC;
7   EOFG       : BOOLEAN;
8 END RECORD;
```

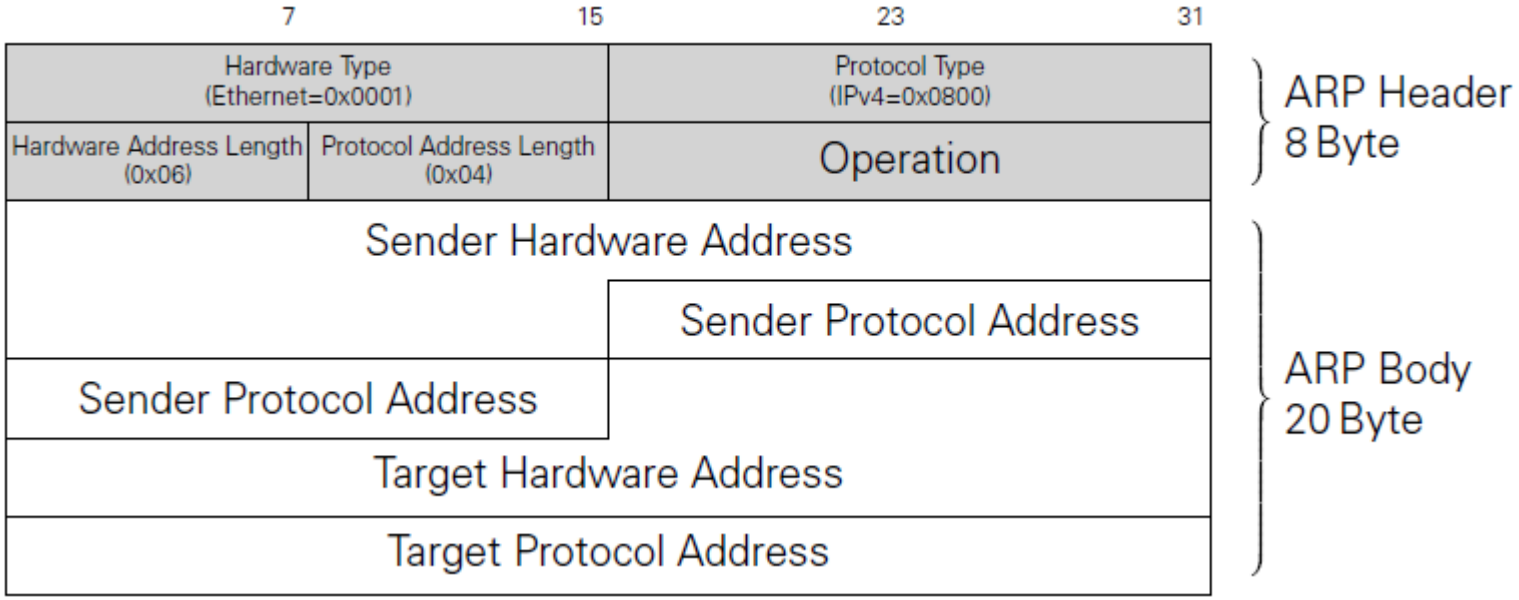
Definition eines 8-Bit bzw. 32-Bit Simulationswortes.



Signalverlauf einer 2 Frames langen Frame-Gruppe.

4 Simulation

Beispiel: ARP-Paket (1)



Aufbau eines ARP-Paketes für IPv4 zu Ethernet Adressauflösungen [3].

4 Simulation

Beispiel: ARP-Paket (2)

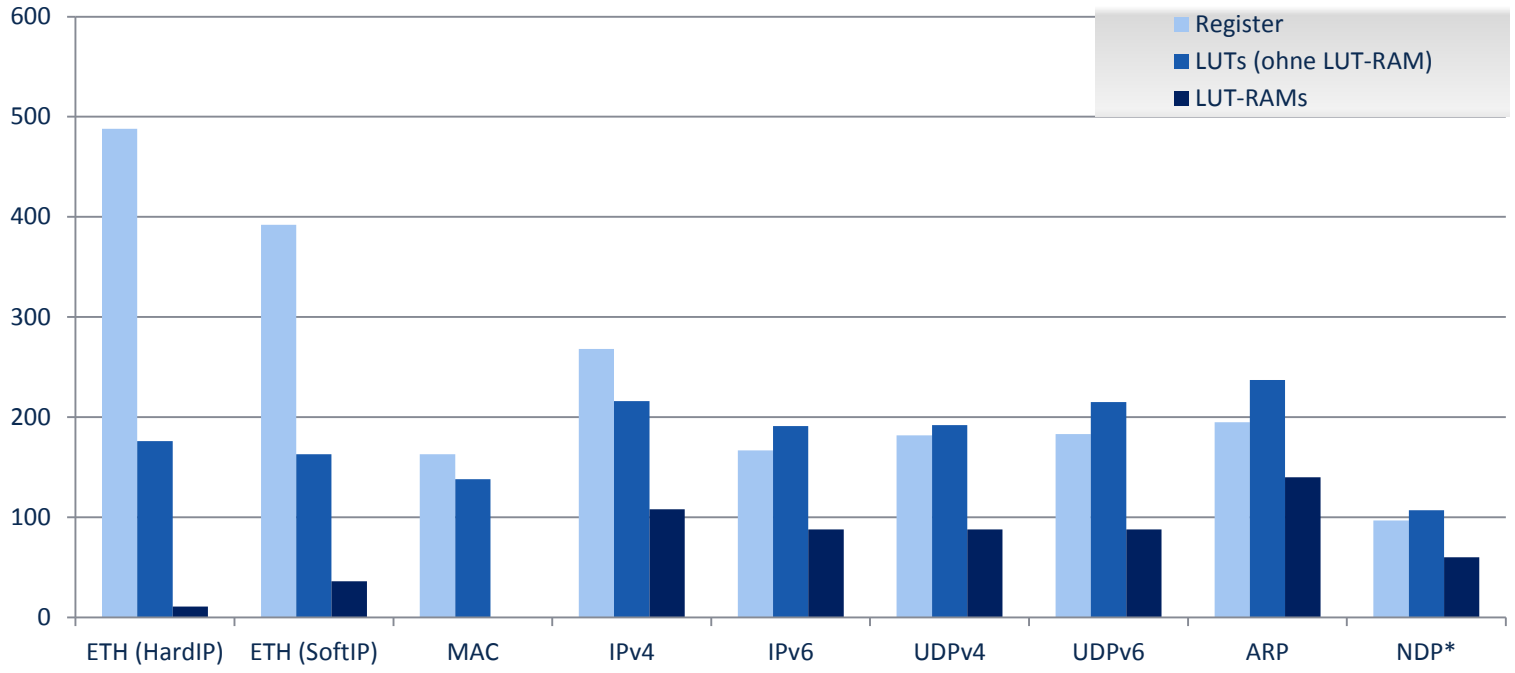
```
-- packet 0 - ARP Packet
-- ------------------------------------------------------------------------------
TC(Cnt).Active          := TRUE;
TC(Cnt).Name            := resize("Frame 0 (ARP UCRsp)", TC(Cnt).Name'Length);
TC(Cnt).PrePause        := 100;
TC(Cnt).PostPause       := 0;
-- ARP Header
TC(Cnt).Data( 0)        :=      sof(x"00");     -- hardware type - byte 1   0x0001 = ethernet
TC(Cnt).Data( 1)        :=      dat(x"01");     -- hardware type - byte 0
TC(Cnt).Data( 2)        :=      dat(x"08");     -- protocol type - byte 1   0x0800 = IPv4
TC(Cnt).Data( 3)        :=      dat(x"00");     -- protocol type - byte 0
TC(Cnt).Data( 4)        :=      dat(x"06");     -- hardware length          48 bit => 6
TC(Cnt).Data( 5)        :=      dat(x"04");     -- protocol length          32 bit => 4
TC(Cnt).Data( 6)        :=      dat(x"00");     -- operation - byte 1       request => 1
TC(Cnt).Data( 7)        :=      dat(x"02");     -- operation - byte 0       response => 2
-- ARP Payload
TC(Cnt).Data( 8 TO 13) :=      dat(dir(to_slvv_8(to_net_mac_address("11:19:99:12:3D:88"))));
TC(Cnt).Data(14 TO 17) :=      dat(dir(to_slvv_8(to_net_ipv4_address("192.168.10.5"))));
TC(Cnt).Data(18 TO 23) :=      dat(dir(to_slvv_8(to_net_mac_address("AA:BB:CC:DD:EE:FF"))));
TC(Cnt).Data(24 TO 27) := eofg(eof(dir(to_slvv_8(to_net_ipv4_address("192.168.10.10")))));
-- update count values
TC(Cnt).DataCount     := CountPatterns(TC(Cnt).Data);
Cnt                   := Cnt + 1;
-- [...]
```

Definition eines ARP-Paketes zur Simulation.

AUSWERTUNG

5 Auswertung

Ressourcenbelegungen je Schicht



* NDP zur Zeit nur mit „Read-Only“-Caches; ohne Update Funktionalität.

Virtex-5 Gesamtressourcen: 28.800 Register; 28.800 LUTs; 60 BlockRAMs

5 Auswertung

Zusammenfassung (1)

- Implementierung der TCP/IP Protokolle als Streaming-Architektur ist möglich:
  - Ethernet PoC-Modul, erweitert um einen PHYController
  - MAC inkl. Umsetzung mehrerer Netzwerkadapter
  - ARP als Cache Testplattform
  - IPv6 Standard Paketformat ohne optionale Felder
  - IPv4 durch Rückportierung von IPv6
  - ICMPv4 Echo_Request/Echo_Reply
  - UDP für IPv4 und IPv6
  - NDP nur Caches mit 2-stufigem Lookup
  - Loopback für div. Schichten

5 Auswertung

Zusammenfassung (2)

- Streaming-Verarbeitung wurde mittels On-Chip Logic-Analyser überprüft
- Geringer Ressourcenbedarf je Schicht
- NDP und weitere Hilfsprotokolle erfordern aufwendige Automaten und Algorithmen

➢ Implementierung mit einem Soft-Prozessor?

QUELLEN

Quellen

[1] The Internet Engineering Task Force – Request for Comments. http://www.ietf.org/rfc.html

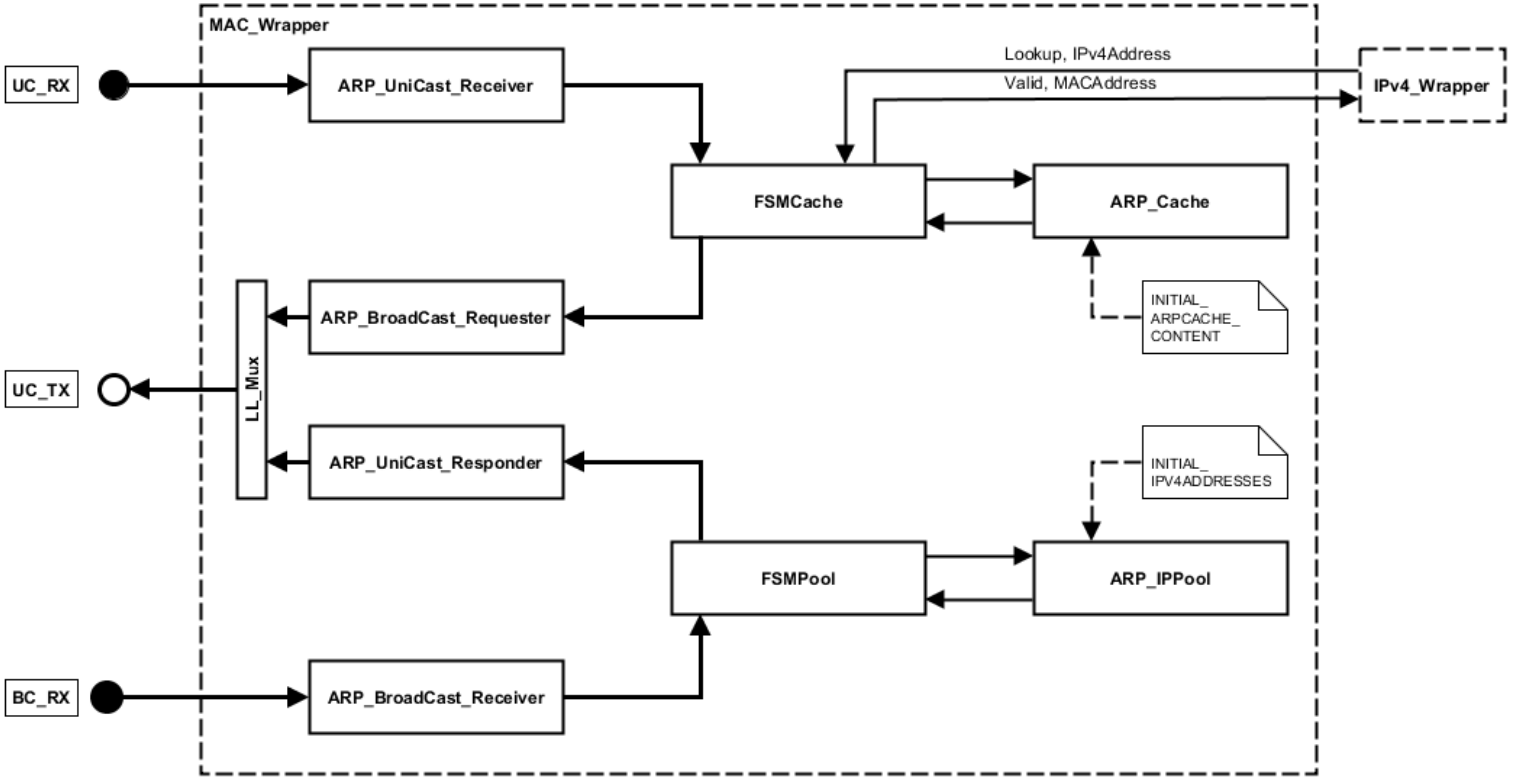
[2] Institute of Electrical and Electronics Engineers: IEEE Standard for Ethernet - Section 1-6. *IEEE Std 802.3-2012. IEEE* 2012.

[3] Plummer, D.: Ethernet Address Resolution Protocol. RFC 826 (Internet Standard). Internet Engineering Task Force, Nov. 1982

ANHANG

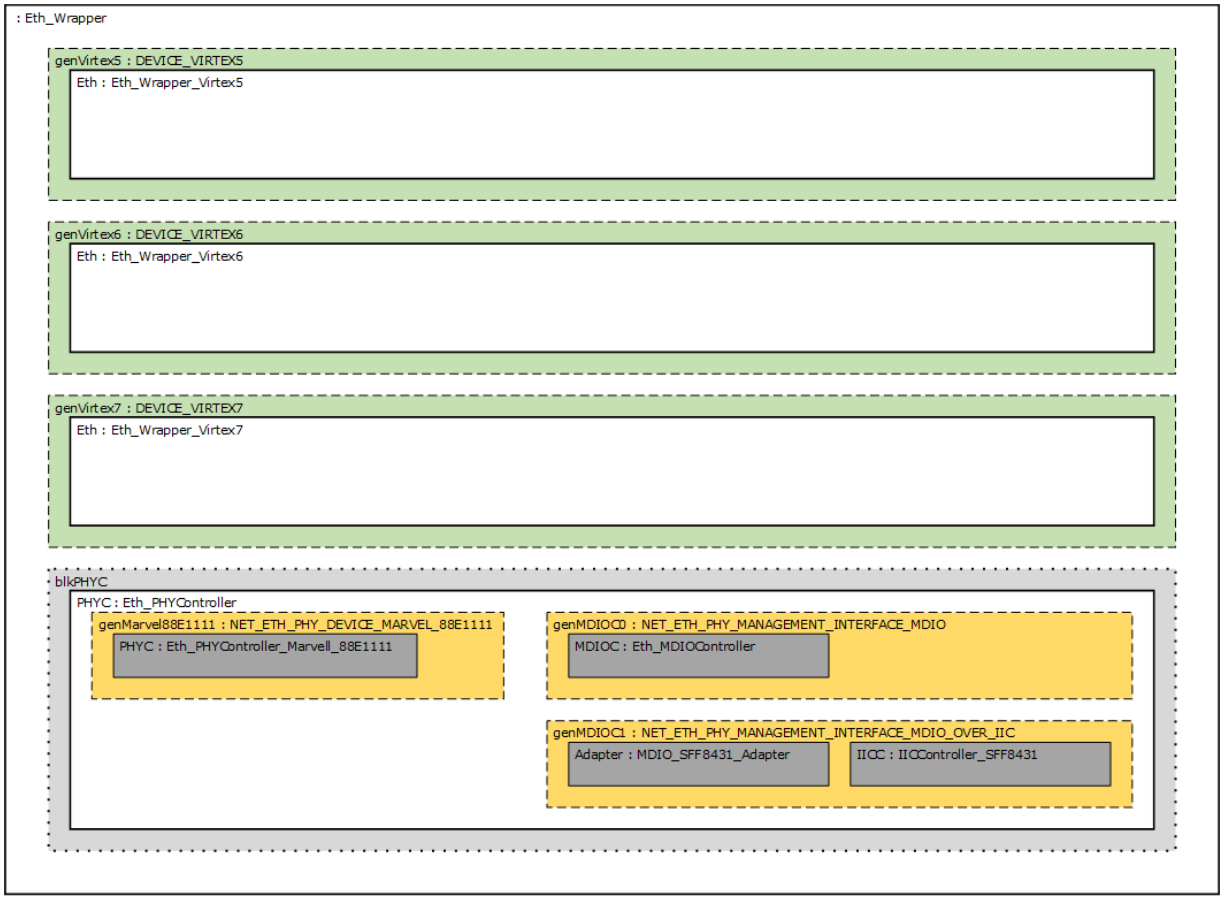
Implementierung

ARP-Layer



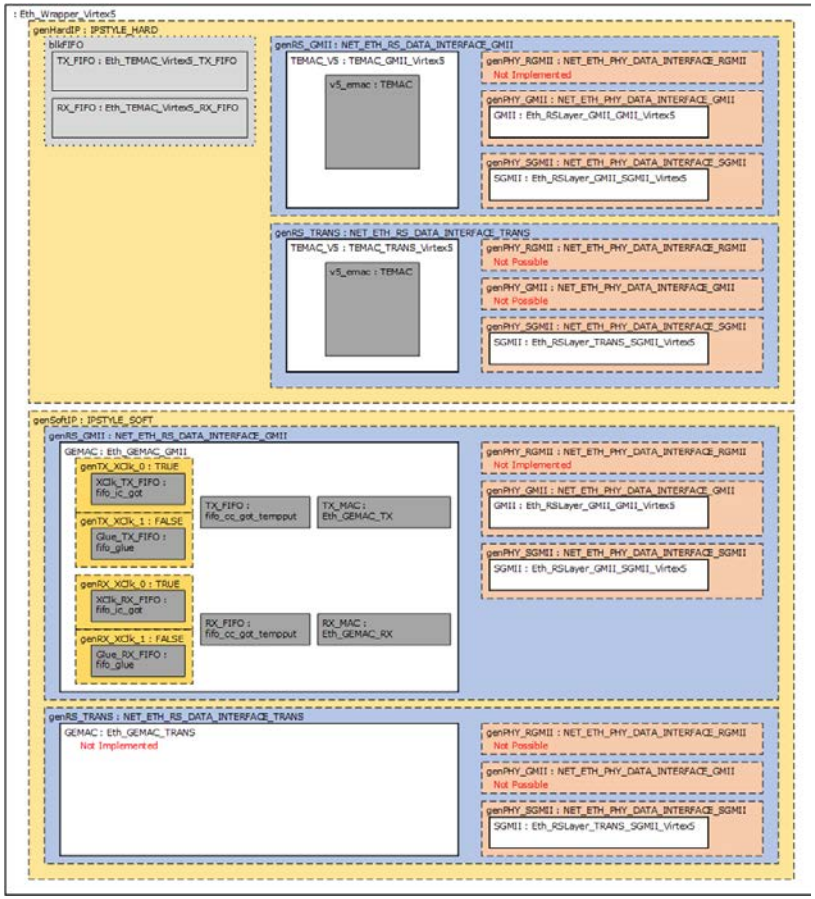
Anhang

Ethernet_Wrapper



Anhang

Ethernet_Wrapper_Virtex5



Anhang

VHDL-Typen