Data Serializer 2
Sequenzer 2
Serializer TX Pfad 3
Serializer RX Pfad 4
RX Pfad 5
TX Pfad 7
CRC Generator 9
Registerbeschreibung 11
Registerübersicht 11
CRC Controlregister 11
Controlregister 11
Statusregister 11
Paket senden 13
Paket empfangen 14

Data Serializer

Der Data Serializer schreibt die Daten in die TX FIFO oder liest diese aus der RX FIFO. Dieser besteht aus den folgenden 3 Teilen.

Sequenzer

Der Sequenzer soll die Wortzugriffe des MIPS Kerns auf die FIFO Bytezugriffe ummünzen. Ein Wortzugriff wird von diesem zu 4 Byte zugriffen der FIFO umgemünzt. Dies gilt für den TX Pfad, beim RX Pfad werden FIFO Bytezugriffe zu Wortzugriffen. Gleichzeitig muss die MIPS Endianess berücksichtigt werden, diese ist Big Endian. Also bei einem Wortzugriff befinden sich das Least Significant Byte auf den Bits 24…31.
Denn der Schreib- und Lesevorgang von 4 Byte soll durch einen Wortbefehl ausgeführt werden für eine Beschleunigung des Paketkopiervorgangs um den Faktor 4! Dies bedingt aber, dass die Bytes vertauscht ankommen (Big Endian).

Der Sequenzer besteht aus einer Reihe von hintereinander geschalteten D-FF. Der Eingang der Kette ist das write enable Signal aus dem Steuerwerk. Aus dieser Kette wird zuerst Byte3 (Datenbits 24…31) angesteuert und danach Byte 2, 1 und 0 (Datenbits 7…0).
Allerdings müssen beim Schreibvorgang auch Halbwort- und Bytezugriffe erlaubt sein. Denn ein Netzwerkpaket hat nicht immer eine /4 teilbare Größe und ein zu großes Paket löst beim Empfänger einen Fehler aus.
Daher muss die Kette aufgeteilt werden:
Ein Bytezugriff darf nur Byte 3 zur FIFO senden.
Ein Halbwortzugriff muss Byte 3 und dann Byte 2 zur FIFO senden.
Ein Wortzugriff muss Byte 3, Byte 2, Byte 1 und Byte 0 zur FIFO senden.

Dies geschieht durch die Zuhilfenahme der nram_we Signale des Speicherbusses. Dieses Signal besteht aus 4 Bit und gibt an welche Bytes geschrieben werden sollen. Der Sequenzer unterstützt somit das Schreiben des Halbwortes in den Bits 16…31 und des Bytes 24..31.
Beim Byte Schreiben wird also nur ein D-FF getriggert das den Byte 3 Enable ansteuert.
Beim Halbwort Schreiben wird ein Schatten D-FF getriggert welches danach das Byte 2 D-FF triggert. Beim Wort Schreiben werden dann 2 Schatten D-FF getriggert und danach 2 D-FF für die Bytes 1 und 0.

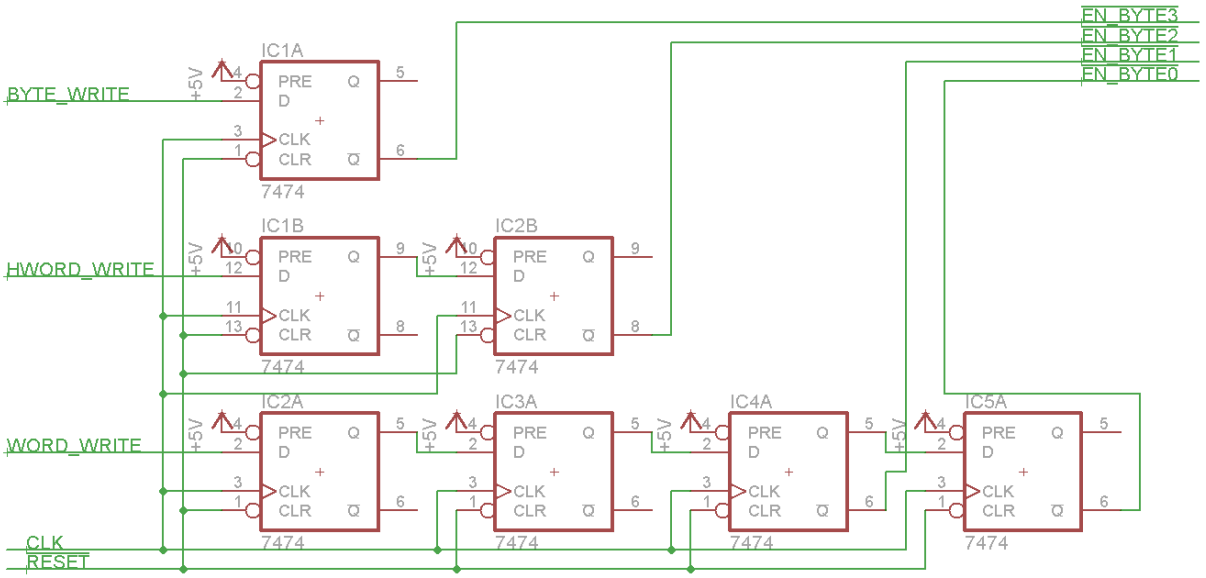

Abbildung 1 Schema des aufgeteilten Sequenzers

Bei einem Lesevorgang (welche immer wortweise erfolgen) werden durch eine Veroderung alle 3 Eingänge des Sequenzers getriggert. Weiterhin ist das nmem_oe Signal für 4 Takte LOW wenn es aus dem Steuerwerk erzeugt wird und für 2 Takte LOW wenn dieses aus dem DMA erzeugt wird. Der Impuls am Eingang des Sequenzers darf aber nur 1 Takt aktiv sein und der Sequenzer darf erst nach dem Auslesen durch CPU/DMA aktiviert werden für den nächsten Lesezugriff. Daher wird auf die steigende Flanke von nmem_oe getriggert.

Serializer TX Pfad

Es existiert kein Steuerregisterbit zum Umschalten zwischen RX und TX Pfad. Dies erfolgt implizit aus den Flags der RX FIFO. Wenn kein RX Vorgang aktiv ist, dann steuert der Sequenzer den TX Pfad. Wenn die RX FIFO leer ist oder das Data Valid Flag false ist, dann ist der TX Pfad aktiv. Ist dieser Pfad nicht aktiv, dann sind Schreibvorgänge verboten! Beide Flags sind daher über das Statusregister auslesbar.

Bei einem Schreibvorgang, werden zunächst die 1 bis 4 Byte in einem Register gespeichert. Die Write Enable Ansteuerung erfolgt byteweise. Nach dem erfolgten Schreibsignal wird der Sequenzer getriggert der die geschriebenen Bytes dann in die FIFO kopiert. Dabei wird dem CRC Generator noch mitgeteilt, dass jetzt Daten anliegen für die CRC Berechnung.
Ob ein Data Valid Signal mit in die TX FIFO geschrieben wird ist über ein Steuerregisterbit gesteuert.

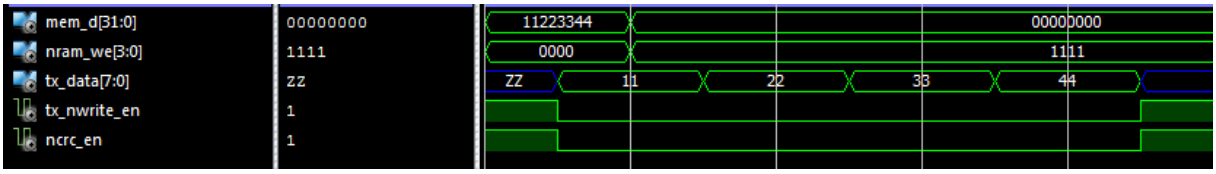

Abbildung 2 Schreiben eines Wortes

Abbildung 3 Schreiben eines Halbwortes

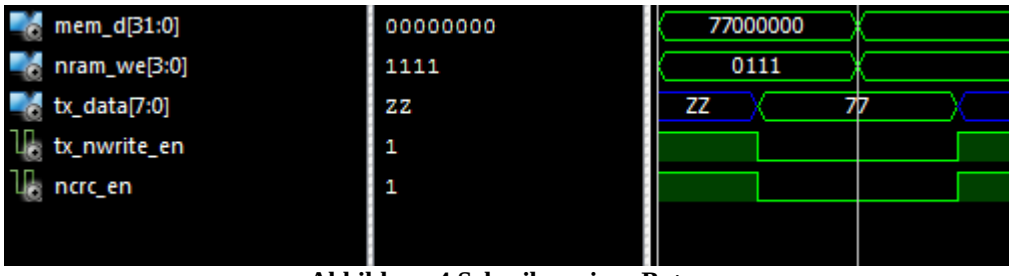

Abbildung 4 Schreiben eines Bytes

Serializer RX Pfad

Der RX Pfad ist unter anderem über das Data Valid Flag der RX FIFO gesteuert, nur wenn dieses HIGH ist werden auch Daten aus der FIFO gelesen. Zwischen 2 empfangen Paketen befindet sich immer ein FIFO Eintrag mit DV=LOW zur Trennung der Pakete.
Daher muss dieser Eintrag vor einem Lesevorgang herausgelesen werden ohne ein triggern des Sequenzers. Dazu existiert das RX_START Bit im Steuerregister. Einmal gesetzt wird für einen Systemtakt die FIFO ausgelesen und danach liegt das erste Byte des Pakets an.
Ist die FIFO Empty, so ist dieser Vorgang wirkungslos, daher ist das RX FIFO Empty Flag im Statusregister auslesbar.

Der RX Sequenzer wird über einen Lesevorgang des Datenregisters getriggert. Dies bedeutet, dass der erste Lesevorgang veraltete Daten enthält und als Fake- oder Triggerlesevorgang gesehen werden muss. Mit dem nächsten Lesevorgang sind dann die ausgelesenen Daten verfügbar.

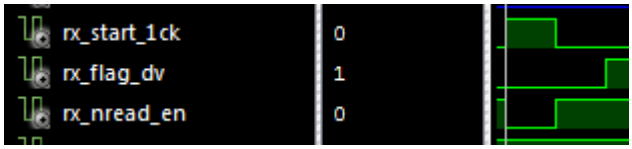

Abbildung 5 Start des RX FIFO Lesevorgangs

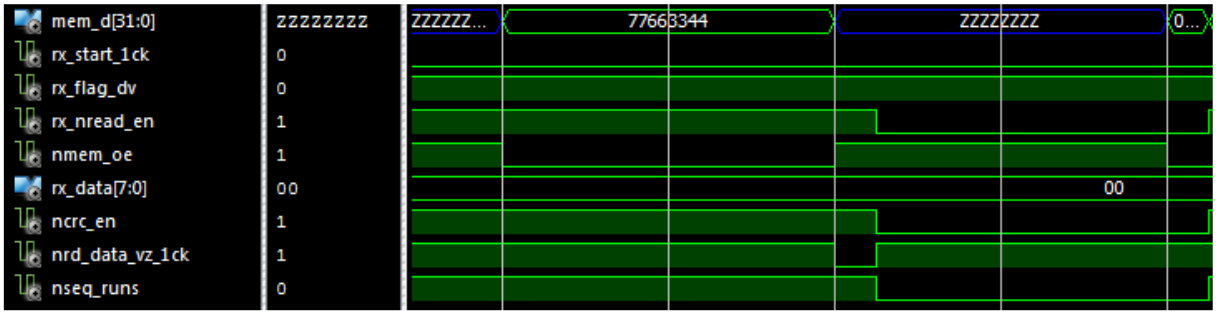

Abbildung 6 Fakeread mit veralteten Daten auf mem_d

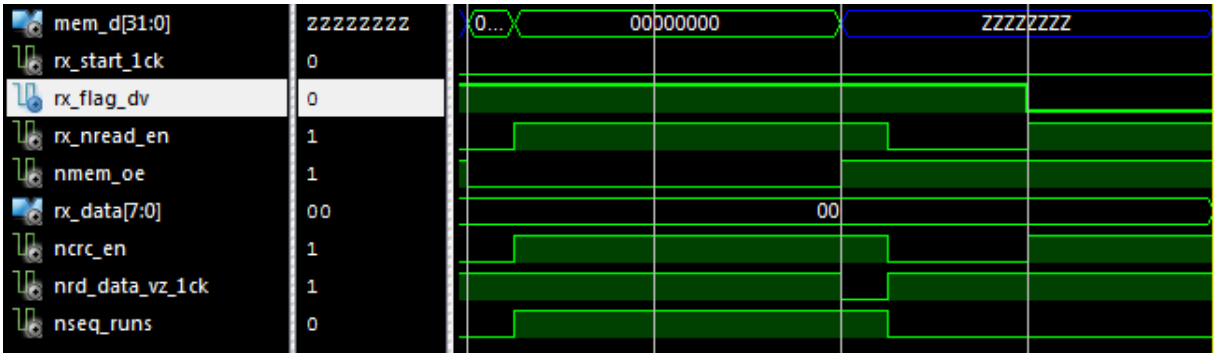

Abbildung 7 Auslesen echter Daten und Paketende (DV)

RX Pfad

Der RX Pfad ließt die Daten aus dem PHY aus und fasst die 4Bit Nibbles des MII kontinuierlich zu Bytes zusammen. Diese Bytes werden nach einem DataValid Low->High auf das SFD untersucht. Dafür sorgt ein Vergleicher, der auf Data Valid und dem Byte des SFD vergleicht. Ist diese Bedingung erfüllt so wird das Data Valid Signal um 1 Takt verzögert weitergereicht. Es erfolgt eine Verzögerung um 1 RX CLK damit das SFD selbst nicht in die FIFO gelangt. Diese Verzögerung wird durch das Nibble -> Byte FlipFlop selbst verursacht.

Ein Zähler zählt die Nibbles um später der CPU ein schnelleres Auslesen des Pakets zu ermöglichen. Denn so muss kein Statusbit gepollt werden, sondern der Lesevorgang erfolgt direkt über eine hochgezählte Variable im Register. Dies beschleunigt den Zugriff um 100%! Um nur eine 9 Bit FIFO nutzen zu müssen wird die Anzahl an Wörtern (=32Bit) gezählt und in der gesonderten FIFO für die Paketgröße gespeichert. Dies stellt keinen Mangel dar, denn der Lesevorgang der CPU kann auch nur Wortweise erfolgen (was für eine Beschleunigung um 400% sorgt). Der Zähler dient gleichzeitig als /2 Teiler für das Schreibsignal der FIFO, denn nur nach 2 Nibbles ist ein Byte verfügbar.

Das Daten FIFO Schreibsignal wird Initial um 1 Takt verzögert, denn sonst würde ein nulltes Byte geschrieben werden. Dieses verzögerungs- FF sorgt gleichzeitig für die H->L Flankenerkennung des Data Valid Signals um das Paketende zu erkennen.
Dieses Paketende Event triggert weitere Vorgänge:

- ein Data Valid = LOW Byte in die FIFO schreiben
- die Paketgröße wird in die FIFO für die Paketgröße gespeichert
- der IRQ wird aktiviert
- der Paketzähler wird erhöht

Der Paketzähler trägt über die Taktdomäne wie viele Pakete empfangen wurden. Die LAN Karte hat einen weiteren Zähler für die gelesenen Pakete. Ist dieser Zählerstand ungleich so sind noch weitere Pakete in der RX FIFO.

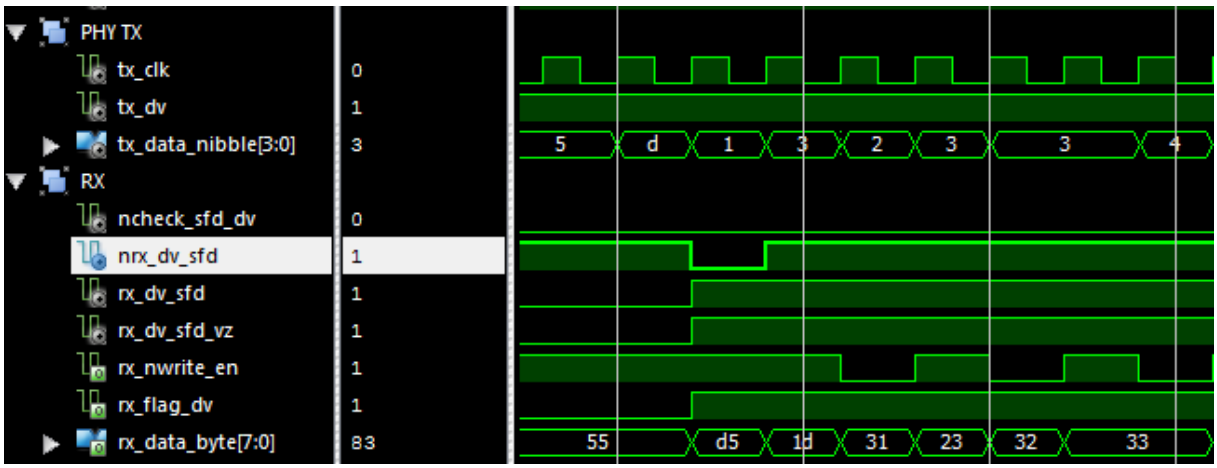

Abbildung 8 SFD Erkennung und FIFO Schreiben

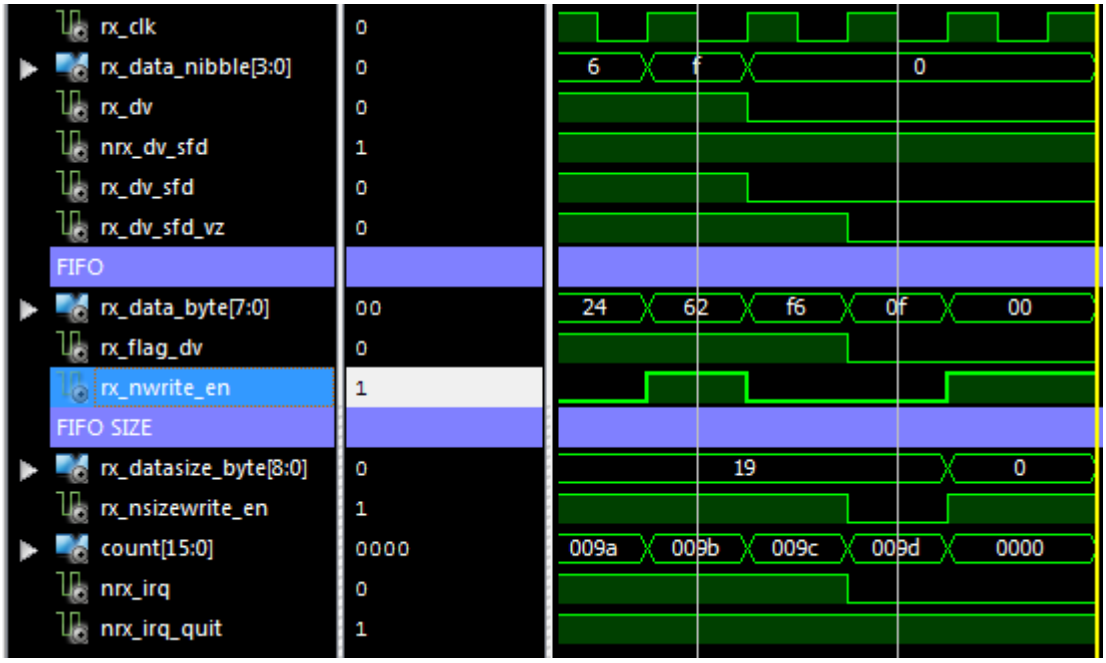

Abbildung 9 Ende eines Paketes

TX Pfad

Der TX Pfad ließt die Pakete aus der FIFO aus und sendet diese zum PHY. Vorher werden noch Präambel und SFD gesendet, dann das Paket und danach muss die Interpaketpause eingehalten werden.

Daher wird hier eine Statemachine benötigt. Diese besteht aus einem Zähler und einem Decoder. Daher sind die Zustände „one hot“ codiert. Diese Statemachine wird durch ein „TX EN“ Bit aus dem Register aktiviert und die Statemachine hält sich selbst aktiv bis das Paket gesendet wurde.

Allerdings darf die Statemachine nicht Takt für Takt durchlaufen, denn 3 der Zustände benötigen eine längere Verweildauer:

- Preambel : 15CK
- Paket: Länge in Bytes *2 CK
- Interpaketlücke 24CK

Die festen Haltezeiten werden durch Zähler verwirklicht und deaktivieren den Statemachinezähler. Die Paketlängeanhaltezeit wird aus dem Data Valid Signal der FIFO gewonnen. Solange dies HIGH ist wird der Zähler gestoppt.

Dieser Zählerstopper ist ein großes UND Gatter. Ein aktiver Zustand ist one hot (LOW) codiert und daher ist der LOW aktive Enableeingang des Zählers dann auch LOW. Bei den Zuständen mit einer Verweildauer ist dies noch ODER Verknüpft mit einem LOW aktiven Zustandssignal für „weiterschalten“.
So ist der Ausgang des Präambelzählers erst bei 255 LOW, sonst HIGH (anhaltend). Daher wird der Hochzähler mit 241 gesetzt und zählt dann bis 255 hoch und in diesem Takt ist der Ausgang LOW -> Die Statemachine kann weiterschalten.
Beim Paket wird das Data Valid Signal LOW nach dem letzten Byte.

Diese Zustände steuern dann den Datenfluss des TX Pfads über einen Multiplexer. Zum PHY muss das Nibble für die Präambel (0x5), den SFD (0xd) sowie die HIGH und LOW Nibbles.

Zustände und Aufgaben:

Idle)
Idlezustand, tut nichts,

FIFO Data Valid auslesen)
Das Data Valid = LOW Byte aus der FIFO lesen damit ein valides Byte anliegt.

Präambel senden)
Der Datenpfad ist auf 0x5 eingestellt und die Statemachine ist angehalten für 15 Takte

SFD senden)
Der Datenpfad ist auf 0xd eingestellt und die Statemachine ist nicht angehalten.

Paket senden)
Der Datenpfad toggelt jeden Takt zwischen HIGH und LOW Nibble hin und her. Nach dem Senden des HIGH Nibble wird ein neues Byte aus der FIFO gelesen. Die Statemachine ist solange angehalten wie valide Bytes aus der FIFO gelesen werden (Data Valid Flag)

Interpaketlücke abwarten)
Es muss für 24 Takte gewartet werden um der Gegenstelle eine Mindestzeit zu gewähren.

IRQ auslösen)
Dies ist ein Extra zustand, weil der Idle Übergangszustand mehrere Takte dauern kann.

Idle gehen)
Das TX En once Bit deaktivieren und warten bis der Idlezustand eintritt.

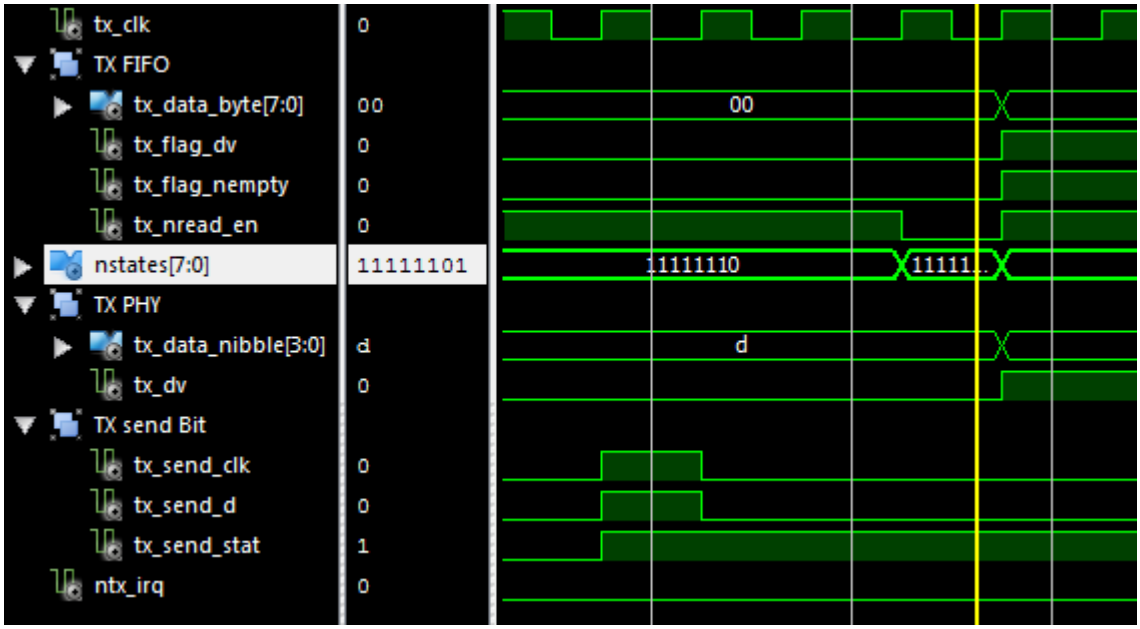

Abbildung 10 aktivieren des TX EN, erstes Byte aus FIFO lesen und Start der Präambel

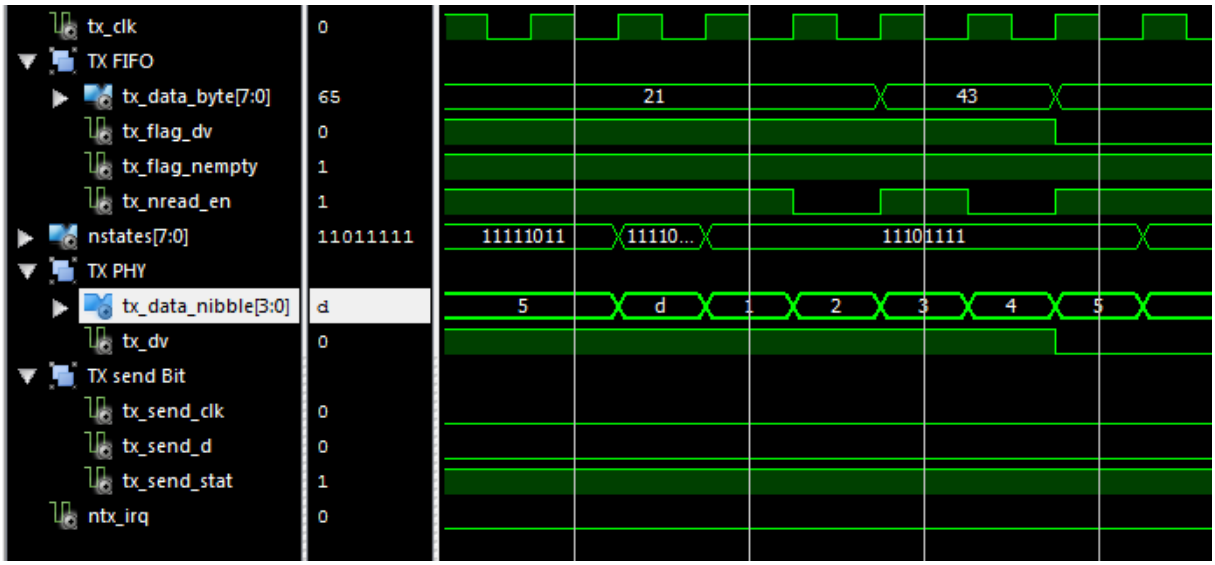

Abbildung 11 Präambel Ende, SFD senden, 2 Byte Paket senden, start des Gap

CRC Generator

Der CRC Generator sorgt für eine Hardwarebeschleunigung beim Empfangen und Senden von Ethernetframes.

Die Netzwerk CRC32 ist folgendermaßen definiert:
Startwert 0xFFFFFFFF
Endwert XOR: 0xFFFFFFFF
Polynom: 0x04C11DB7
Reflect In
Reflect Out

Reflect bedeutet, dass die Bitreihenfolge (nicht die Bytereihenfolge) umgedreht wird. Aus dem LSB wird das MSB.

Beim Senden wird die CRC32 über den Frameinhalt (ohne Preambel und SFD) berechnet und angehangen. Beim Empfangen wird die CRC32 über den Frameinhalt einschließlich der mitgesendeten CRC32 berechnet und dies ergibt unreflected das Magic Word 0xC704DD7B oder der empfangene Frame ist fehlerhaft.

Der CRC32 Generator läuft immer dann mit wenn Bytes aus der RX FIFO gelesen werden oder in die TX FIFO geschrieben werden. Dieser läuft also immer transparent mit. Damit dies funktioniert generiert der Sequenzer ein CRC Enable Signal, damit dieser nur dann rechnet wenn ein valides Datenbyte anliegt.
Der Startwert wird über einen Registerzugriff gesetzt und der Endwert wird über einen Registerzugriff gelesen. Der Endwert wird nicht automatisch an ein TX Frame rangehangen!

Um zu testen ob eine CRC Implementierung stimmt existiert die so genannte Checkvalue. Diese ist das Endergebnis wenn die CRC über die Bytes des Strings „123456789“ berechnet wurde.

In diesem Bild ist zu sehen wie die Payload „123456789“ per Netzwerkkarte versendet wird und die CRC32 die passende Checkvalue berechnet:

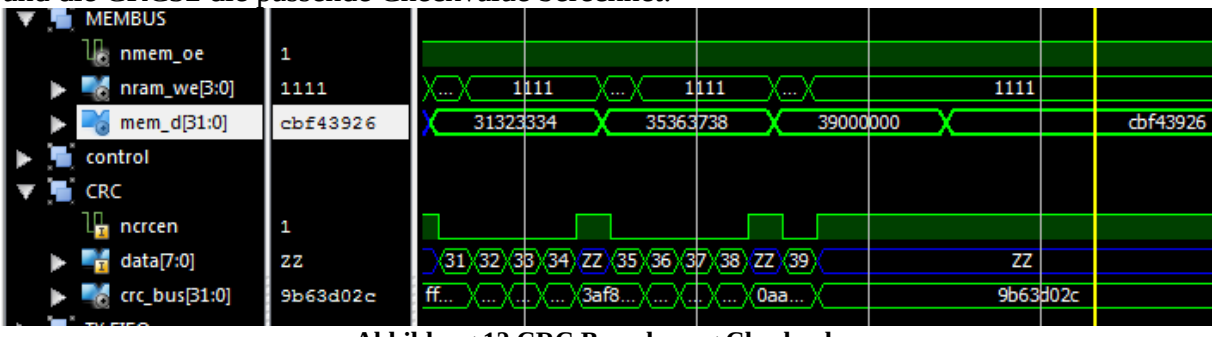

Abbildung 12 CRC Berechnung Checkvalue

Das nicht reflektierte Endergebnis ist auf dem „crc_bus“ zu sehen.
Das Auslesen nach „mem_d“ reflektiert den Wert und dann ist es auch die passende Checkvalue von 0xCBF43926.
Der CRC32 Wert aus dem uint32_t Register muss noch in der Endianess gedreht werden, also es wird 0x2639F4CB zur TTL Lankarte zurückgeschrieben.

Beim Auslesen eines empfangenen Ethernetframes aus der RX FIFO läuft der CRC32 Generator wieder an und berechnet die CRC32 über den Inhalt und die empfangene CRC32. Am Ende erscheint das Magic Word auf dem unreflektierten crc_bus. In der Software muss sich daher ein reflektiertes Magic Word von 0x2144DF1C gemerkt werden.

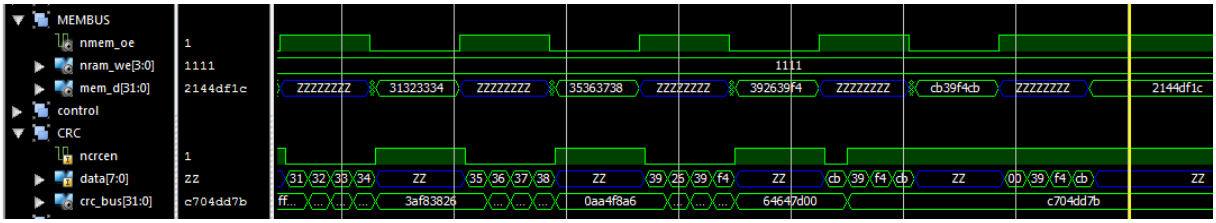

Abbildung 13 CRC Berechnung über empfangenes Frame

Die CRC32 Berechnung in Hardware stellt eine gewaltige Beschleunigung des Netzwerkverkehrs des MIPS TTL dar. Selbst ohne das ReflectIn würde ein CRC Zyklus 10 Befehle benötigen. Dabei benötigen ALU befehle 6 Takte und ein Paket kann bis zu 1500 Bytes haben.
1500*10*6 = 90000 Takte.
Bei 4MHz sind das 0,0225s -> 22,5ms!

Das Wortlesen statt Bytelesen ist eine Beschleunigung um den Faktor 4 und jedes LOAD braucht 8 Takte, speichern sind 6 Takte.
Pro Wort sind das also 8+6 Takte + etwas Schleifenabfrage des memcpy.
(1500/4)*(8+6) = 5250 Takte. Das sind 1,3ms.

Der Hardware CRC32 beschleunigt die Kommunikation daher um den Faktor 17!

Durch diese Beschleunigung wurde sich dazu entschieden den CRC Generator auch für andere Polynome und Bitbreiten zu ertüchtigen. Am OTP ROM sind noch Adressleitungen frei für die Auswahl weiterer Polynome. ReflectIn und ReflectOut sind per Controlregister de-/aktivierbar. Die Schiebeweite für CRC32, 16 und 8 ist auch im Controlregister steuerbar.

Registerbeschreibung

Registerübersicht

Offset	R/W	Zugriff	Inhalt	Beschreibung
0x00	R	32 Bit	Paketdaten RX	nur wortweise lesen!
0x00	W	8/16/32 Bit	Paketdaten TX	byte/hw/word schreiben erlaubt
0x04	R	32 Bit	RX Länge	9bit Wert als floored Words
0x08	RW	32 Bit	CRC Register	für CRC Initwert und auslesen
0x0C	W	32 Bit	CRC LUT SEL	Polynom der CRC Berechnung
0x10	W	32 Bit	CRC Control	CRC Einstellen
0x14	RW	32 Bit	Control	
0x18	R	32 Bit	Statusregister	

CRC Controlregister

Bit	RW	Name	interne Polarisierung	Beschreibung
0	W	Reset	nreset	Lankarte resetten nach Fehler
1	RW	CRC8 shift	nCRC8 shift	Auswahl des CRC Shifts nach Bitbreite, **nur eines darf aktiv sein!**
2	RW	CRC16 shift	nCRC16 shift	s.o.
3	RW	CRC32 shift	nCRC32 shift	s.o.
4	RW	ReflectIn		Bitreflektierung am Eingang
5	RW	ReflectOut		Bitreflektierung am Ausgang
6	RW	CRC_EN	nCRC_EN	CRC ist aktiv

Controlregister

Bit	RW	Name	interne Polarisierung	Beschreibung
0	W	RX_START		Das eine DV = LOW aus der RX FIFO "weglesen"
1	RW	TX_FLG_DV		DV in die TX FIFO schreiben und danach ein DV = LOW byte
2	RW	TX_EN	nTX_EN	TX Pfad ist aktiv
3	W	IRQ_RX_clear	nIRQ_RX_clear	RX Pfad IRQ quittieren
4	RW	IRQ_RX_Enable		RX IRQ aktivieren
5	W	RX nextlen		RX Paketgöße FIFO nächstes Größe auslesen und RX Packetcounter++
6	RW	TX_SEND		1 reinschreiben sendet Paket, auslesen um zu checken ob Senden noch aktiv (resettet sich selbst) **niemals 0 reinschreiben während eines Sendevorgangs!**
7	W	IRQ_TX_clear	nIRQ_TX_clear	TX Pfad IRQ quittieren
8	RW	IRQ_TX_Enable		TX IRQ aktivieren

Statusregister

Bit	RW	Name	Beschreibung
0	R	nRX_FIFO_EMPTY	RX FIFO Empty Flag
1	R	RX_FIFO_FLG_DV	RX FIFO Data Valid Flag
2	R	RX FIFO Overrun	RX FIFO enthält korrupte Pakete (-> Karte resetten)

3	R	RX Packets available	Packetcounter RX != Packetcounter Bus -> Pakete in FIFO vorhanden
4	R	RX IRQ active	RX IRQ hat gefeuert
5	R	TX IRQ active	TX IRQ hat gefeuert
6	R	nTX_FIFO_FULL	TX FIFO vollgeschrieben

Paket senden

Die Vorgehensweise um ein Paket zu senden:

1)
CRC Generator einstellen, dazu ein 0xFFFFFFFF in das CRC Register schreiben, die passende Poly LUT auswählen. Dazu „CRC_EN“ und ReflectIn sowie Out aktivieren.

2)
TX Pfad aktivieren, dazu „TX_FLG_DV“ und „TX_EN“ im Controlregister setzen.

3)
Paket wortweise (32Bit) in das Paketdaten TX Register schreiben.

4)
Abschließend (wenn vorhanden) das abschließende Byte und/oder Halbwort in das Paketdaten TX Register schreiben.

5)
CRC aus dem CRC Register auslesen, negieren, endianess drehen und als Wort in das Paketdaten TX Register schreiben.

6)
„TX_FLG_DV“ im Controlregister löschen und ein Byte 0x00 in das Paketdaten TX Register schreiben.

7)
„TX_EN“ im Controlregister löschen.

8)
„TX_SEND“ im Controlregister setzen. Jetzt wird das Paket gesendet. Der Sendevorgang ist abgeschlossen wenn „TX_SEND“ wieder gelöscht ist und wenn optional der TX IRQ ausgelöst wurde.

Paket empfangen

Die Vorgehensweise um ein Paket zu empfangen:

1)
Herausfinden ob ein Paket empfangen wurde. Dies geschieht über das Polling von „RX Packets available“ im Statusregister oder wenn der RX IRQ ausgelöst wurde.

2)
TX Pfad deaktivieren durch löschen von „TX_FLG_DV“ und „TX_EN“ im Controlregister.

3)
RX Pfad aktivieren durch setzen von „RX_START“ und „RX nextlen“ im Controlregister. Dies ließt den Pakettrenner (DV = 0) aus der RX FIFO und ließt die Paketgröße aus der RX Size FIFO.

4)
Paketlänge auslesen aus dem „RX Länge“ Register. Das Register enthält 9 Bit und enthält die Länge des RX Paktes in Wörtern. Dies ist ein unsigned integer der RX Bytes /4, also nach n Wörtern können noch bis zu 3 weitere Bytes in der FIFO sein. Daher werden n+1 Wortzugriffe zum Auslesen benötigt!

4)
CRC Generator einstellen, dazu ein 0xFFFFFFFF in das CRC Register schreiben, die passende Poly LUT auswählen und ReflectIn sowie Out aktivieren. Wichtig: „CRC_EN“ deaktivieren, dieses Signal wird nur beim Senden benötigt.

5)
Sequenzer starten durch ein Wort lesen aus dem „Paketdaten RX“ Register und verwerfen. Jetzt werden die ersten Bytes aus der RX FIFO gelesen.

6)
RX Daten wortweise aus dem „Paketdaten RX“ Register lesen (n+1).

7)
CRC Wert auslesen, negieren und mit 0x2144DF1C vergleichen.
Wenn ungleich, dann ist das Paket defekt -> verwerfen.
(Oder nicht negiert vergleichen mit 0xDEBB20E3)

8)
„RX Packets available“ im Statusregister auslesen ob weitere Pakete vorhanden sind. Wenn ja, dann goto 3).