ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA
SEDE DI CESENA

SECONDA FACOLTÀ DI INGEGNERIA CON SEDE A CESENA
CORSO DI LAUREA IN INGEGNERIA ELETTRONICA, INFORMATICA E TELECOMUNICAZIONI

TITOLO DELLA TESI

SVILUPPO DI UN MODULO VHDL PER INTERFACCIAMENTO ETHERNET – FPGA IN APPLICAZIONI DI HIGH THROUGHPUT SCREENING

Elaborato in

ELETTRONICA DEI SISTEMI DIGITALI

Relatore

Prof. Ing. ALDO ROMANI

Correlatore

Prof. Ing. FEDERICO THEI

Presentata da

ALESSANDRO RENZI

Sessione I
Anno Accademico 2011-2012

Indice

Capitolo 1

Introduzione

In molti settori della ricerca in campo biologico e biomedico si fa ricorso a tecniche di High Throughput Screening (d'ora in poi HTS), le quali consistono nell'esecuzione di moltissimi esperimenti in parallelo (reazioni chimiche di vario tipo e analisi dei risultati) possibilmente in modo automatizzato, provando il maggior numero di combinazioni possibili.
Questo tipo di approccio (che volendo fare un paragone con la sicurezza informatica si potrebbe definire "brute force") consente di identificare alcuni composti utili a precisi scopi.
Esistono inoltre alcuni ambiti in cui si studiano reazioni o fenomeni molto veloci, un esempio è lo studio dei canali ionici [Thei11].
In questo campo si studia la conduzione di ioni attraverso una membrana cellulare durante fenomeni che durano solo alcuni millisecondi.
Per farlo si usano dei sensori e dei convertitori A/D molto veloci che digitalizzano i dati per inviarli al PC attraverso una opportuna interfaccia.
Questa interfaccia deve necessariamente rispettare alcuni requisiti per poter rappresentare un'efficace interconnessione tra sistema di elaborazione e PC, in particolare deve garantire una elevata banda di trasmissione per poter trasferire senza perdite l'enorme mole di dati che tanti sensori affiancati possono generare, e dato che i fenomeni studiati sono molto veloci deve anche garantire una ridotta latenza.
Solitamente per questo scopo viene usata l'interfaccia USB, la quale però soffre di una limitazione di banda ed una latenza non ottimale.
Un buon candidato per sopperire a queste mancanze è l'interfaccia Ethernet, su cui è possibile lavorare per migliorare le caratteristiche che ci interessano.
Con questa tesi si tenterà di implementare un modulo VHDL per la comunicazione Ethernet, individuando le possibili ottimizzazioni che è possibile fare per renderla una valida alternativa all'USB.

Capitolo 2

Ethernet

In questo capitolo verranno illustrate le caratteristiche dell'interfaccia Ethernet e dei protocolli che vengono sfruttati nelle comunicazioni di rete, indicando anche quali migliorie è possibile introdurre per raggiungere lo scopo prefisso.

2.1 Tecnologie

L'interfaccia Ethernet dispone di diverse tecnologie per la trasmissione di dati su cavo [ieee802.3], le quali si differenziano principalmente per il throughput di trasmissione ma non solo.
Le possibili bitrate sono: 10Mbps, 100Mbps, 1Gbps, 10Gbps, 40Gbps, 100Gbps.
Le bitrate fino ad 1Gbps possono essere implementate su cavo di rame CAT5 o CAT6 intestato con un connettore RJ45 maschio.
Per le bitrate superiori diventa obbligatoria la fibra ottica perchè il rumore e la capacità parassita del cavo ne impediscono l'utilizzo.

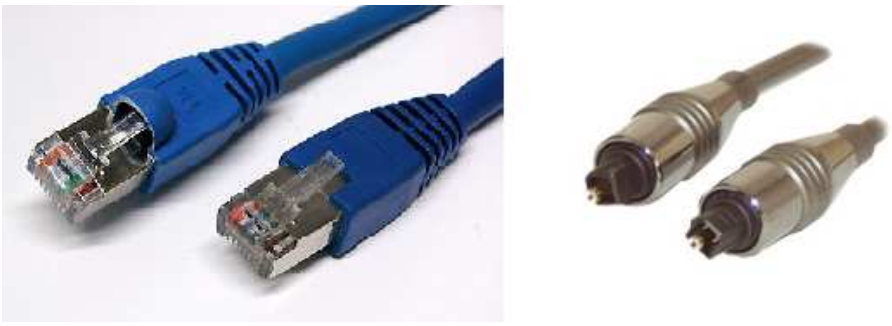


Figura 1: RJ45 e fibra ottica

Per raggiungere le bitrate più alte sulla fibra ottica si ricorre alla trasmissione multimodale, cioè si trasmettono contemporaneamente più colori. Questa tecnica è una vera e propria multiplazione a divisione di frequenza in quanto diversi colori corrispondono a diverse lunghezze d'onda della luce che possono coesistere sullo stesso canale e poi verranno opportunamente filtrate.
Per quanto riguarda la trasmissione a 100Mbps su rame in realtà vengono

sfruttati solo due doppini su quattro, mentre per quella a 1Gbps, sempre su rame, vengono usate tutte e quattro le coppie per raggiungere la massima velocità possibile.
Per entrambi la massima distanza permessa del cavo è di 100 metri, se la distanza che si deve comprire è maggiore è necessario inserire un rigeneratore oppure uno switch o un router.
Una volta per connettere insieme due host (due PC o comunque due terminali) era necessario incrociare un doppino sul connettore, ma adesso grazie alla massiccia adozione di hardware compatibile, non è più necessario in quanto nei nuovi connettori femmina (MAGJACK) sono presenti dei trasformatori.
Questi oltre a fornire isolamento dalla linea rendono indipendente dalla polarità dei fili.

2.2 Stack TCP/IP

Le comunicazioni di rete sfruttano diversi protocolli di comunicazione che sono organizzati seguendo un modello ben preciso, questo modello si chiama *internet protocol stack* [Kurose10] che letteralmente vuole dire "pila dei protocolli di internet", questo modello è anche conosciuto come *stack TCP/IP* ed è composto da 5 strati.

Lo stack TCP/IP si fonda sul principio che ogni strato possa dialogare solo con lo strato immediatamente superiore o inferiore.
Il livello di applicazione rappresenta una generica applicazione che vuole inviare o ricevere dei dati attraverso la rete, per farlo consegna i suoi dati allo strato di trasporto.
Lo strato di trasporto si occupa di ricevere i dati da tutte le applicazioni mantenendoli distinti, per farlo usa le porte, ogni porta è associata ad un flusso di informazioni di una applicazione (niente vieta ad una applicazione di aprire più porte e quindi più flussi contemporaneamente).

Per svolgere questo compito lo strato di trasporto si può servire di due protocolli, TCP e UDP.
Il protocollo TCP è un protocollo orientato alla connessione, cioè quando si vuole iniziare una trasmissione si deve stabilire una connessione con l'host remoto, il quale deve confermare di essere attivo e pronto alla ricezione, mentre quando la trasmissione è terminata la connessione viene chiusa e l'host remoto conferma la chiusura della connessione.
Questo meccanismo consente anche di garantire che tutti i pacchetti inviati arrivino a destinazione e vengano consegnati all'applicazione nello stesso ordine in cui sono stati inviati, questo è necessario perchè su internet ad un pacchetto possono succedere molte cose, ad esempio due pacchetti verso lo stesso destinatario possono prendere strade diverse e può capitare che un pacchetto arrivi prima del suo predecessore. Oppure un pacchetto può anche andare perduto per vari motivi.
Alla luce di questo si rende necessario un protocollo che tenga traccia dei pacchetti inviati e riceva conferma dell'avvenuta ricezione.
Questo è proprio quello che il protocollo TCP si propone di fare.
Questi meccanismi però richiedono una gestione più complessa e dispendiosa in termini di risorse.
Se invece non è richiesta garanzia di ricezione e non importa che i pacchetti arrivino in ordine casuale allora si può ricorrere al protocollo UDP.
Il protocollo UDP è definito *connectionless* cioè "senza connessione", questo perchè a differenza del protocollo TCP non richiede alcuna connessione preliminare e quindi non fornisce tutte le garanzie di quest'ultimo, i pacchetti vengono semplicemente inviati al destinatario.
Il vantaggio del protocollo UDP è che risulta molto più semplice da gestire e la sua intestazione è molto più piccola, questo lo rende molto appetibile per il nostro scopo.
Dopo che lo strato di trasporto ha aggiunto ai dati dell'applicazione la sua intestazione consegna tutti questi dati allo strato di rete.
Questo ha il compito di distinguere ogni host presente su internet sulla base di un indirizzo univoco chiamato IP, come il protocollo.
L'insieme degli host con un indirizzo IP forma la *rete internet*.
Il protocollo IP è di tipo *best effort* cioè non garantisce l'arrivo dei pacchetti ma fa del suo meglio per consegnarli.
Un datagramma (pacchetto) IP viene instradato verso l'host di destinazione da aparati di rete detti router che grazie a delle tabelle interne sanno come raggiungere tutti gli host di internet.
A volte può capitare che un pacchetto sia troppo grande per un router, allora è possibile frammentare questo pacchetto in due più piccoli che rispettino le

dimensioni massime per quel router, questo è possibile grazie ad alcuni campi dell'intestazione IP che hanno lo scopo di supportare questa funzionalità.
Una volta che il datagramma IP è stato formato viene consegnato allo strato di collegamento che è quello che interagisce con il mezzo di comunicazione secondo le specifiche dello strato fisico.
Lo strato di collegamento dipende dalla tecnologia con cui si trasmette la trama, ad esempio per trasmettere su cavo si usa lo standard Ethernet, per le trasmissioni senza fili si usa un altro protocollo, per trasmettere su linea ADSL si usa il protocollo PPP, ecc.
Lo standard Ethernet (ieee 802.3) precedentemente presentato comprende lo strato di collegamento (protocollo MAC) e lo strato fisico, definendo i cavi e i connettori da usare per una trasmissione Ethernet.
Il protocollo MAC è quello che si occupa di dialogare con altri host presenti sulla stessa rete fisica, cioè connessi allo stesso cavo o allo stesso switch, indipendentemente dal fatto che questi appartengano a reti IP diverse.
Le trame MAC vengono identificate da degli indirizzi univoci detti *MAC address* diversi dagli indirizzi IP. Ogni scheda di rete ha un indirizzo univoco associato e registrato.

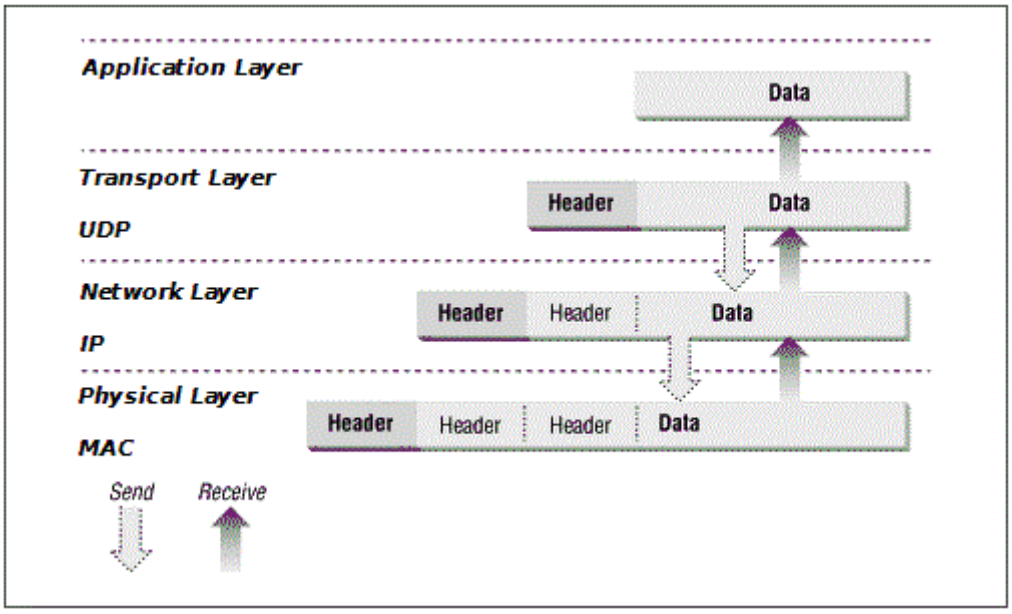


Figura 2: Internet Protocol Stack

2.3 Pacchetto UDP

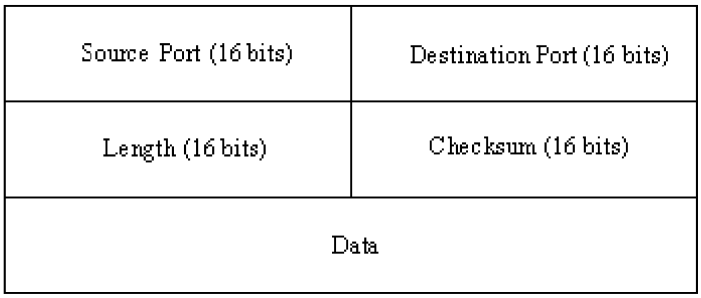


Figura 3: Intestazione UDP

Come si può vedere dalla figura 3 l'intestazione dei pacchetti UDP è molto semplice [RFC768], è composta da quattro campi per un totale di solo otto byte (l'intestazione del pacchetto TCP è lunga almeno venti):

- **Source port**: questo è il numero identificativo della porta usata dall'applicazione sull'host che trasmette il pacchetto, in questo modo chi riceve il pacchetto saprà su quale porta inviare la risposta.
- **Destination port**: questo è l'identificativo della porta di destinazione, in questo modo lo strato di trasporto dell'host ricevente saprà a quale applicazione consegnare i dati ricevuti.
- **Length**: in questo campo va inserita la lunghezza totale del pacchetto, quindi compreso l'header, espressa in numero di byte. In questo modo l'host ricevente sa quanto è lungo il pacchetto.
- **Checksum**: questo campo serve per fornire un controllo errori sul pacchetto. Consiste nel calcolare tramite un opportuno procedimento [RFC1624] un "riassunto" del pacchetto completo più una parte dell'intestazione IP [RFC768] in modo che il ricevente ricalcolandolo con lo stesso procedimento sui dati ricevuti possa rilevare eventuali errori. Questa operazione è molto dispendiosa in termini di tempo e risorse in quanto richiede che i dati vengano bufferizzati interamente in locale. Per fortuna è possibile disabilitare la validazione del checksum da parte del ricevente semplicemente scrivendo un checksum composto da tutti zeri, questa convenzione è possibile grazie al fatto che il checksum non può mai assumere tale valore [RFC768]. Quando un host riceve un pacchetto UDP il cui checksum è zero lo

ritiene automaticamente valido e quindi non perde tempo a calcolarlo, questo fa risparmiare qualche operazione anche sull'host ricevente riducendo ulteriormente la latenza.

- **Data**: qui vengono inseriti i dati da trasmettere, l'insieme di intestazione e dati costituisce il pacchetto UDP.

Dato che i campi delle porte sorgente e destinazione sono lunghi 16 bit ne risulta che il numero massimo di porte disponibili è 65536, però alcune di queste sono riservate o registrate [IANA].
In particolare le porte dalla 0 alla 1023 sono riservate a determinate applicazioni, ad esempio la porta 80 è riservata ai webserver, mentre le porte dalla 1024 alla 49151 sono registrate da vari enti o aziende per riservarsene l'uso.
Le porte dalla 49152 alla 65535 sono liberamente utilizzabili da qualunque applicazione.

2.4 Datagramma IP

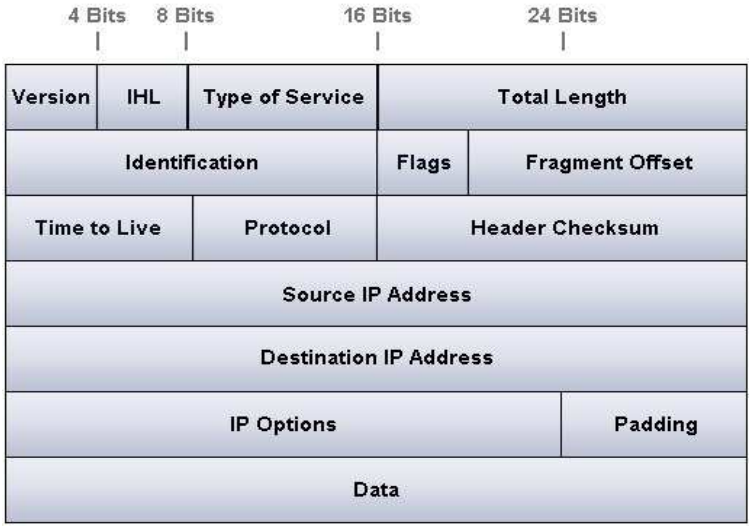


Figura 4: Intestazione IP

L'intestazione IP è formata da diversi campi per una lunghezza minima di venti byte [Kurose10], potrebbe essere anche più lunga nel caso fossero presenti opzioni aggiuntive, ma solitamente non lo sono.
I campi sono i seguenti:

- **Version:** indica la versione del protocollo IP usata, la più comune e diffusa è la versione quattro (questa è anche la versione che verrà

illustrata), ma ultimamente è iniziata la transizione alla versione sei per motivi che verranno illustrati successivamente.

- **IHL:** *internet header length*, questo campo indica la lunghezza dell'intestazione misurata in parole di 32 bit. Questa informazione è indispensabile in quanto la lunghezza del campo di opzioni è variabile e quindi si rende necessario un meccanismo poterla calcolare.
- **Type of service:** questo campo (che verrà illustrato in modo più approfondito in seguito) serve per indicare ai router se il pacchetto necessita di un trattamento privilegiato. Nel caso in cui i dati trasportati appartengano ad esempio ad un protocollo di videoconferenza, questi dovranno essere consegnati con la minore latenza possibile, se necessario anche a discapito di altri protocolli che non richiedono tale prontezza.
- **Total length:** lughezza totale del pacchetto misurata in byte, questa informazione, insieme a quella sulla lunghezza dell'header, permette di conoscere la lunghezza dei dati trasportati.
- **Identification:** nel caso in cui il pacchetto debba essere frammentato, questo campo identifica il pacchetto originale a cui appartiene il frammento.
- **Flags:** il campo flags è composto da tre bit, di cui il primo è sempre zero, il secondo indica se il pacchetto possa essere ulteriormente frammentato. Se il router necessita di frammentare ulteriormente il pacchetto per poterlo gestire, ma il secondo bit vale uno, allora il pacchetto verrà scartato. Sarà eventualmente compito degli strati superiori gestire la perdita di informazione. Il terzo bit indica se il pacchetto corrente è l'ultimo frammento di un pacchetto originariamente più grande, oppure se vi sono altri frammenti successivi.
- **Fragment offset:** questo pacchetto indica lo sfasamento del frammento corrente dall'inizio del pacchetto espresso in unità di otto byte. Quindi il frammento iniziale avrà sfasamento zero e un pacchetto potrà essere frammentato al più in unità di otto byte.
- **Time to live:** indica il tempo massimo di vita del pacchetto, dopo questo tempo il pacchetto verrà scartato, questo è necessario per evitare che pacchetti con un destinatario sconosciuto ai router possano continuare a girare in circolo per la rete e quindi rallentarla. Ogni volta che un pacchetto attraversa un router, questo decrementa il valore contenuto in questo campo di una quantità pari al tempo che è stato impiegato per processarlo. Il tempo viene espresso come un numero intero di secondi, quindi i tempi inferiori al secondo vengono

arrotondati ad uno. Dato che ormai tutti i router sono in grado di processare in modo continuo i pacchetti in un tempo di molto inferiore ad un secondo, questo campo è diventato una sorta di contatore dei router che il pacchetto attraversa.

- **Protocol:** il campo di protocollo indica quale protocollo di strato superiore è contenuto del campo dati, questo può servire per permettere ai router di trattare i vari protocolli in modo differente in modo da assicurare la massima qualità di servizio. La lista dei protocolli viene mantenuta da una organizzazione chiamata IANA.
- **Header checksum:** in questo campo è contenuto il controllo degli errori calcolato sulla sola intestazione, il metodo usato per il calcolo è analogo a quello dell'intestazione del pacchetto UDP. Il fatto che questo calcolo riguardi solo l'intestazione permette l'esecuzione di tale calcolo a priori, dato il destinatario, e quindi non richiede che i dati vengano bufferizzati in locale per essere processati. Questa è una caratteristica molto importante per minimizzare la latenza.
- **Source address:** qui viene inserito l'indirizzo IP sorgente, cioè l'indirizzo di chi manda il pacchetto, grazie a questa informazione chi riceve il pacchetto sa a chi inviare l'eventuale risposta. L'indirizzo IP è formato da quattro byte e il modo più comune per rappresentarlo è la notazione decimale, nella quale si indica il valore dei quattro byte in base decimale separati da un punto, ad esempio: 192.168.1.1. Dato che gli indirizzi IP hanno una lunghezza di 32 bit esiste un limite al numero di host che possono essere presenti contemporaneamente su internet. Questo limite è stato raggiunto e quindi si è reso necessario migrare alla nuova versione del protocollo IP: IPv6 che utilizza indirizzi a 128 bit. Questo permette una quantita tale di indirizzi che probabilmente non riuscirà ad essere saturata entro i prossimi secoli.
- **Destination address:** qui invece viene inserito l'indirizzo dell'host a cui si intende inviare il pacchetto.
- **IP option:** questo campo contiene eventuali opzioni che possono essere necessarie per estendere il protocollo IP, solitamente però queste non sono presenti.
- **Padding:** dato che la lunghezza dell'intestazione viene espressa in parole da 32 bit, se le opzioni non hanno una lunghezza multipla di questo valore è necessario aggiungere degli zeri per renderla tale, il campo padding rappresenta proprio questi eventuali zeri aggiuntivi.
- **Data:** qui vengono inseriti i dati provenienti dallo strato superiore.

2.5 Trama Ethernet

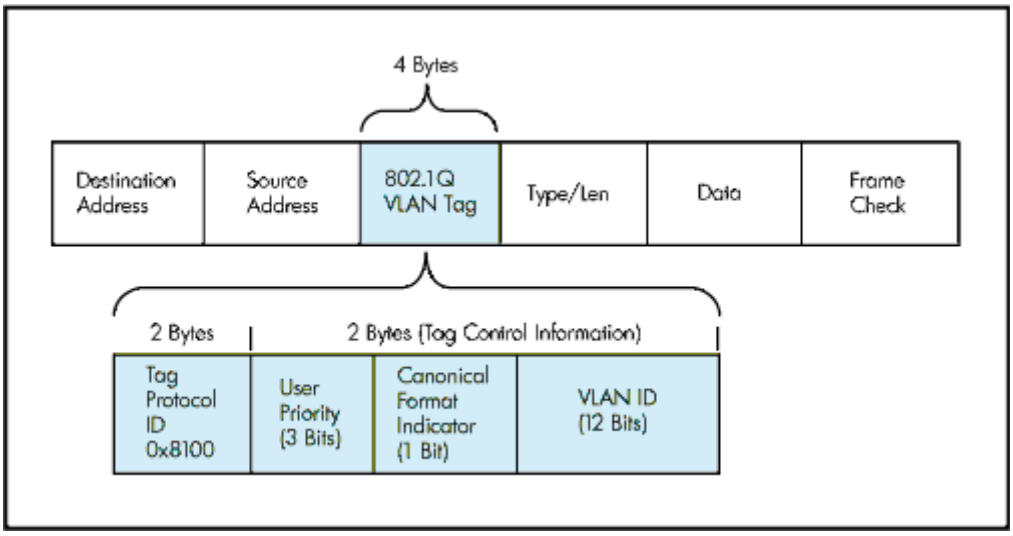


Figura 5: Trama Ethernet

L'intestazione della trama Ethernet è, come l'intestazione UDP, molto essenziale, dato che deve solo gestire la comunicazione al livello più basso:

- **Destination address:** questo è l'indirizzo MAC di destinazione, chi riceve la trama controlla questo campo, se corrisponde al suo indirizzo allora consegna i dati allo strato superiore.
- **Source address:** l'indirizzo MAC sorgente, come per l'intestazione IP, serve al ricevente per sapere a chi rispondere.
- **VLAN Tag:** questo è un campo opzionale che serve per implementare delle LAN virtuali, normalmente non è presente per collegamenti tradizionali.
- **Type:** qui viene indicato il protocollo di strato superiore che è presente nei dati trasportati dalla trama Ethernet. La lista dei protocolli viene mantenuta dall'IEEE.
- **Data:** i dati dello strato superiore vengono inseriti in questo campo. Lo standard impone una lunghezza massima dei dati di 1500 byte e una lunghezza minima di 46 byte, se i dati da trasmettere sono più corti allora si dovranno aggiungere degli zeri fino ad arrivare alla lunghezza minima.
- **Frame check:** qui viene inserito un codice di rilevazione degli errori CRC32 lungo 32 bit.

Capitolo 3

FPGA

Il modulo oggetto di questa tesi verrà sviluppato per essere usato su un FPGA (Field Programmable Gate Array). Un FPGA è un circuito integrato digitale pensato appositamente per poter essere riconfigurato anche dopo la sua costruzione, cioè per mezzo di opportune istruzioni può essere modificato per svolgere diverse funzioni definite dal progettista.
Dato che è l'hardware stesso ad essere modificato, gli FPGA sono potenzialmente molto più veloci e versatili dei microprocessori, in particolare questo è vero in ambiti dove si eseguono algoritmi altamente parallelizzabili, infatti gli FPGA vengono molto usati per eseguire elaborazioni parallele su dati proveniente ad esempio da schiere di sensori.
Il fatto che siano riconfigurabili li rende anche molto interessanti per le fasi di prototipazione di un integrato digitale, in quanto costruire un prototipo di un integrato è molto costoso e quindi il progetto deve essere ben collaudato.
La capacità degli FPGA di farsi riconfigurare è dovuta al fatto che sono composti da migliaia di blocchi logici elementari le cui interconnessioni sono modificabili tramite una quantità innumerevole di interruttori. Questi interruttori sono impostati secondo delle istruzioni (bitstream) che vengono caricate sull'FPGA ad ogni accensione da una memoria flash.
Il bitstream viene generato da un apposito ambiente di sviluppo che compila del codice di descrizione hardware che il progettista scrive.
Uno dei linguaggi di descrizione hardware più comuni è il VHDL (Very high speed integrated circuits Hardware Description Language) [Perry02].

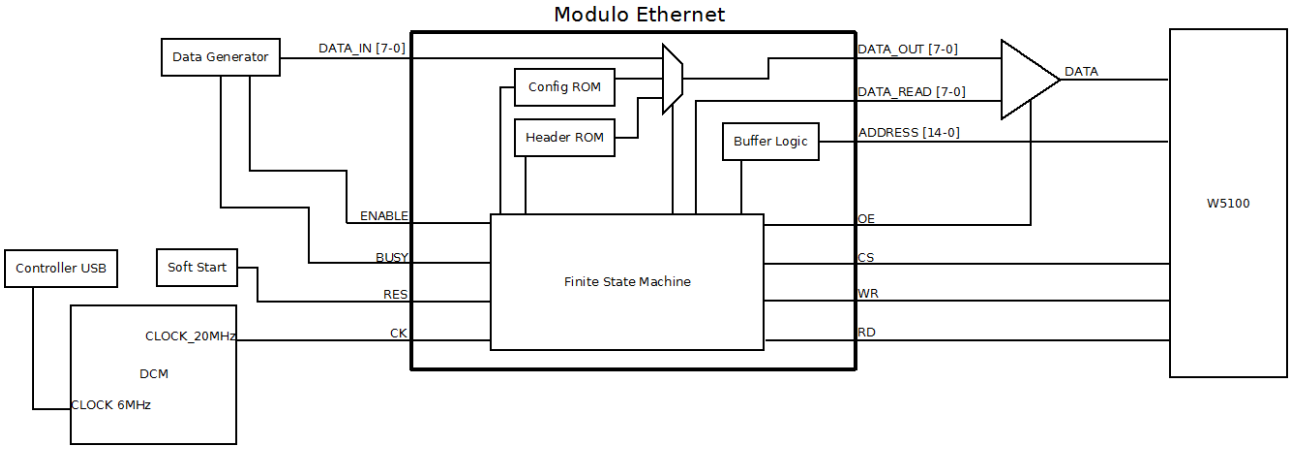


Figura 6: Schema Generale

3.1 Modulo DLP

Figura 7: Modulo DLP

Per eseguire i test è stato usato un FPGA Xilinx Spartan3 montato su un modulo apposito (fornito dalla ditta DLP Design) che integra tutto il necessario per gestire la programmazione dell'FPGA stesso e una RAM statica. La comunicazione con il mondo esterno è garantita da due file di strip montate nella parte inferiore connesse direttamente ai pin di I/O dell'FPGA.
La programmazione della memoria flash viene effettuata attraverso la porta USB mediante un apposito software fornito in dotazione. Questo software permette la cancellazione, la scrittura e la verifica della corretta scrittura del bitstream sulla memoria.
Il controller USB sulla scheda fornisce anche un clock di 6MHz all'FPGA.

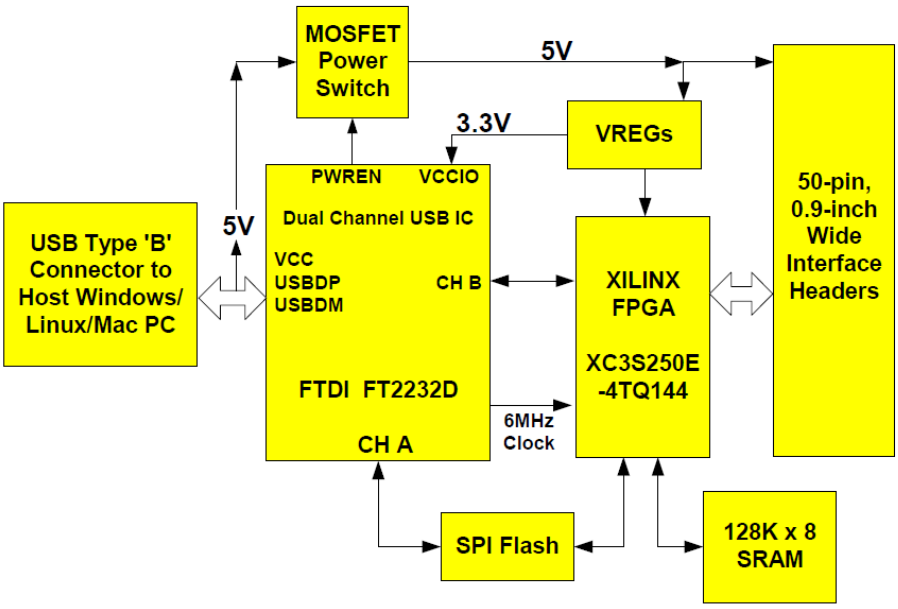


Figura 8: Schema modulo DLP

3.2 Memoria ROM e DCM

Per l'implementazione di questo modulo è stato necessario salvare alcuni dati in modo statico sull'FPGA, come ad esempio le intestazioni dei pacchetti. Questo è stato possibile grazie ad alcuni blocchi di memoria presenti sull'integrato che possono essere configurati a piacere sia come RAM che come ROM di lunghezza e larghezza arbitrarie.

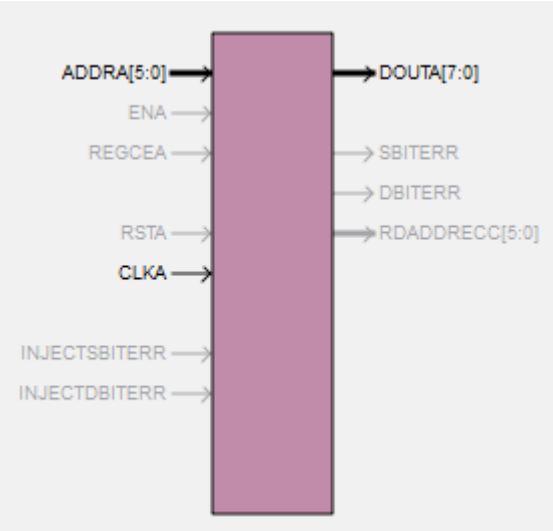


Figura 9: Blocco ROM

Questo elemento è istanziabile tramite una primitiva integrata nell'ambiente di sviluppo. I dati che saranno caricati nella ROM vengono forniti tramite un file con estensione *.coe* il quale deve avere una specifica formattazione illustrata nel datasheet della ROM.

L'FPGA però ha anche bisogno, per funzionare, di un clock, purtroppo quello fornito dal controller USB è troppo lento e quindi è stato necessario sfruttare un'altra funzionalità integrata, il DCM (Digital Clock Manager).
Il DCM fornisce funzionalità avanzate per la gestione del clock come l'eliminazione dello skew o lo sfasamento arbitrario del clock di multipli di novanta gradi. Inoltre è in grado di moltiplicare o dividere la frequenza per un certo fattore tramite un DLL (Delay Locked Loop).
Anche questo componente può essere istanziato tramite una primitiva dell'ambiente di sviluppo, inoltre dispone di un utile procedura guidata per la configurazione che, tra l'altro, permette di selezionare quali uscite si desidera utilizzare e impostare il fattore moltiplicativo da applicare al clock.

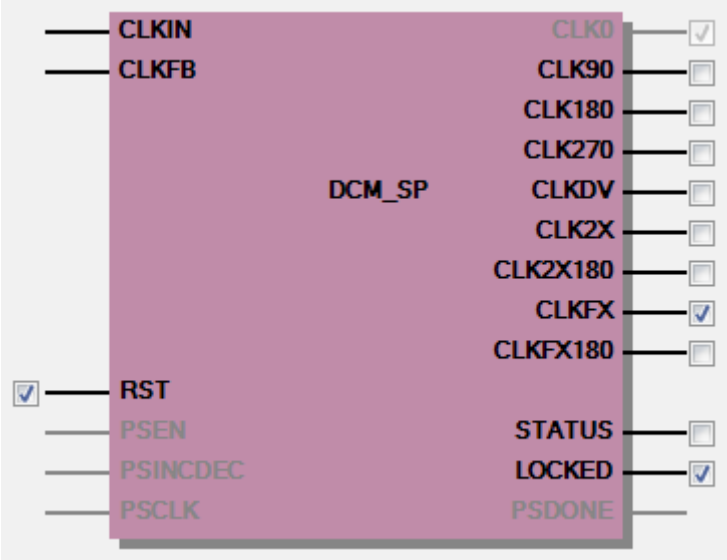


Figura 10: DCM

Per questo progetto è stato scelto di utilizzare un clock a 20 Mhz, quindi partendo dal clock a 6 Mhz del controller USB la procedura ha calcolato in automatico i fattori di divisione e moltiplicazione da applicare, in particolare divide per sei e moltiplica per venti.

Capitolo 4

Chip W5100

Dato che per interfacciarsi al mezzo di comunicazione (cavo) sono necessarie alcune funzionalità analogiche è stato necessario dotarsi di un chip che le implementasse, la figura seguente è abbastanza esplicativa di quante siano le funzioni che deve svolgere:

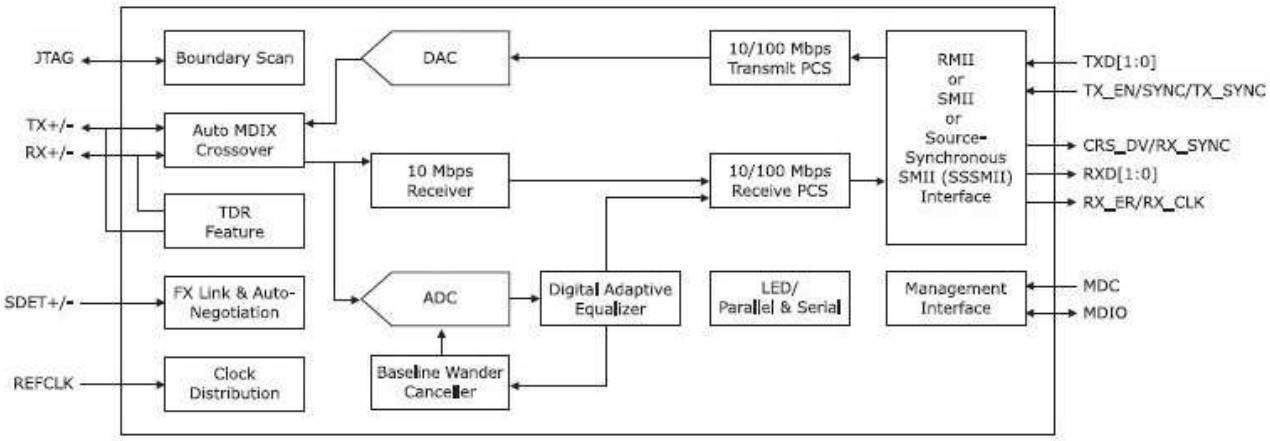


Figura 11: Schema PHY

Il chip scelto è il Wiznet W5100, questo non è solo un banale chip di strato fisico, ma un completo controller che è in grado di gestire i protocolli dello stack fino allo strato di trasporto.

4.1 Modulo Wiznet

Il chip è montato su un apposito modulo dotato di connettori a pettine nella parte inferiore, così può essere facilmente inserito e disinserito da un circuito.
Sul modulo è anche presente della circuiteria aggiuntiva per supportare le funzionalità del chip e un oscillatore al quarzo da 25Mhz.
Il modulo è inoltre dotato di un connettore MAGJACK che permette di non preoccuparsi di utilizzare un cavo incrociato per collegarsi ad un PC.

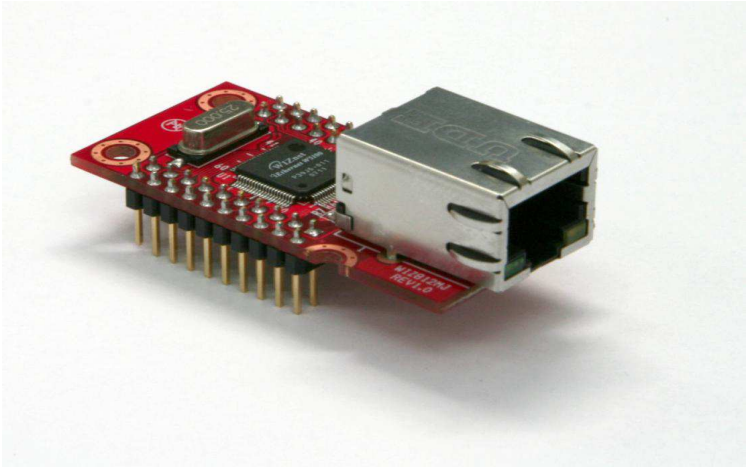


Figura 12: Modulo Wiznet

4.2 Modalità di funzionamento

Il chip contiene vari registri di configurazione che possono essere scritti in diversi modi.
Una possibilità è usare l'interfaccia SPI attraverso gli appositi pin.
In alternativa è possibile accedere ai registri indirizzandoli direttamente attraverso un apposito bus e scrivendo o leggendo il dato su un altro bus in modo molto simile ad una RAM.
Per il progetto è stato scelto il metodo dell'indirizzamento diretto che risulta il più semplice da gestire a scapito di un maggior numero di linee necessarie per usarlo. Lo schema è il seguente:

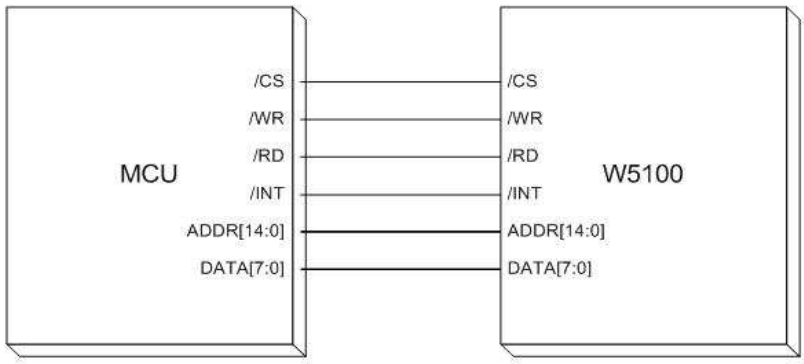


Figura 13: Interfaccia W5100

Questo integrato è molto interessante perchè non solo si occupa dello strato fisico, ma integra anche tutte le funzionalità necessarie a gestire il protocollo MAC, il protocollo IP, e i protocolli TCP e UDP ma non solo. Gestisce in automatico il protocollo ARP per la corrispondenza tra indirizzi IP e MAC e risponde ai ping (usato per controllare se un host è attivo) senza che sia

richiesto all'utente di fare niente. Questa è una funzionalità molto utile in quanto permette di verificare immediatamente se le configurazioni di base sono state scritte correttamente e se il chip è operativo.
Il chip inoltre integra al suo interno due buffer da 8KB, uno per la trasmissione e uno per la ricezione.

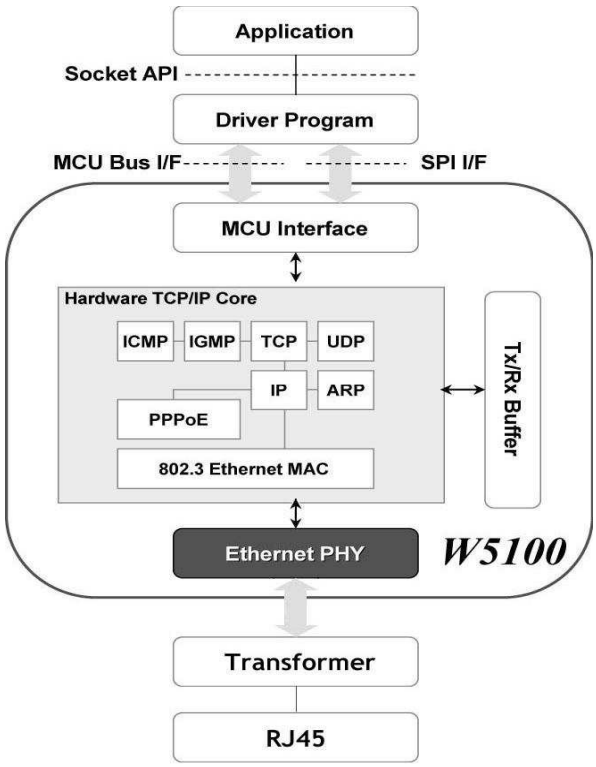
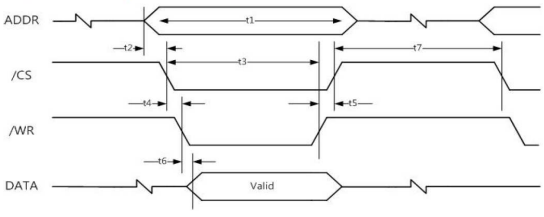


Figura 14: Schema a blocchi del W5100

4.3 Scrittura e lettura registri

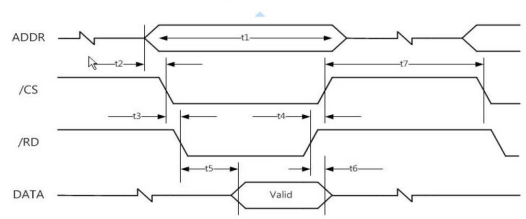
La scrittura e la lettura dei registri del chip W5100 per riuscire con successo devono rispettare delle determinate tempistiche che sono riassunte nelle seguenti tabelle estrapolate dal datasheet:

Register/Memory WRITE Timing



Symbol	Description	Min	Max
t1	Write Cycle Time	70 ns	-
t2	Valid Address to /CS low time	7 ns	-
t3	/CS low to /WR high time	70 ns	
t4	/CS low to /WR low time	0 ns	-
t5	/WR high to /CS high time	0 ns	-
t6	/WR low to Valid data time	-	7 ns
t7	/CS high to next /CS low time	32 ns	-

Register/Memory READ Timing



Symbol	Description	Min	Max
t1	Read Cycle Time	80 ns	-
t2	Valid Address to /CS low time	8 ns	-
t3	/CS low to /RD low time	0 ns	
t4	/RD high to /CS high time	0 ns	-
t5	/RD low to Valid data output time	48 ns	-
t6	/RD high to Data High-Z Output time	-	1 ns
t7	/CS high to next /CS low time	32 ns	-

Illustration 15: Tempistiche indirizzamento diretto

Le operazioni di scrittura e lettura dei registri sono state implementate con una macchina a stati, e quindi i tempi di attesa sono tarati per un clock a 20 Mhz.
Nel caso si volesse modificare il clock andrebbero aggiustati di conseguenza tutti i tempi di attesa. Un esempio di macchina a stati per la scrittura di un registro è il seguente:

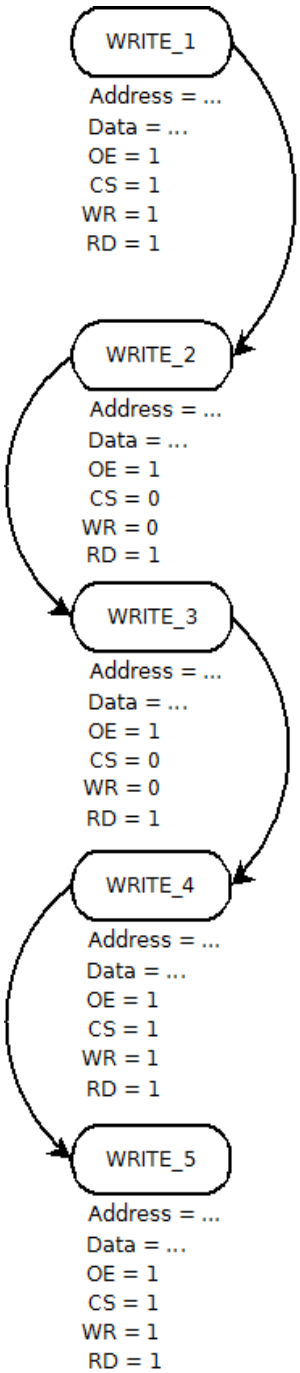


Figura 16: Scrittura su registro

Il segnale OE indica l'*output enable* per il buffer tristate di I/O.

Mentre per la lettura le operazioni sono queste:

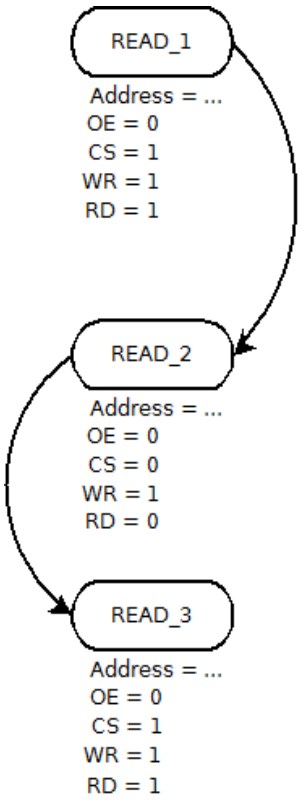


Figura 17: Lettura registro

4.4 Descrizione registri W5100

Il chip W5100 dispone di due tipi di registri: i registri di configurazione globale e registri dei socket. I socket rappresentano le connessioni che vengono aperte sul dispositivo da parte delle applicazioni, il chip supporta fino a quattro connessioni aperte simultaneamente.
La prima tipologia di registri rappresenta le configurazioni generali che sono comuni a tutte le connessioni, mentre la seconda tipologia rappresenta le configurazioni specifiche di ogni socket.
I registri globali sono i seguenti:

- **MR (Mode Register):** questo registro permette specificare opzioni aggiuntive riguardo all'indirizzamento, resettare il chip via software o abilitare le risposte al ping.
- **GWR (Gateway IP Address Register):** qui viene inserito il default gateway, cioè l'indirizzo del router della rete a cui l'integrato è connesso. Questo avrà il compito di consegnare tutti i pacchetti che

non sono indirizzati ad host della rete locale.

- **SUBR (Subnet Mask Register):** la *subnet mask* permette di distinguere l'indirizzo di rete dall'indirizzo dell'host.
- **SHAR (Source Hardware Address Register):** in questo registro è possibile scrivere l'indirizzo MAC sorgente associato al dispositivo. Gli indirizzi MAC dei dispositivi commerciali devono essere registrati presso l'IEEE.
- **SIPR (Source IP Address Register):** questo registro rappresenta l'indirizzo IP sorgente del chip, può essere un IP pubblico assegnato dall'ISP (Internet Service Provider) o un IP appartenente ad una rete privata.
- **IR (Interrupt Register):** questo è un registro di sola lettura che permette all'utente di conoscere la causa scatenante di un evento di interrupt, come ad esempio quale socket lo ha scatenato oppure se il chip non riesce a comunicare in rete (sintomo di un errore di configurazione).
- **IMR (Interrupt Mask Register):** qui è possibile mascherare alcuni eventi di interrupt, per farlo è sufficiente porre a zero il bit corrispondente all'interrupt che si desidera che non causi interruzione.
- **RTR (Retry Time-value Register):** qui viene impostato il tempo di *time out* che rappresenta il tempo massimo entro cui si attende una risposta, è usato prevalentemente per il protocollo TCP.
- **RCR (Retry Count Register):** questo valore rappresenta il numero massimo di tentativi di ritrasmissione di un pacchetto.
- **RMSR (RX Memory Size Register):** in questo registro viene impostato come suddividere la memoria del buffer di ricezione tra i quattro socket.
- **TMSR (TX Memory Size Register):** come per il precedente, il valore contenuto in questo registro descrive come deve essere suddivisa la memoria del buffer di trasmissione.

Ci sarebbero anche altri registri di configurazione generale, però questi non riguardano le operazioni che ci interessano.
Per questo progetto sarà necessaria solo una connessione, quindi verrà usato solo il socket 0, di cui verranno illustrati i registri di funzionamento:

- **S0_MR (Socket 0 Mode Register):** qui si imposta il protocollo che verrà usato dal socket per trasmettere o ricevere dati, e anche altre opzioni meno rilevanti.
- **S0_CR (Socket 0 Command Register):** questo registro è usato per

impartire comandi al chip come l'apertura di un socket o l'invio dei pacchetti bufferizzati.

- **S0_IR (Socket 0 Interrupt Register):** normalmente i bit di questo registro sono tutti a zero, se però si verifica un evento di interrupt relativo al socket, la causa scatenante viene segnalata ponendo ad uno il relativo bit.
- **S0_SR (Socket 0 Status Register):** il registro di stato è di sola lettura e serve per capire se il socket si è aperto correttamente nella modalità desiderata.
- **S0_PORT (Socket 0 Source Port Register):** quando il socket viene usato in modalità TCP o UDP la porta sorgente deve essere specificata in questo registro.
- **S0_DHAR (Socket 0 Destination Hardware Address Register):** è possibile anche specificare l'indirizzo MAC di destinazione scrivendolo in questo registro.
- **S0_DIPR (Socket 0 Destination IP Address Register):** l'indirizzo IP di destinazione deve essere scritto qui.
- **S0_DPORT (Socket 0 Destination Port Register):** qui deve essere scritta la porta di destinazione alla quale i dati sono indirizzati.
- **S0_TX_FSR (Socket 0 TX Free Size Register):** leggendo il valore di questo registro è possibile sapere quanto spazio libero è rimasto sul buffer di trasmissione.
- **S0_TX_RR (Socket 0 TX Read Pointer Register):** in questo registro è contenuto il primo indirizzo del buffer che verrà trasmesso dal socket.
- **S0_TX_WR (Socket 0 TX Write Pointer Register):** il valore contenuto in questo registro permette di calcolare l'indirizzo del buffer di trasmissione da cui iniziare a scrivere i dati da inviare.
  La procedura per determinare l'indirizzo è la seguente:
  si determina un valore denominato *offset* eseguendo l'AND bit a bit tra il valore del registro S0_TX_WR e un valore costante che dipende da come è stata configurata la ripartizione della memoria del buffer denominato TX_MASK.
  Dopodichè si calcola l'indirizzo sommando all'*offset* un altro valore costante anch'esso dipendente dalla configurazione della memoria chiamato TX_BASE.
  Occorre prestare attenzione al fatto che se viene raggiunto il limite superiore della memoria, allora i rimanenti byte dovranno essere scritti a partire dall'indirizzo TX_BASE.

Una volta terminata la scrittura il valore di S0_TX_WR dovrà essere incrementato di una quantità pari a quella dei byte scritti.
La seguente immagine illustra in modo abbastanza chiaro questa procedura:

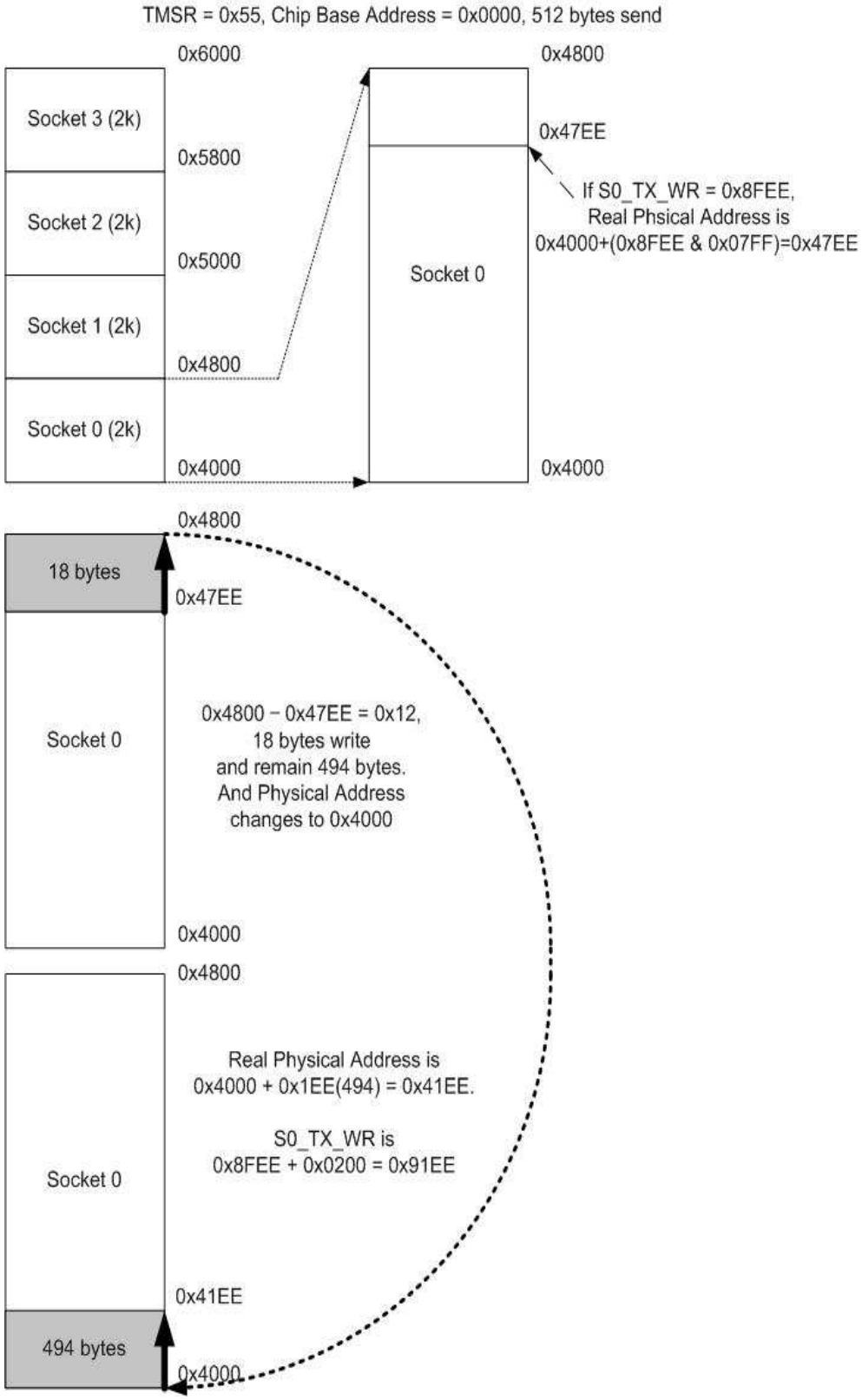


Figura 18: Scrittura dei dati sul buffer di trasmissione

4.5 Operatività

Il chip W5100 permette di essere usato in diverse modalità, quelle più utili al fine di questo progetto sono due: la modalità UDP e la modalità MACRAW.
Nella modalità UDP il chip si occupa di tutti gli strati e di tutti i protocolli dello stack permettendo di inviare direttamente dati al PC. Questa modalità è utile per verificare la correttezza di tutte le configurazioni preliminari, ed avere conferma che la procedura usata per scrivere i registri e quella per scrivere il buffer di trasmissione sono corrette.
La modalità MACRAW invece consente di usare il chip al più basso livello possibile, cioè lascia all'utente l'onere di formare i pacchetti UDP, IP e MAC e poi li invia direttamente.
Questo permette di verificare la correttezza delle intestazioni che verranno create.

Tutte le operazioni vengono eseguite da una macchina a stati finiti, di seguito verranno illustrate nel dettaglio tutte le operazioni necessarie al funzionamento commentando i relativi diagrammi a stati.
Lo schema fondamentale di funzionamento è il seguente:

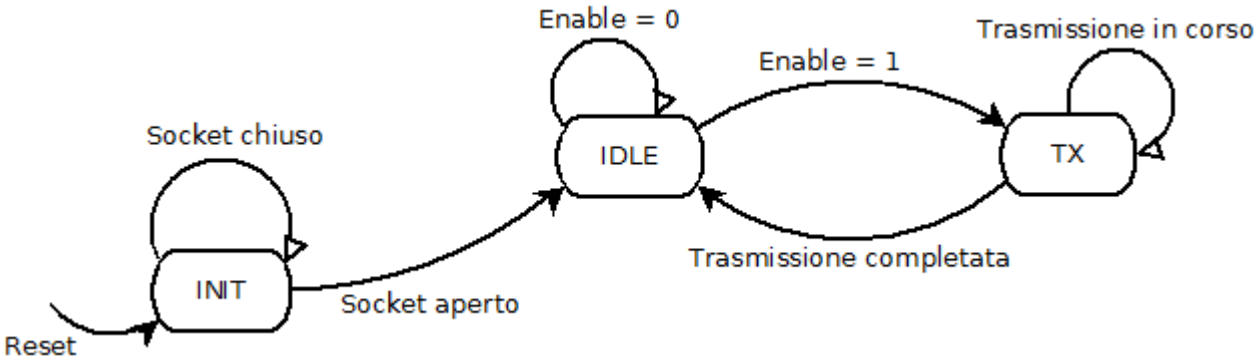


Figura 19: Schema fondamentale

Appena l'FPGA viene acceso inizia la fase di inizializzazione che consiste nel caricare tutte le configurazioni nei registri dell'integrato e lanciare il comando di apertura del socket.
Quando il socket si apre si entra in uno stato di attesa nel quale viene atteso l'enable con cui si inizia a ricevere i dati ed a caricarli nel buffer, una volta terminato il caricamento dei dati viene lanciato il comando di trasmissione.
Una volta che la trasmissione è conclusa si ritorna nello stato di attesa di nuovi dati da inviare.

4.6 Inizializzazione

La fase di inizializzazione ha il compito di leggere le configurazioni dalla ROM e caricarle nei registri del W5100.
Per controllare la ROM è stato necessario creare una logica di gestione, lo schema è il seguente:

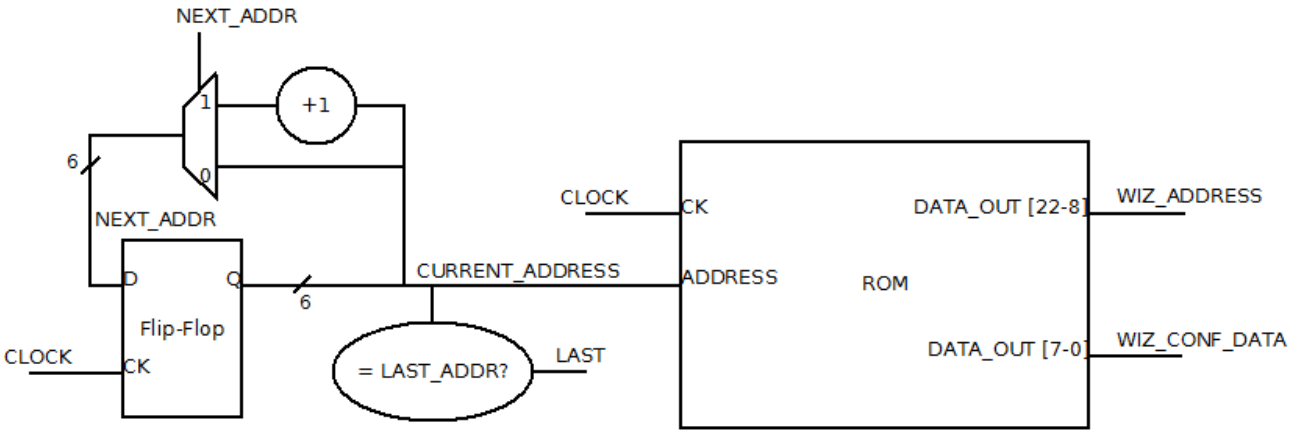


Figura 20: Logica gestione indirizzi ROM

Il flip-flop memorizza l'indirizzo corrente, quando si vuole incrementare l'indirizzo è sufficiente porre il segnale NEXT_ADDR ad uno per un ciclo di clock.
Questo contatore avanza finchè il confronto con la costante LAST_ADDR non da esito positivo, il che segnala il raggiungimento dell'ultimo indirizzo utile della ROM.
I dati nella ROM sono stati organizzati nel seguente modo: dato che gli indirizzi dei registri in cui scrivere non sono contigui allora era necessario associare a ciascun dato il suo indirizzo.
Questo è stato fatto semplicemente istanziando una ROM con una larghezza in bit dei dati contenuti in essa pari alla larghezza degli indirizzi più quella dei dati delle configurazioni, nello specifico gli indirizzi del chip sono larghi 15 bit e i dati 8 bit, quindi è stata impostata una larghezza complessiva di 23 bit. Così ogni parola da 23 bit contiene, nella parte alta l'indirizzo, e nella parte bassa il dato, come si può vedere anche dalla figura.
A questo punto la macchina a stati finiti non deve fare altro che ripetere le operazioni di scrittura prima illustrate fino a quando non viene asserito il segnale LAST.
L'ultima locazione della ROM contiene il comando di apertura del socket che andrà scritto nel registro S0_CR.

La macchina a stati finiti è la seguente:

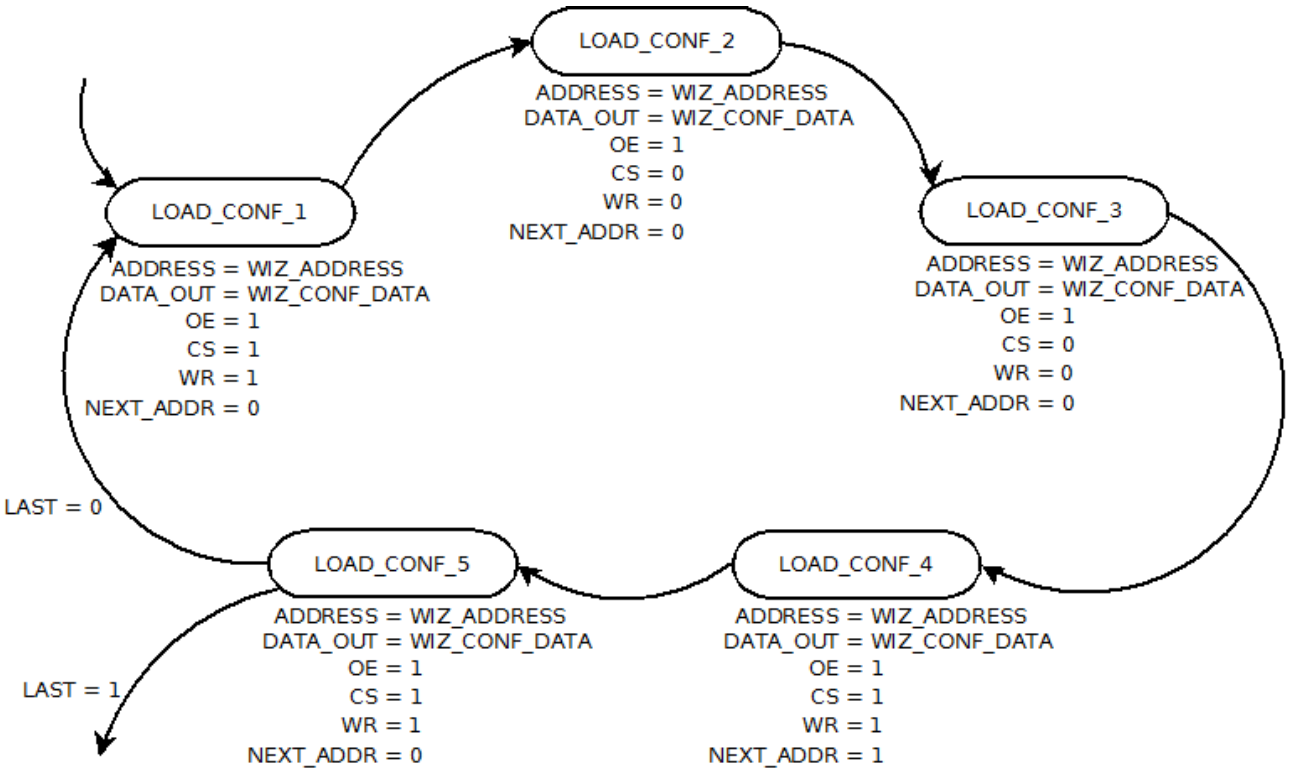


Figura 21: Macchina a stati finiti per la configurazione iniziale

Nelle figure delle macchine a stati finiti vengono indicati solo i segnali rilevanti, in realtà l'ambiente di sviluppo VHDL vuole che per ogni stato vengano specificati i valori di tutti i segnali.
Una volta che il segnale LAST informa che il caricamento delle configurazioni nei registri è terminato, si entra in un loop che legge continuamente il registro S0_SR, sempre con le motalità precedentemente spiegate, in attesa che indichi l'apertura nella modalità desiderata.
Per la modalità UDP deve assumere il valore 34, mentre per la modalità MACRAW deve assumere il valore 36.
Una volta confermata l'apertura del socket si entra nello stato di attesa.

4.7 IDLE

Lo stato di IDLE è lo stato in cui, dopo che il chip W5100 è stato inizializzato e configurato, si attende che l'utilizzatore richieda di trasmettere dei dati.
Per segnalare la disponibilità a trasmettere viene posto il segnale BUSY a zero, così l'utente è in grado conoscere la disponibilità a trasmettere del modulo.

Quando l'utente pone l'ingresso ENABLE a uno vuol dire che intende trasmettere dei dati, quindi si entra nello stato di trasmissione.
Lo stato di IDLE viene rappresentato così:

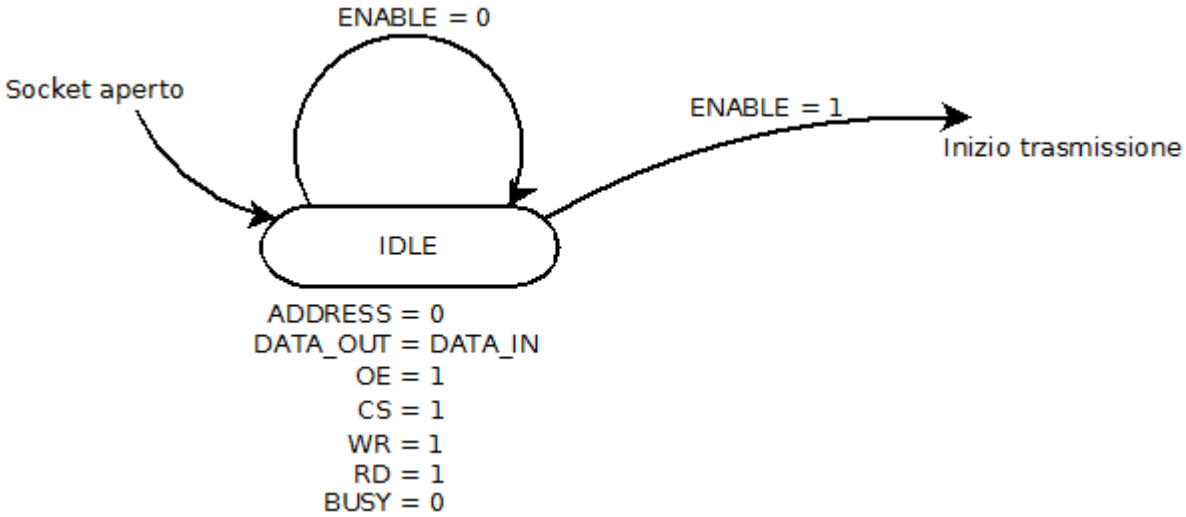


Figura 22: Stato di IDLE

4.8 Trasmissione UDP

Per la trasmissione dei pacchetti UDP generati dal chip le configurazioni sono le seguenti:

Mode Register: 0

Si vuole che il chip risponda ai ping, quindi non si abilità il ping block, non serve abilitare il protocollo PPPoE e non utilizziamo l'indirizzamento diretto ridotto, quindi tutte le opzioni del Mode Register rimangono a zero.

Gateway: 192.168.2.1

Nel caso di un collegamento punto-punto il gateway è una informazione inutile, quindi viene solitamente settato uguale all'indirizzo di uno dei due host che vengono collegati.

Subnet Mask: 255.255.255.0

La subnet mask è impostata sul valore di default per le reti private comuni.

Source Hardware Address: 02:02:02:02:02:02

L'indirizzo MAC in caso di applicazioni commerciali deve essere registrato all'ieee, per questa applicazione è possibile settarlo su un valore qualunque, purchè diverso dall'indirizzo MAC dell'host a cui ci si connette. Questo valore è stato scelto per essere facilmente individuabile.

Source IP Address: 192.168.2.10

L'indirizzo IP sorgente è stato scelto su una sotto rete diversa da quella delle altre connessioni di sistema del PC a cui ci si connette, in questo caso la 192.168.2.X, per evitare conflitti di indirizzi con gli altri host presenti sulla rete. L'indirizzo dell'host (10) è stato scelto fra uno dei 254 disponibili nella sottorete, l'importante è che sia diverso dall'indirizzo IP del PC a cui ci si connette.

Interrupt Mask: 0

Per adesso non si è interessati agli interrupt quindi vengono tutti mascherati.

RX Memory Size: 85

Con questo valore vengono assegnati 2KB di memoria di ricezione a ciascun socket, è stato scelto così perchè tutti gli esempi sul datasheet sono fatti basandosi su questo valore e non c'è una vera necessità che sia diverso.

TX Memory Size: 85

Per questo valore valgono le stesse considerazioni fatte per il registro precedente.

Socket 0 Mode Register: 2

Impostando il Mode Register con questo valore viene selezionata la modalità UDP per il socket 0.

Socket 1,2,3 Mode Register: 0

Gli altri socket vengono mantenuti chiusi scrivendo il valore zero nei rispettivi Mode Register

Socket 0 Source Port: 50000

Questo valore è stato scelto per essere facilmente individuabile nell'intervallo di porte che sono liberamente utilizzabili.

Socket 0 Destination IP Address: 192.168.2.1

Questo indirizzo è stato scelto nella stessa sotto rete dell'indirizzo del chip, altrimenti non potrebbero comunicare. Questo è l'indirizzo che dovrà essere assegnato alla scheda di rete del PC a cui ci si connette.

Socket 0 Destination Port: 50000

Anche la porta di destinazione, come quella sorgente, è stata scelta nell'intervallo delle porte liberamente utilizzabili. Il valore è stato scelto uguale sia per uniformità che per renderlo più facile da ricordare.

Socket 0 Destination Hardware Address: XX:XX:XX:XX:XX:XX

Questo è l'indirizzo MAC della scheda di rete del PC a cui ci si connette (dato che varia da PC a PC non ha senso inserire un valore specifico).

Socket 0 Command Register: 1

Questo valore impartisce al chip W5100 il comando di apertura del socket.

Questi sono i valori che verranno caricati nella ROM al fine configurare il chip W5100 per trasmettere pacchetti UDP.

Il procedimento precedentemente illustrato per ricavare l'indirizzo del buffer in cui scrivere i dati da inviare necessita di una logica di supporto, questa viene presentata nell'immagine seguente:

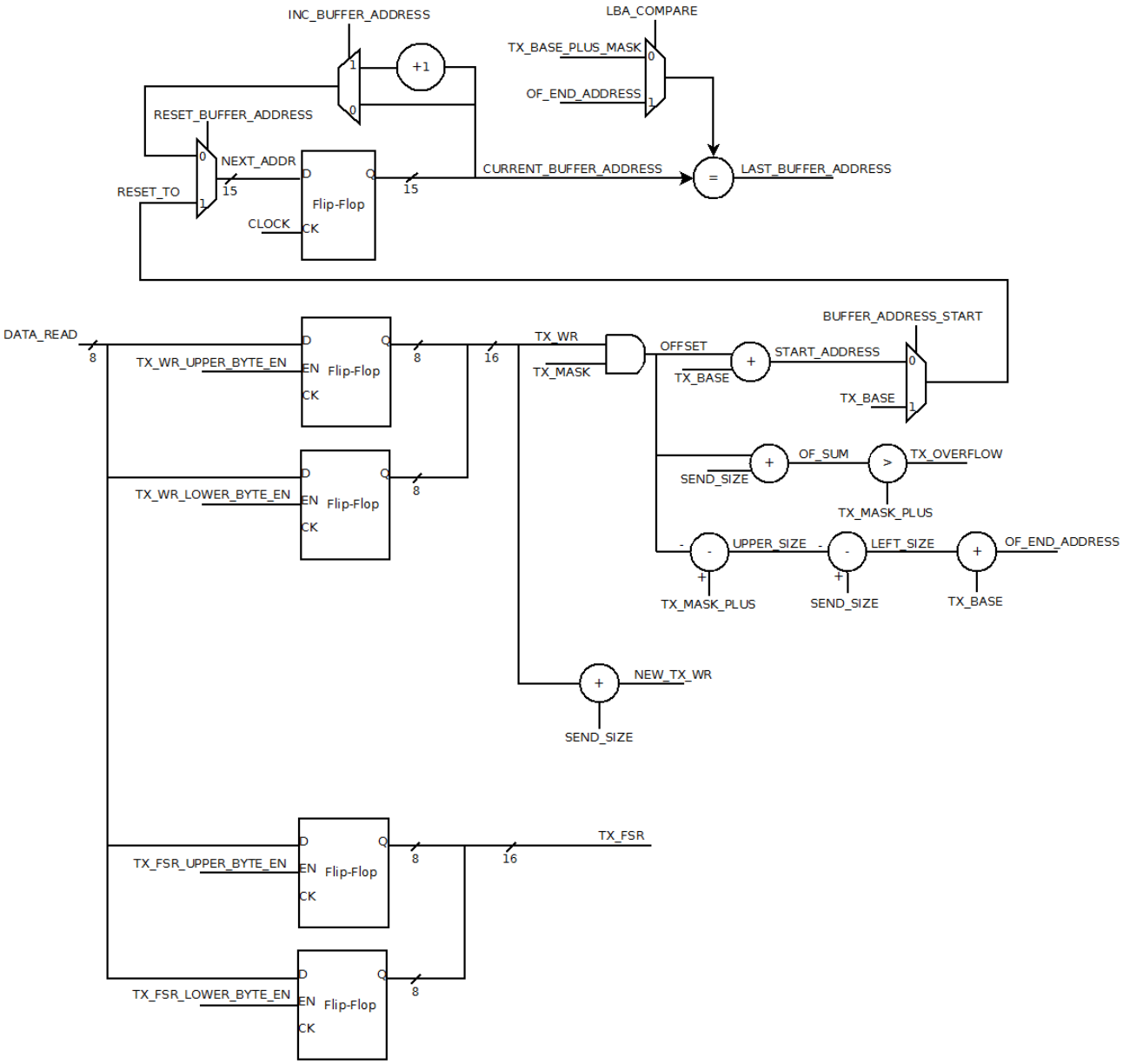


Figura 23: Logica indirizzi buffer

Per velocizzare i calcoli e ridurre le risorse occupate sono state create più costanti nelle quali sono stati inseriti tutti i valori ausiliari necessari. Per esempio in alcuni punti è necessario eseguire un calcolo con il valore di TX_MASK+1, in casi come questo si è preferito creare una ulteriore costante contenente il valore (TX_MASK_PLUS) piuttosto che inserire ulteriori sommatori.
Questa logica viene gestita dalla macchina a stati finiti, in particolare dagli stati

presenti nella sezione di trasmissione, questi stati sono rappresentati nell'illustrazione seguente:

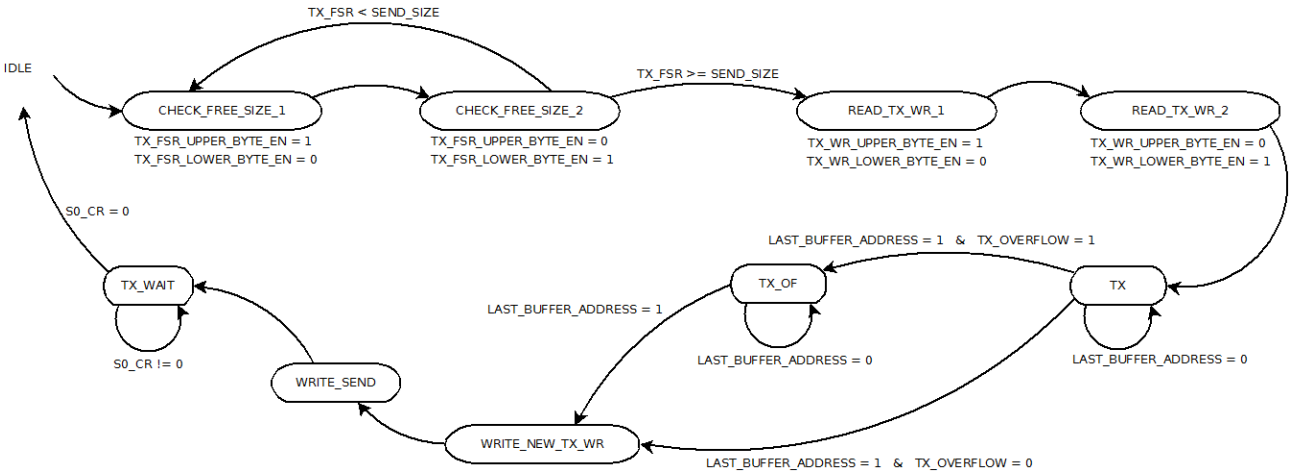


Figura 24: Stati della sezione di trasmissione

Per prima cosa viene controllato che nel buffer ci sia abbastanza spazio libero per la quantità di dati che si intende trasmettere, se lo spazio non basta allora il loop si ripete finchè non c'è abbastanza spazio.
Successivamente viene letto il registro TX_WR e campionato nei flip-flop, su questo valore vengono effettuati tutti i calcoli necessari.
A questo punto si entra in un loop che copia i dati da trasmettere nel buffer, gli stati per questa operazione verranno spiegati in seguito.
Se viene raggiunto il limite superiore del buffer (condizione espressa dal segnale TX_OVERFLOW) allora si entra nel loop che gestisce questa condizione (TX_OF), che riprende la scrittura dei rimanenti byte dall'inizio del buffer.
Terminata questa fase o se non viene raggiunto il limite del buffer si procede ad aggiornare il valore di TX_WR incrementandolo della quantità di byte che si intende trasmettere.
Dopodichè si lancia il comando di invio dei dati caricati sul buffer e si attende che questo sia terminato, quindi si ritorna nello stato di attesa (IDLE).
Il ciclo di caricamento dei dati da trasmettere nel buffer è illustrato nella figura seguente e inizia con uno stato di *setup* usato per resettare il contatore al valore corretto calcolato con l'apposita circuiteria.
I cinque stati seguenti sono uguali a quelli già visti per scrivere i registri del chip e fanno le stesse azioni. Le uniche differenze sono nel quarto stato, dove viene asserito per un ciclo di clock il segnale di incremento dell'indirizzo e nel quinto.
Nel quinto stato viene asserito per un ciclo di clock il segnale BUSY, per segnalare all'utente che il caricamento del dato è terminato, e che al prossimo

ciclo di clock dovrà presentare sul bus di ingresso DATA_IN il dato successivo.

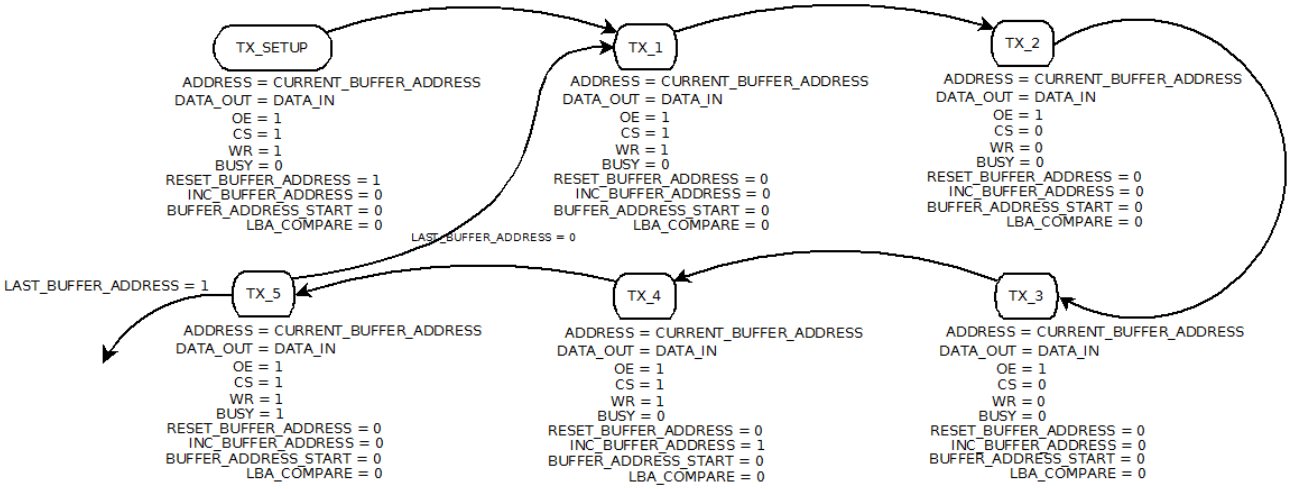


Figura 25: Caricamento buffer

Se il ciclo termina perchè il buffer è pieno allora tutti gli stati successivi dovranno porre il segnale BUSY ad uno. Quindi se l'utente vede il segnale BUSY ad uno per più di un ciclo di clock capisce che il buffer è pieno e quindi deve aspettare che venga terminata la trasmissione.
Quando il sistema tornerà nello stato di IDLE il segnale BUSY verrà posto nuovamente a zero per segnalare la disponibilità da parte del sistema ad inviare nuovi dati.

Se dovesse accadere un overflow del buffer, cioè gli indirizzi raggiungono il limite della memoria assegnata allora si dovrà entrare in un nuovo loop per gestire la situazione e terminare la scrittura dei dati. Questo loop è illustrato nella figura seguente:

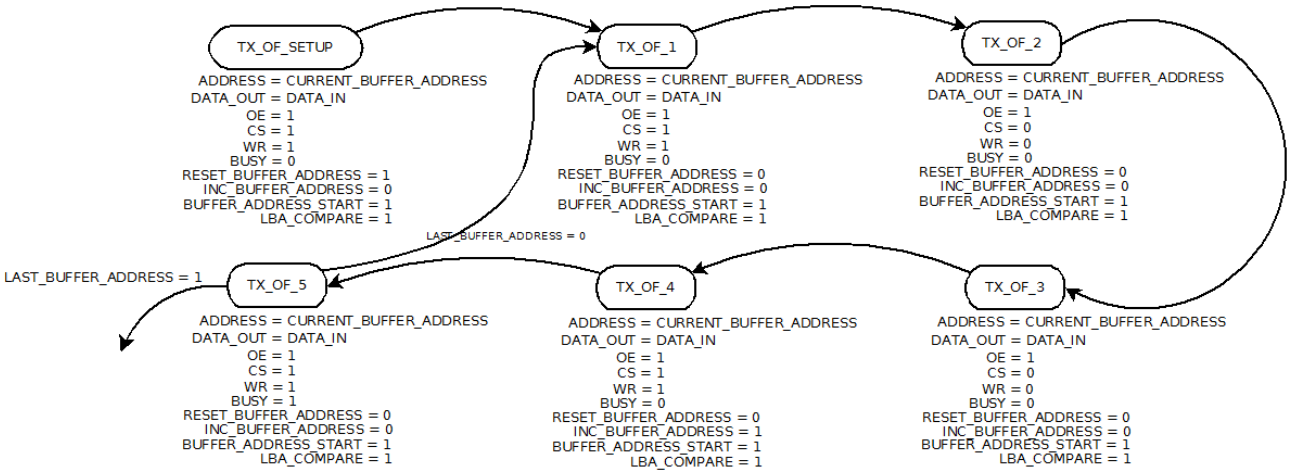


Figura 26: Cricamento buffer in condizione di overflow

Questo loop è sostanzialmente identico al precedente, l'unica differenza è che i

segnali BUFFER_ADDRESS_START e LBA_COMPARE sono posti ad uno, questi fanno in modo che il contatore, durante la fase di setup, venga resettato al valore corretto e che il segnale che avvisa il raggiungimento del termine del loop, venga asserito confrontando il valore dell'indirizzo attuale con il corretto riferimento imposto dalla condizione di overflow.

4.9 Trasmissione MACRAW

La modalità MACRAW permette di inviare dati grezzi, quindi sarà compito dell'utente creare le intestazioni e anteporle ai dati dei vari strati dello stack.
I valori scelti per queste intestazioni verranno ora presentate a partire da quella del pacchetto UDP:

Source Port: 50000

Per le porte sorgente e destinazione vengono mantenute le stesse configurazioni precedentemente illustrate per la trasmissione dei pacchetti UDP, in questo modo viene mantenuta uniformità di notazione.

Destination Port: 50000

Per questo valore valgono le considerazioni effettuate anche per il precedente.

Length: 1380

Questo campo contiene la lunghezza totale del pacchetto, quindi compresa l'intestazione. Questo valore è stato scelto in base a delle considerazioni che verranno illustrate successivamente.

Checksum: 0

Questo è il campo del codice per il rilevamento degli errori, come spiegato nel capitolo dedicato al protocollo UDP, questo calcolo può essere omesso imponendo il valore 0. In questo modo l'host che riceve il pacchetto non lo scarterà credendolo errato.
Il fatto di rinunciare al controllo degli errori potrebbe sembrare potenzialmente pericoloso per l'integrità dei dati, ma in realtà bastano alcune semplici considerazoni per comprendere come questo non sia un problema.
Normalmente il controllo degli errori è necessario, perchè un pacchetto che potenzialmente può arrivare al capo opposto del pianeta, attraverserà

una quantità considerevole di nodi di rete e verrà ritrasmesso tra un nodo e l'altro con tecnologie anche molto diverse tra di loro. Questo rappresenta un serio pericolo per l'integrità dei dati, senza considerare poi che il protocollo IP non garantisce l'assenza di errori.
A questo punto però è doveroso notare che lo scopo di questo modulo non è attraversare il pianeta, bensì comunicare con un PC connesso direttamente.
Inoltre le tecnologie dei chip di strato fisico hanno raggiunto una tale qualità da rendere la possibilità di errore davvero molto rara se non quasi impossibile, e anche se dovesse succedere che un bit venga corrotto può sempre essere rilevato dal controllo errori dello strato di collegamento.
Per questi motivi possiamo tranquillamente ignorare il checksum del protocollo UDP. Questa è una cosa molto positiva per lo scopo del progetto in quanto permette di creare a priori l'intestazione UDP e salvarla in modo statico in una ROM dell'FPGA. In questo modo è possibile annullare la latenza dovuta alla creazione dell'intestazione del protocollo.

Conclusa l'intestazione per il pacchetto UDP, verranno ora presentati i valori scelti per l'intestazione IP:

Version: 4

Come già spiegato, viene usata la versione quattro del protocollo IP, cioè quella più comunemente utilizzata e supportata.

Header Length: 5

Questo è il numero di parole di 32 bit che compongono l'intestazione, dato che non sono presenti opzioni aggiuntive l'intestazione ha la lunghezza minima di cinque parole.

Type of service: 56

Questo valore indica che è richiesta una bassa latenza ed una elevata banda di trasmissione [RFC791]. In realtà questo campo potrebbe anche essere ignorato perchè queste sono informazioni che vengono prese in considerazione dai router che inoltrano il pacchetto, quando il collegamento è punto-punto questa informazione perde di significato.
Per gli ultimi due bit di questo campo fare riferimento all'RFC 3168 pagina 6.

Total Length: 1400

Il motivo che ha portato a scegliere questo valore per la lunghezza totale

del pacchetto IP verrà, come per quello UDP, spiegata in seguito.

Identification: 0

Questo campo serve per supportare le funzionalità di frammentazione del protocollo IP, è evidente però che in un collegamento punto-punto come quello che si intende realizzare, dove è tutto noto, è possibile evitare a priori la frammentazione scegliendo opportunamente la dimensione del pacchetto inviato. Così si può evitare di spendere tempo e risorse per gestire questo campo a tutto vantaggio di una minore latenza.

Flags: 2

Questo valore indica che il pacchetto non deve essere ulteriormente frammentato (compito che sarebbe dell'eventuale router che però noi sappiamo già non essere presente), e che non sono presenti ulteriori frammenti di questo datagramma IP.

Fragment Offset: 0

Per le stesse considerazioni del campo *Identification* questo campo può essere tranquillamente lasciato a zero.

Time To Live: 20

Anche questo valore non ha molta importanza in quanto il chip è connesso direttamente all'host, quindi un qualunque numero maggiore di uno è corretto.

Protocol: 17

Questo valore indica che nei dati trasportati è presente un paccheto del protocollo UDP, il valore è stato scelto dalla tabella ufficiale gestita dalla IANA.

Header Checksum: 56289

Il checksum viene calcolato solo sull'intestazione, nel caso venga modificato qualche parametro dovrà essere ovviamente ricalcolato.

Source Address: 192.168.2.10

L'indirizzo sorgente rimane lo stesso della modalità UDP.

Destination Address: 192.168.2.1

Anche l'indirizzo di destinazione viene mantenuto uguale a quello scelto nella modalità UDP.

Questa era l'intestazione del pacchetto IP, di seguito invece verranno illustrati i valori scelti per l'intestazione Ethernet.

Destination MAC Address: XX:XX:XX:XX:XX:XX

Dato che l'indirizzo MAC cambia in base al PC a cui ci si collega è inutile scrivere un valore specifico., l'importante è sapere che è lungo sei byte.

Source MAC Address: 02:02:02:02:02:02

Anche qui come per la modalità UDP è stato scelto un valore facilmente identificabile.

Type: 2048

Questo valore indica che nei dati trasportati è presente un pacchetto IP, il valore è stato preso dalla relativa tabella dello standard 802.3.

Frame Check: In realtà questo campo pur facendo parte dell'intestazione Ethernet, molto spesso viene gestito ad un livello più basso. Questo è possibile perchè il campo è posizionato alla fine di tutti i byte trasmessi e il codice CRC32 può essere calcolato in diretta sul flusso di bit che scorrono sul trasmettitore. Quindi risulta trasparente alla trama Ethernet, e addirittura non è possibile visualizzarlo neanche con un analizzatore di protocollo su PC.

Secondo lo standard 802.3 la massima dimensione di una trama Ethernet è di 1500 byte, superati i quali è necessario frammentare il pacchetto, quindi è stato scelto, rimanendo un po' conservativi, una dimensione massima dei dati trasportati dalla trama Ethernet di 1400 byte, questo è il motivo del valore scelto per il campo *Total Length* dell'intestazione IP.
Di conseguenza la dimensione massima del pacchetto UDP sarà di 1380 byte, che è appunto il valore inserito nell'apposito campo di dimensione del pacchetto.
Considerando che l'intestazione UDP occupa 8 byte ne consegue che alla fine rimangono 1372 byte da poter riempire con dati utente da trasmettere.
Dato che l'intestazione Ethernet è lunga 14 byte, in totale vengono caricati sul buffer di trasmissione 1414 byte, questo è il valore che assumerà la costante SEND_SIZE presente nel codice VHDL.
A questo punto è molto importante notare che tutti i campi delle intestazioni sono costanti e non rimane niente da calcolare in modo dinamico.
Questo è estremamente positivo in quanto permette di salvare interamente le intestazioni in una memoria ROM dedicata e quindi annullare la latenza dovuta

all'assemblamento dei pacchetti. L'unico impegno che le intestazioni richiedono è di essere trasmesse prima dei dati.
Questa ROM, come quella di configurazione, richiede una logica di controllo come la seguente:

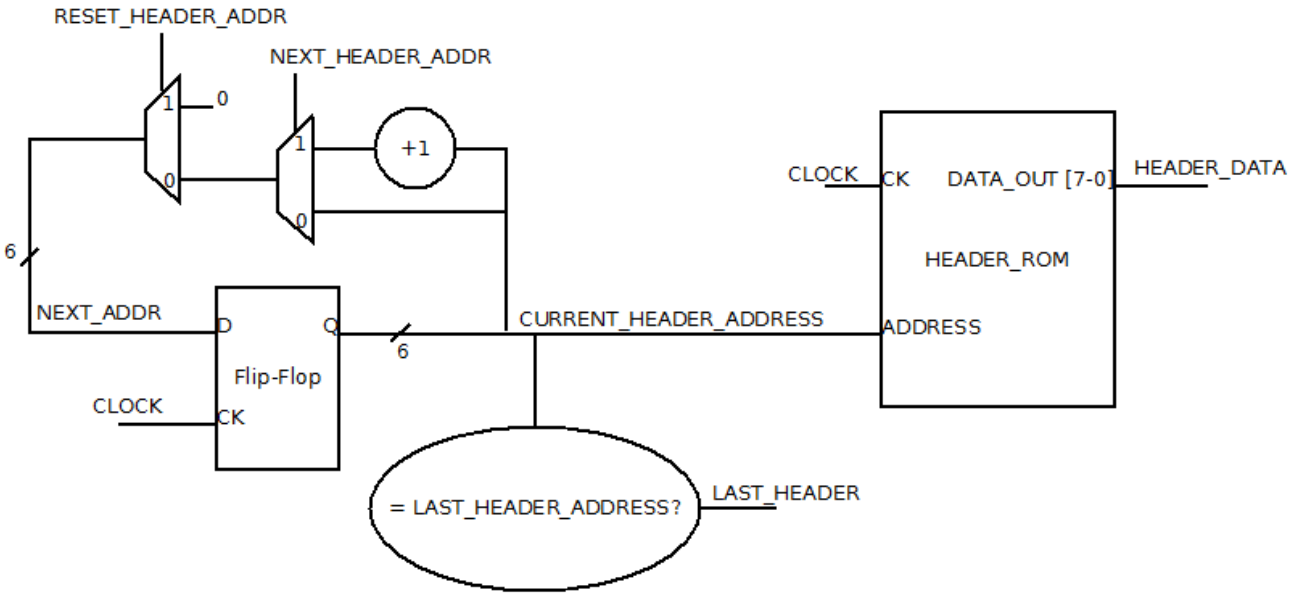


Figura 27: Logica di gestione della ROM delle intestazioni

Questa logica è del tutto simile a quella usata per la ROM delle configurazioni, ed è gestita dai seguenti stati:

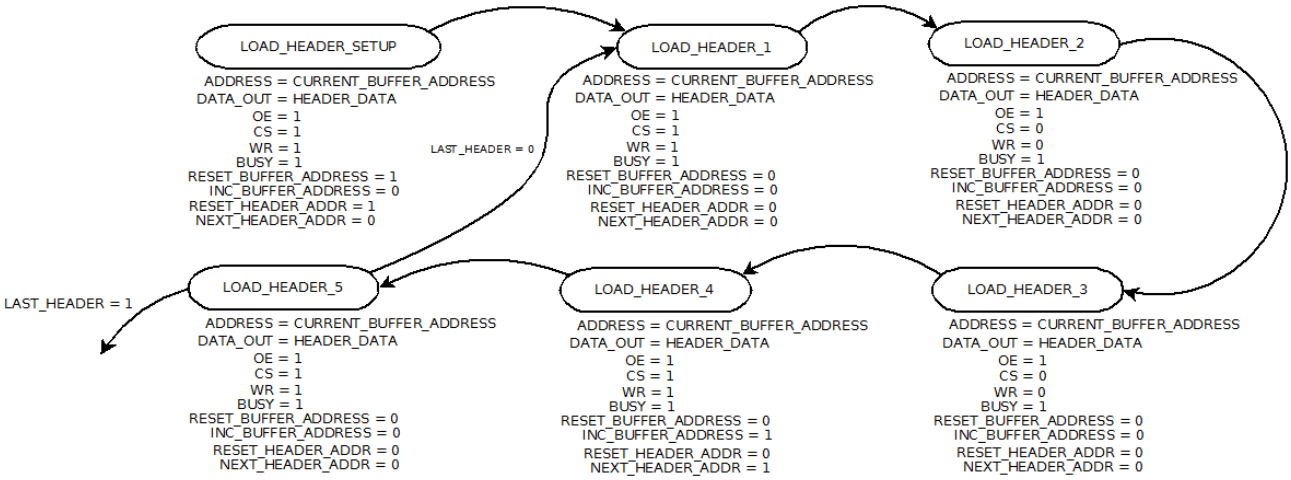


Figura 28: Stati caricamento intestazioni

Le intestazioni devono essere caricate sul buffer di trasmissione prima dei dati utente, quindi il loop di caricamento delle intestazioni deve avvenire tra gli stati di lettura di TX_WR e il loop TX, come illustrato dal seguente diagramma a stati finiti:

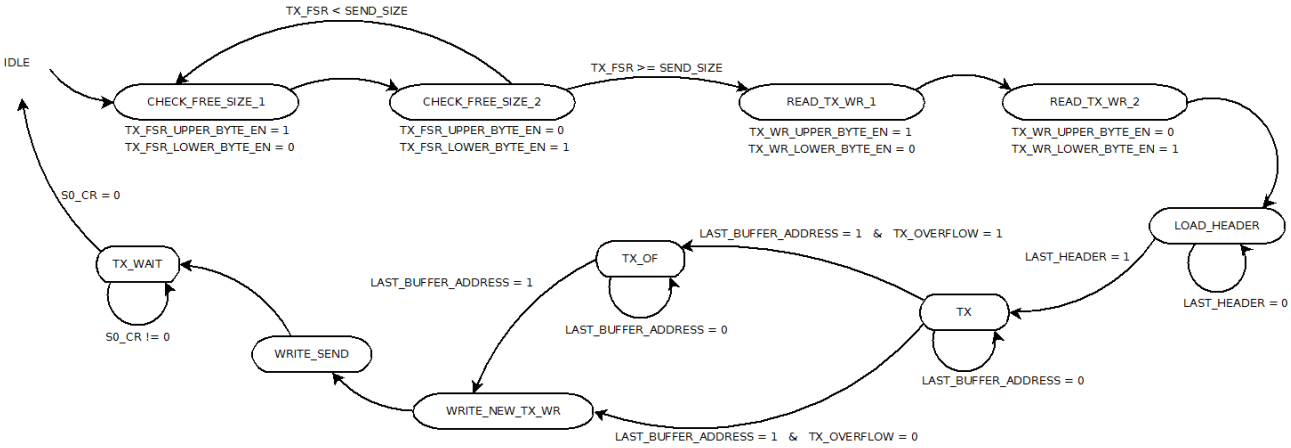


Figura 29: Stati di trasmissione in modalità MACRAW

Come per la trasmissione in modalità UDP, anche nella modalità MACRAW prima di caricare i dati nel buffer è necessario eseguire alcune operazioni preliminari, che in realtà sono proprio le stesse della modalità UDP, terminate queste è possibile procedere a caricare le intestazioni.
Una volta che sono state caricate le intestazioni e i dati sul buffer di trasmissione, ed è stato aggiornato il registro TX_WR con il nuovo valore, è possibile lanciare il comando di invio dei dati presenti sul buffer scrivendo l'apposito regstro.
Prima di tornare nello stato di attesa si entra in un loop dove viene letto in continuazione un registro che segnala il termine dell'operazione di invio, quando l'esito risulta positivo allora si può concludere la fase di trasmissione ritornando nello stato di attesa.

Capitolo 5

Piattaforma di test

5.1 Scheda di interfaccia

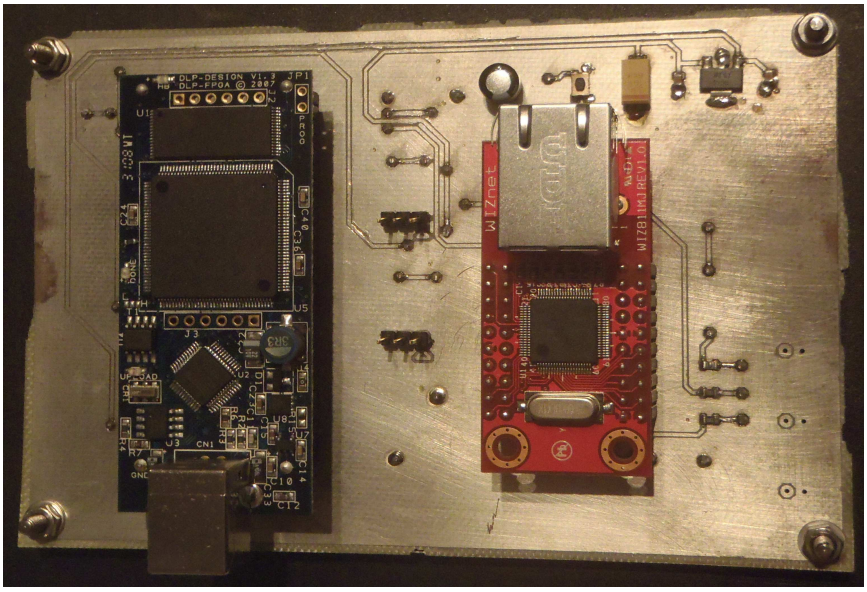


Figura 30: Scheda di interfaccia

Per interconnettere due i due moduli è stata utilizzata una apposita scheda di interfaccia.
Questa scheda si occupa di collegare opportunamente i pin del modulo Wiznet con quelli del modulo DLP e quindi di mettere in comunicazione il chip W5100 con l'FPGA.
Inoltre integra un regolatore di tensione che alimenta il modulo Wiznet con una tensione continua e stabilizzata a 3.3V ed un pulsante per resettare il chip W5100.
Sono anche presenti due jumper che rendono possibile selezionare la modalità di interfacciamento basata su SPI, i jumper servono per disconnettere due pin dell'indirizzamento diretto per collegare due pin dell'SPI. Questo espediente è

stato reso necessario dall'insufficienza di linee del modulo DLP.

5.2 Generatore di dati

Per eseguire i test è stato anche necessario realizzare una logica che generasse dei dati campione da trasmettere, e che effettuasse una richiesta di trasmissione di tali dati per verificare il corretto funzionamento del modulo.
Per generare questi dati è stato usato un contatore che genera un numero ad 8 bit che viene progressivamente incrementato man mano che il buffer viene riempito. La realizzazione è molto simile a quella della logica di gesione degli indirizzi delle ROM ed è illustrata nella seguente figura:

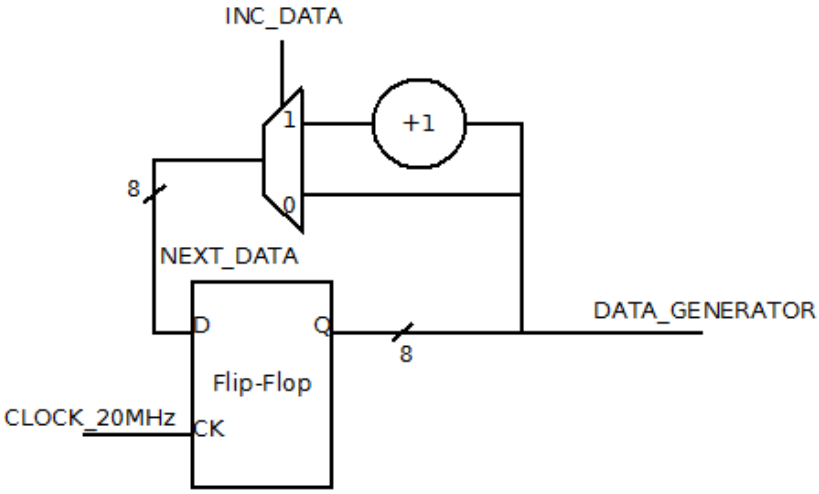


Figura 31: Logica generatore dati

Questa logica è stata inserita in un circuito più grande che oltre a gestirla serve anche per istanziare ed interconnettere il modulo che è stato sviluppato con il DCM, il buffer tri-state per l'I/O dei dati e il generatore stesso.
Inizialmente durante i test preliminari i registri non venivano scritti correttamente e, dopo varie prove, è stato realizzato che era necessario attendere un po di tempo prima di incominciare l'inizializzazione, per dare tempo al chip W5100 di avviarsi ed essere pronto a ricevere le configurazioni. Quindi è stato inserito un circuito di soft start, cioè un contatore molto lungo che tenga alto il segnale di reset del modulo Ethernet che è stato sviluppato per almeno 1.5 secondi.
Dato che il clock è di 20 MHz è stato necessario un bus di 24 bit per riuscire ad attendere per un tempo così lungo, una volta che il conteggio raggiunge una certa cifra il reset viene rilasciato e il conteggio stoppato in modo che la

configurazione possa avere inizio, lo schema è il seguente:

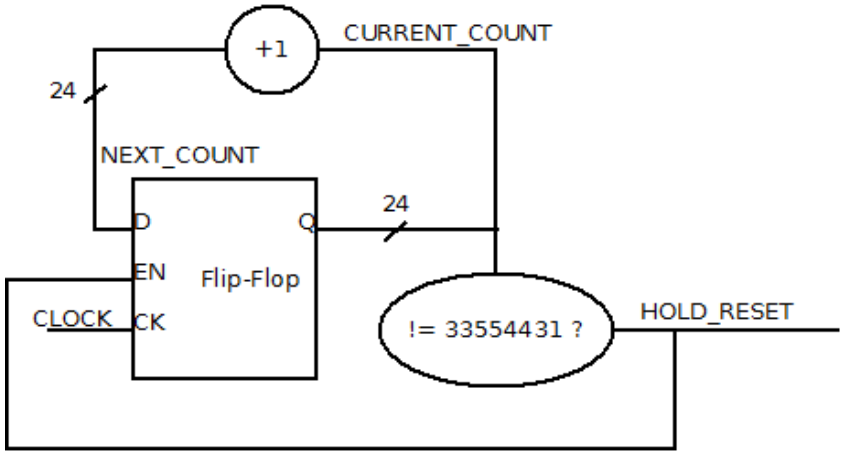


Figura 32: Soft start

Lo schema generale del tutto è il seguente è il seguente:

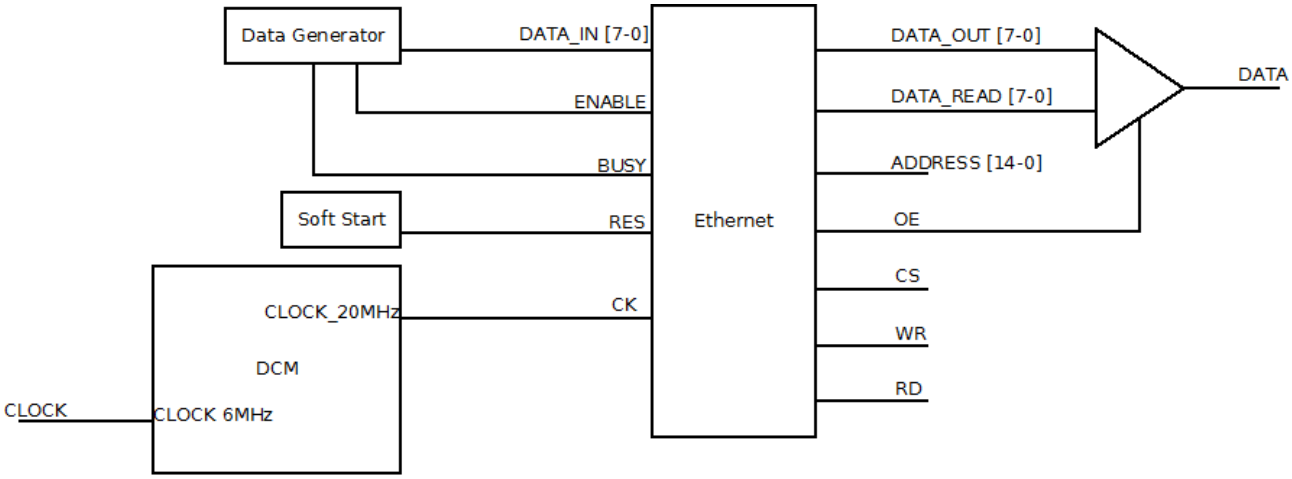


Figura 33: Schema logica di test

I bus DATA e ADDRESS, e i segnali CS, WR, RD sono connessi ai pin di uscita dell'FPGA e quindi al chip W5100.

5.3 Wireshark

Per studiare i pacchetti ricevuti sul PC è stato usato l'analizzatore di protocollo open source Wireshark.

Questo è un programma davvero completo e molto potente che permette di analizzare tutto il traffico di rete su qualunque interfaccia presente sul PC.
Dispone di un avanzato sistema di filtri, che permettono di isolare qualunque tipo di traffico dalla moltitudine di pacchetti di vario tipo che transitano in rete, ne consente l'analisi approfondita a qualunque livello dello stack e l'interpretazione secondo le direttive dei diversi protocolli.
Durante i test in modalità UDP è stato possibile osservare il flusso dei pacchetti ricevuti dal chip W5100, questo ha confermato la corretta configurazione del chip stesso. Quindi si è passati alla modalità MACRAW ed anche qui sono stati osservati i pacchetti ed è stato inoltre possibile analizzare le intestazioni dei vari protocolli per verificare che fossero corrette.
L'immagine seguente mostra un esempio dell'analisi di un pacchetto che è stato possibile effettuare con questo potente programma:

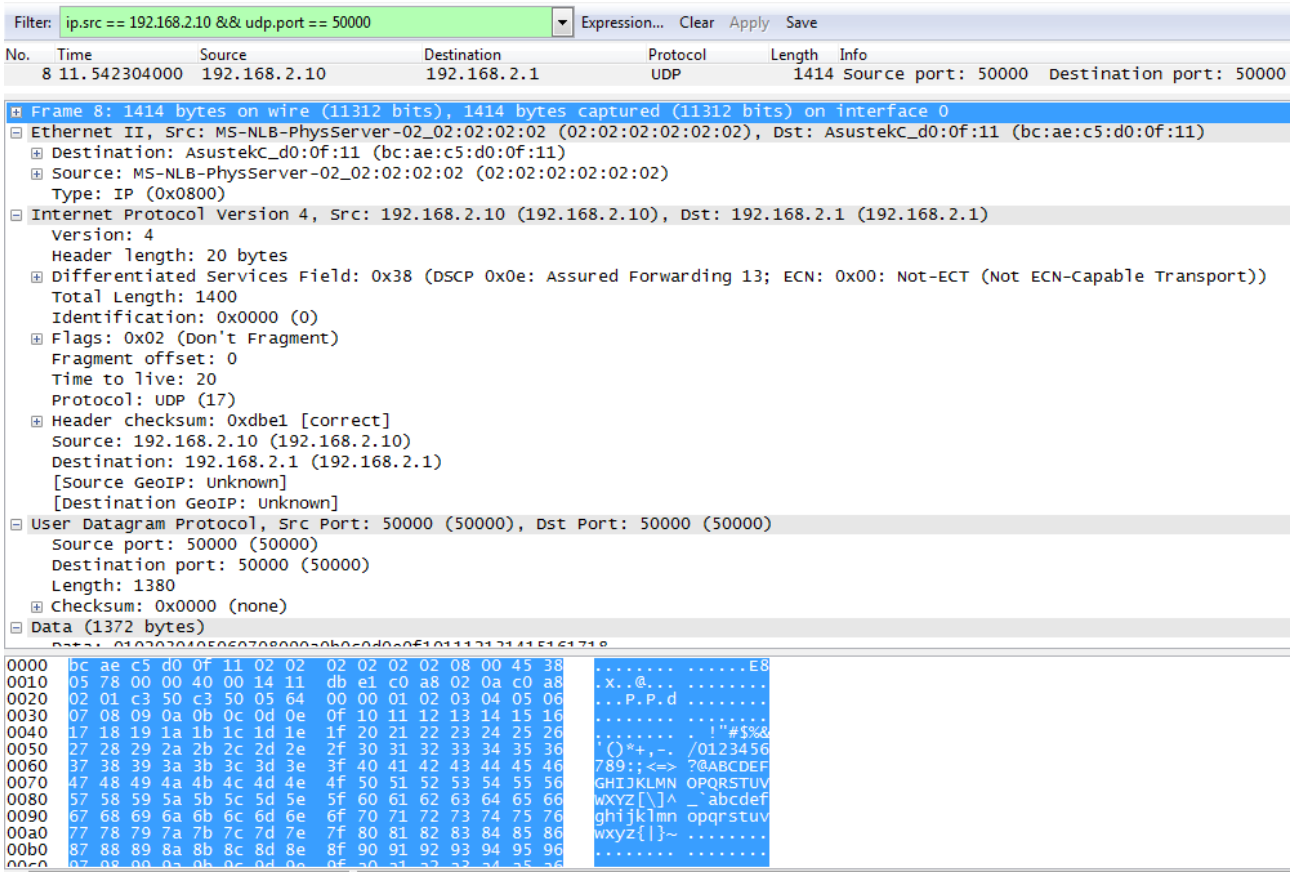


Figura 34: Analisi approfondita di un pacchetto UDP

Come è possibile osservare tutte le impostazioni descritte nei capitoli precedenti corrispondono perfettamente, questo ne dimostra la correttezza.

Capitolo 6

Conclusioni

Con questa tesi si è dimostrato come sia, non solo possibile usare l'interfaccia Ethernet come mezzo di comunicazione ad alta velocità con il PC, ma anche ottimizzarla per raggiungere la massima velocità e la minima latenza possibili con successo. Inoltre è stata realizzata una implementazione di esempio che è stata anche testata e verificata.

La massima ottimizzazione è stata raggiunta rendendo le intestazioni dei pacchetti completamente statiche e quindi salvabili direttamente sull'FPGA, eliminando così ogni latenza dovuta al calcolo dinamico di alcuni campi e alla genstione di altri.
Mettendo a rapporto le dimensioni delle intestazioni e la quantità totale di byte trasmessi, otteniamo un overhead di appena $\frac{42}{1414} = 3\%$, cioè su 100 byte trasmessi solo 3 non sono dati utili, ma la cosa importante è che questi 3 byte non hanno richiesto il lungo tempo tempo di elaborazione di una trasmissione tradizionale.

Durante i test però è anche emersa una limitazione del chip W5100, la massima bitrate che è stata misurata, usando il *timestamp* dei pacchetti su wireshark, è dell'ordine di soli 100-200 KB/s nonostante il chip sia compatibile con lo standard 100Mbps.
Questo si spiega con il fatto che il chip richiede molte operazioni di gestione prima e dopo una trasmissione e queste richiedono molto tempo.
Inoltre anche l'interfaccia ad indirizzamento diretto pone un limite alla velocità con cui le operazioni di gestione possono essere svolte, il limite teorico di trasferimento di questa interfaccia è di 10MB/s. Quindi anche se non dovessimo effettuare operazioni di gestione non sarebbe comunque possibile raggiungere la massima bitrate.
Alla luce di ciò possiamo concludere che il chip W5100 è adatto solo per operazioni a bassa velocità e per sgravare l'unità elaborativa dall'onere di creare

i pacchetti, oppure per operazioni di test come quelle che sono state eseguite per questo progetto. Per raggiungere le massime prestazioni è comunque necessario un chip PHY.

Nell'ottica di un ulteriore miglioramento, nell'ipotesi che venga usata un chip PHY Gigabit, sarebbe utile l'utilizzo dei jumbo frames.
Un jumbo frame è una particolare trama Ethernet la cui dimensione eccede il limite di 1500 byte, fino ad un massimo di 9000 byte.
Questa miglioria permetterebbe di portare l'overhead a $\frac{42}{9000} = 0.47\%$, un valore talmente irrisorio da rendere praticamente nulla la latenza dovuta alle intestazioni. Questo perchè le intestazioni verrebbero trasmesse meno volte rispetto alla quantità di dati inviati.

Bibliografia

[Thei11]: Federico Thei, “A Hybrid Technology For Parallel Recording Of Single Ion Channels”, 2011

[Perry02]: Douglas L. Perry, “VHDL Programming by Example”, 2002

[ieee802.3]: http://standards.ieee.org/about/get/802/802.3.html

[Kurose10]: James F. Kurose, “Computer Networking, a top-down approach” international edition, 2010

[RFC768]: IETF, “User Datagram Protocol”, RFC768

[RFC1624]: IETF, “Computation of the Internet Checksum via Incremental Update ”, RFC1624

[IANA]: http://www.iana.org/

[RFC791]: IETF, “Internet Protocol ”, RFC 791