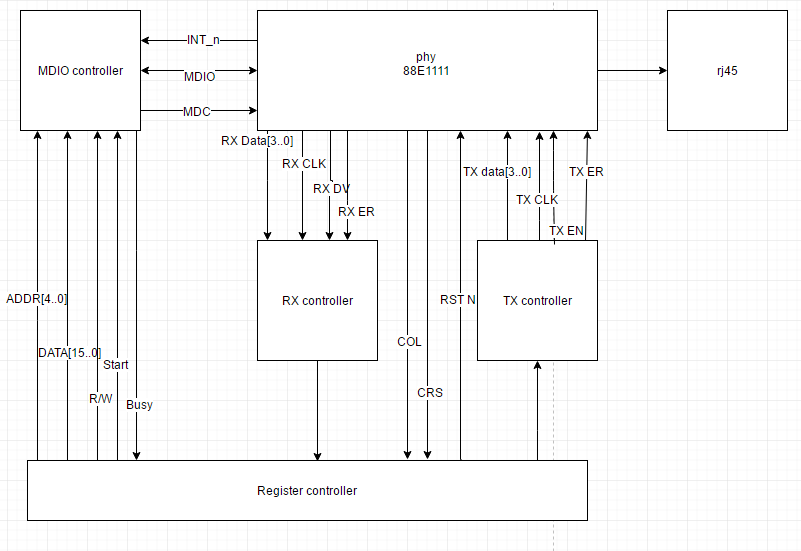
# System spesifikasjon

Ethernett mac MII 10/100 Mbps kontroller som er kompatibel med IEEE 802.3 med DE2-115 kortet som «target hardware» og komuniserer med 88E1111 phy chipene.

Her er et blokk diagram over modulen:



### MDIO Controller

Mdc er på 25 MHZ og bare på når den sender eller mottar data

MDIO endres på midten av den lave klokkeverdien, phy addressen kan endres vi generic i mdio controller koden.

## Ethernet Frame

|  |  |
| --- | --- |
| Destination address | 6 Bytes |
| Source address | 6 Bytes |
| Length / type | 2 Bytes |
| MAC client data | 46-1500 Bytes |
| Pad | Added to reach minimum length |
| Frame check sequence | 4 Bytes |

## PHY interface ports

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Signal name | Name | Width | Direction | Descpription |
| ENET0\_GTX\_CLK | GTX\_CLK | 1 | O | GMII Transmit Clock |
| ENET0\_INT\_N | Int\_n | 1 | I | Interrupt open drain output |
| ENET0\_LINK100 |  |  |  |  |
| ENET0\_MDC | MDC | 1 | O | Management data clock reference |
| ENET0\_MDIO | MDIO | 1 | IO | Management data |
| ENET0\_RST\_N | RESET\_n | 1 | O | Hardware reset signal |
| ENET0\_RX\_CLK | RX\_CLK | 1 | I | GMII and MII receive clock |
| ENET0\_RX\_COL | COL | 1 | I | GMII and MII collision |
| ENET0\_RX\_CRS | CRS | 1 | I | GMII and MII carrier sense |
| ENET0\_RX\_DATA[3:0] | RXD[3:0] | 4 | I | GMII and MII receive data[3:0] |
| ENET0\_RX\_DV | RX\_DV | 1 | I | GMII and MII receive data valid |
| ENET0\_RX\_ER | RX\_ER | 1 | I | GMII and MII receive error |
| ENET0\_TX\_CLK | TX\_CLK | 1 | I | MII transmit clock |
| ENET0\_TX\_DATA[3:0] | TXD[3:0] | 4 | O | MII transmit data[3:0] |
| ENET0\_TX\_EN | TX\_EN | 1 | O | GMII and MII transmit enable |
| ENET0\_TX\_ER | TX\_ER | 1 | O | GMII and MII transmit error |

## Registers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Address | Width | Access | Description |
| ModeReg | 0x00 | 12 | RW | Mode register |
| MINFL |  | 16 | RW | Minumum  pakkelengde register |
| MAXFL |  | 16 | RW | Maximum pakkelengde register |
| CollConf |  | 10 | RW | Collision og retry configuration register |
| MIIModeReg |  | 9 | RW | MII mode register |
| MIICommand |  | 7 | RW | MII command and PHY Address register |
| RegAddr |  | 5 | RW | Reg address register |
| DataO |  | 16 | RW | MII transmit data register |
| DataI |  | 16 | R | Mii recieve data register |
| MIIstatus |  | 2 | RW | MII status register |
| MACAddr0 |  | 48 | RW | MAC addr register |

Legge til mdio register fram github

|  |  |  |
| --- | --- | --- |
| ModeReg |  |  |
| Bit # | Access | Description |
| 11 | RW | Recieve Small Packets  0= Packets smaller than MINFL are ignored  1 = Packets smaller than MINFL are accepted |
| 10 | RW | Padding enabled  0 = dont pad small frames  1 = add pad to small frames to reach MINFL |
| 9 | RW | Huge Packets enable  0 = All bytes above MAXFL in frame is discarded  1 = frames up to 64 KB |
| 8 | RW | CRC enable  0 = TX controller does not append the CRC  1 = TX controller appends the CRC to every frame |
| 7 | RW | Delayed CRC enable  0 = CRC calculation starts after SFD (start frame delimiter)  1 = CRC calc starts 4 bytes after SFD |
| 6 | RW | Full Duplex  0 = Half Duplex mode  1 = Full duplex mode |
| 5 | RW | No backoff  0 = normal backoff algorithm  1 = TX retransmit immediatly after collision |
| 4 | RW | Loop back  0 = normal operation  1 = TX is looped back to the RX |
| 3 | RW | Promiscuous  0 = check destination address of incoming frames  1 = Receive frame regardless of address |
| 2 | RW | No Preamble  0 = normal operation (7 byte preamble)  1 = no preamble is sent |
| 1 | RW | Transmit enable  0 = disabled  1 = enabled |
| 0 | RW | Receive enable  0 = disabled  1 = enabled |

Ikkje endre register etter at det er skrivd til transmit enable eller receive enable.

|  |  |  |
| --- | --- | --- |
| MINFL |  |  |
| Bit # | Access | Description |
| 15-0 | RW | MINFL - Minimum Frame Length  Standard value 64 byte  to receive smaller asset Receive small packets (ModeReg)  To transmit small asset Padding enabled (ModeReg) |

|  |  |  |
| --- | --- | --- |
| MAXFL |  |  |
| Bit # | Access | Description |
| 15-0 | RW | MAXFL - Maximum Frame length  standard value 1536 byte (0x0600)  maximum ethernet packet is 1518 byte  Asset Huge Packets enable (ModeReg) to send bigger packets |

|  |  |  |
| --- | --- | --- |
| CollConf |  |  |
| Bit # | Access | Description |
| 6-9 | RW | Maximum Retry Maximum  numbers of consequential retransmission attempts after collision is detected. When reached TX reports an error and sotp transmitting.  Standard value 0xF (15) ethernet standard |
| 5-0 | RW | Collision Valid  Collision Time Window late collision occuring later than 64 bytes from preamble. Transmission of the current packet is aborted.  Standard value is 0x3F |

|  |  |  |
| --- | --- | --- |
| MIIModeReg |  |  |
| Bit # | Access | Description |
| 8 | RW | No preamble  0 = 32-bit preamble sent  1 = no preamble sent |
| 7-0 | RW | CLK Divider Host clock divider factor |

|  |  |  |
| --- | --- | --- |
| MIICommand |  |  |
| Bit # | Access | Description |
| 7 | R | Ack – Ack when done |
| 6 | WI | Rd – Read |
| 5 | WI | WR – Write |
| 4-0 | RW | PhyAddr - PHY address |

|  |  |  |
| --- | --- | --- |
| RegAddr |  |  |
| Bit # | Access | Description |
| 0-4 | RW | Register address |

|  |  |  |
| --- | --- | --- |
| DataO |  |  |
| Bit # | Access | Description |
| 0-15 | RW | Data out to PHY |

|  |  |  |
| --- | --- | --- |
| DataI |  |  |
| Bit # | Access | Description |
| 0-15 | R | Data in from PHY |

|  |  |  |
| --- | --- | --- |
| MIIStatus |  |  |
| Bit # | Access | Description |
| 1 | R | Busy |
| 0 | R | Linkfail |

|  |  |  |
| --- | --- | --- |
| MACAddr |  |  |
| Bit # | Access | Description |
| 40-47 | RW | Byte 0 of ehternet MAC address |
| 32-39 | RW | Byte 1 of ehternet MAC address |
| 31-24 | RW | Byte 2 of ethernet MAC address |
| 23-16 | RW | Byte 3 of ethernet MAC address |
| 15-8 | RW | Byte 4 of ethernet MAC address |
| 0-7 | RW | Byte 5 of ethernet MAC address |

## Ram

Held data for transmitter og lagrer data fra receiver

|  |  |
| --- | --- |
| Memory Range (hex) | Memory Usage |
| 00000 - 0FFFFF | Transmitter buffer - holds data to be sent by the transmitter |
| 10000 - 1FFFF | Free |
| 20000 - 2FFFF | Free |
| 30000 - 3FFFF | Receiver buffer - stores data received by the receiver |
| 40000 - 4FFFF | Free |
| 50000 - 5FFFF | Free |
| 60000 - 6FFFF | Status buffer - stores status bits produced by the receiver |
| 70000 - 7FFFF | Free |

## C:\Users\Anders\Documents\6 semester\bachelor\Design spec\blokkdiagram_krav0001.jpg

## Ethernet MAC Receiver

Moduler:

* RX State Machine

Når RX\_dv går høg, indikerar det at det kjem inn data pakkar, som aktiverer SFD Detector. Når SFD er detektert vil frame-en bli holdt i buffer, mens CRC og destination address blir sjekka. Frame-en blir deretter sent i ein FIFO, for å takast vidare. På slutten av mottakinga vil state machina merke at pakken har blitt mottatt med å sette opp rx\_status.

* SFD Detector

Detekterer at starten av frame har blitt mottatt.

Aktiverast av signal fra RX State Machine etter rx\_dv er mottatt. Når SFD er detektert returnerar den eit signal.

* RX Buffer

Held frame som er mottatt, før dei blir sendt vidare til FIFO.

Lar SFD detector og AML hente ut data heile tida, og sender vidare data når den får klarsignal.

* Address Matching Logic

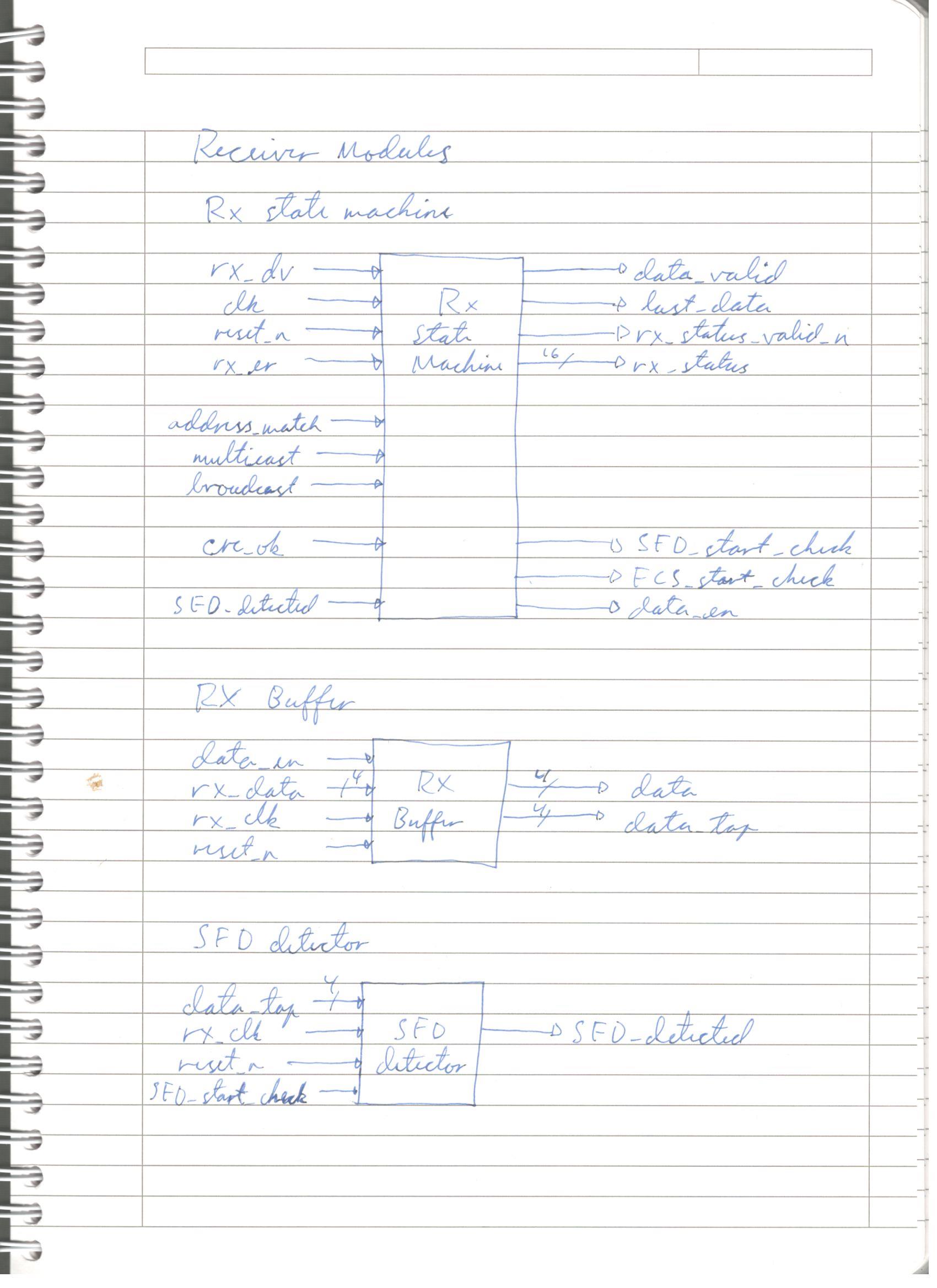
Sjekkar mottakar addresse mot ethernet addressa til FPGAen.

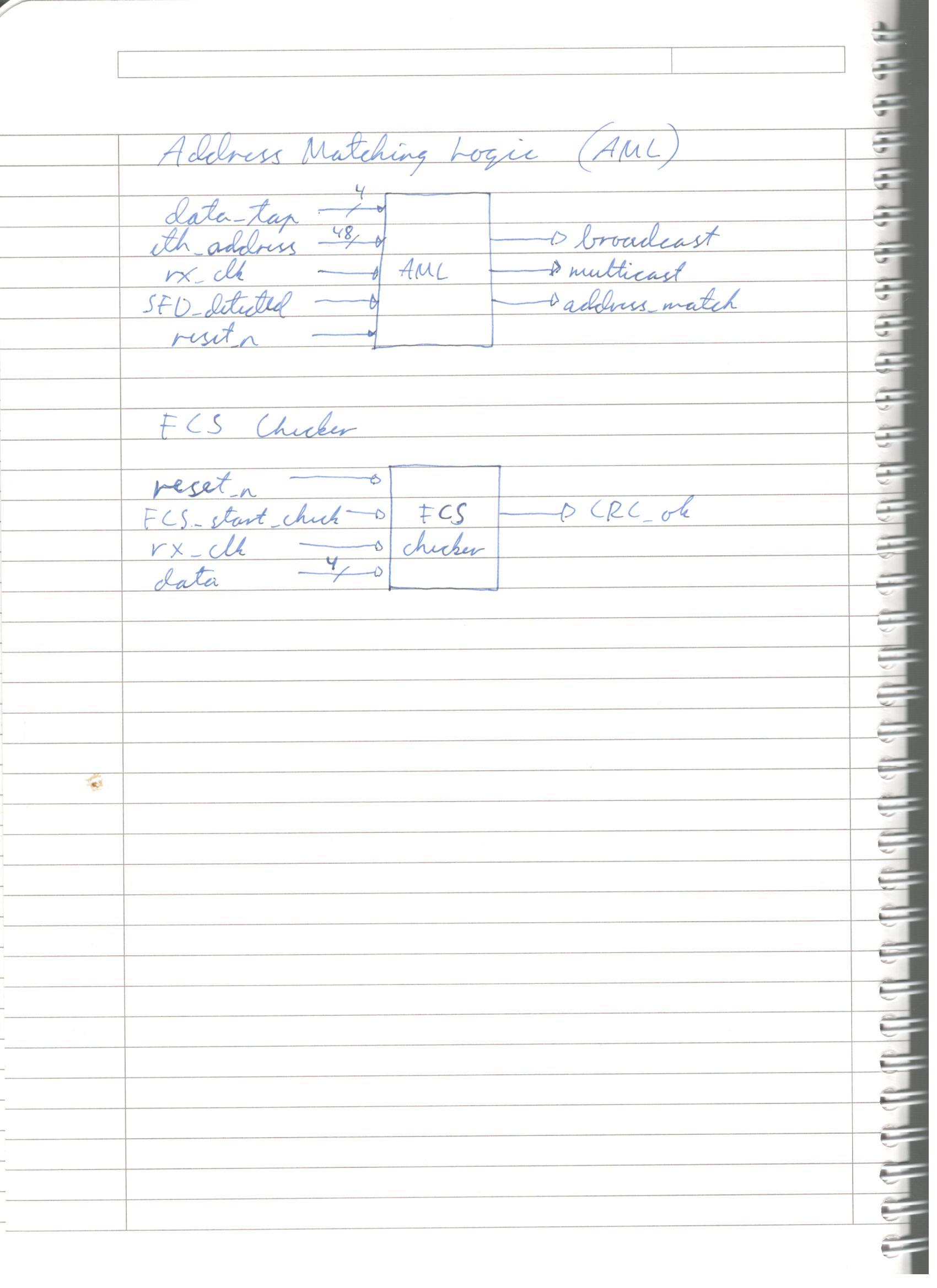
Rett addresse gjev ok signal, feil gjev error.

* FCS Checker

Sjekkar CRC av frame.

Sendar ut ok signal om det stemmer, error om feil.





## Ethernet MAC Transmitter

Moduler

* FIFO Sync

Synkroniserer output med rising edge av tx\_clk. Sender ut data nibble når transmit\_enable fra State Machine. Sender ut tx\_en og tx\_er og.

* Inter Frame Gap (IFG) Timer

Skal sørge for at tidsintervallet mellom to datapakker (Inter frame gap) er på 96 bit tid (24 klokke sykluser). Viss crs kjem i første 60 bit tid blir counter reset. Siste 36 bit tid vil tellaren gå ut.

Half duplex: Tellaren må komme seg igjennom IFG uten crs signal. Då blir transmit\_available pulse sendt.

Full duplex: Ignorerar crs, og startar å telle med ein gong transmit\_enable blir låg.

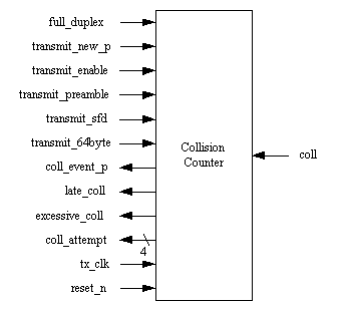
* Defer Counter

Begrensar ventetid for å vente på ein muligheit på å sende ein ny pakke med data. Får puls fra IFG om at ein ny pakke er klar for sending. Startar ein tellar, og

* Frame Length Counter

Teller lengden på frame som har blitt sendt til PHY. Teller så lenge transmit\_enable er aktiv, ellers reset. Etter å ha telt frame, telles FCS. Sørger for at pading blir lagt til om for liten, og kuttar om for stor.

* Collision Counter (tonast ned)



Teller antall kollisjon event, nummer av kollisjonar på rad og seine kollisjonar. Dersom det er for mange kollisjonar (15) vil sendingen bli avbryt.

* Random Number Generator

Genererar eit tilfeldigt nummer mellom 0 og (2^k)-1, kor k er antall kollisjonar som har vårte telt. Deretter blir det sendt vidare til backoff timer.

* Backoff Timer

Bestemmer kor mange interval ein backoff operasjon skal bruke.

* Jam Timer

Bestemmer kor langt JAM mønster har blitt sendt etter ein kollisjon.

* CRC Generator

Genererar CRC nummeret, som blir lagt til frame som FCS feltet. Kalkulert fra måladdresse til datafelt.

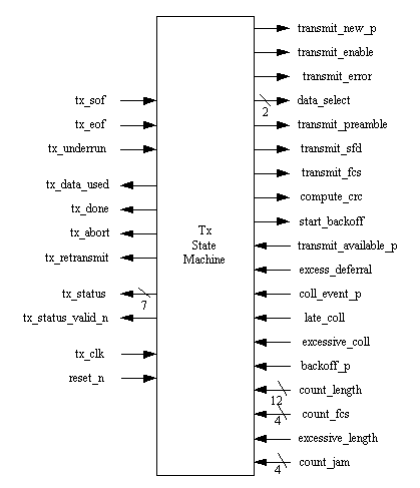
* Data Multiplexer

Bestemmer kva data som blir sendt til PHY. Får data\_select fra state machine.

Preamble, JAM og PAD = 4’b0101 = 0x5

SFD = 4’b1101 = 0xD

* TX State Machine



TX State Machine skal holde kontroll på transmit prosessen. Får tx\_sof når modulen får ein pakke som skal sendes. Deretter sendes ut signal for å restarte Defer Counter og Collision Counter og venter på klarsignal fra IFG timer. Deretter blir Preamble og SFD sendt ut. Venter på nytt klarsignal fra IFG, mens det følges med på Defer Counter. Når IFG tida har passert og det har oppstått ein muligheit for å sende ut data, vil State Machine sende signal til Multiplexer og eit enable signal til FIFO sync, Collision Counter, Frame Length Counter og IFG Timer.

Sender ut Preamble Field, når det får klarsignal for sending. Sendes ut for dei neste 14 klokkesyklusane. Deretter blir SFD Field sendt ut i løp av ein klokkeperiode. Me har då fått 14 tilfeller av 0x5, etterfølgt av 0xD. Deretter blir Data Field blir sendt om count\_length er lang nok til at tx\_eof blir detektert. Viss lengden av count\_length er mindre enn 64 bit, vil PAD bli sendt ut. Dersom dette ikkje er tilfelle vil PAD vere deaktivert. Etter data field har blitt sendt ut og tx\_eof er vorte detektert, vil FCS field bli transmita. Når data field starte å sende blir tx\_data\_used og compute\_crc blir sett. Deretter blir tx\_eof overvåka. Når den siste nibblen er plassert i tx\_data og me får tx\_eof og me ikkje treng å sende PAD, blir compute\_crc satt låg og me starter sendingen av FCS ved å sende transimt\_fcs til Frame Length Counter.

Så lenge sendingsoperasjonen pågår vil tx\_underrun, coll\_event\_p og excessive\_length bli overvåka. Dersom ein av dei blir detektert vil sendingsprosessen bli avbryt eller droppa. Dersom coll\_event\_p blir detektert, vil SM sende ut tx\_retransmit og start\_backoff. Samt transmit\_enable, compute\_crc og tx\_data\_used blir reset. Ventar så på backoff\_p for å så sette transmit\_enable som startar sendinga fra starten av preamble field. Eller dropper sendinga viss excessive\_coll vert detektert.

