Heterogeneous IP Block Interconnection versio 2.0

Toiminnallinen määrittely

Taulukko 1: Yleistä

TTKK	**Digitaali- ja tietokonetekniikka**
Tekijä	Erno Salminen
Jakelu	AWT Engine
Dokumentin tila	Työversio
Muokattu	
Tulostettu	toukokuuta 19, 2003

Taulukko 2: Versiohistoria

Versio	**Päiväys**	**Tekijä**	**Selite**
0.1	02.10.2001	ES	Dokumenttipohja
	23.10	ES	Dataleveydet
	20.12	ES	Kimmon haluamat muutokset

1. JOHDANTO

1.1.Tarkoitus ja kattavuus

Tässä dokumentissa määritellään *Heterogeneous IP Block Interconnection version 2*:n (HIBI2) toiminnallisuus. Ensimmäinen versio HIBI:stä on kuvattu dokumenteissa [1] [2] [3].

1.2.Tuote ja ympäristö

Kyseessä on järjestelmäpiirillä käytettävä piirin sisäinen väylä.

1.3.Määritelmät, termit ja lyhenteet

Taulukko 3. esittelee joitakin tässä dokumentissa käytettyjä termejä. Dokumentissa viitataan HIBIn versioon 2 lyhyesti sanalla hibi tai hibiväylä. Puhuttaessa ensimmäisestä versiosta mainitaan ensimmäinen versio erikseen.

Taulukko 3: Termit

Termi	Merkitys
Agentti	Yleisnimitys väylään kytketystä lohkosta. Hibissä agentti koostuu liityntälohkosta ja yhdesta tai useammasta toiminnallisesta lohkosta.
Arbitrointi	Tapa, jolla päätetään mikä lohko voi kirjoittaa väylälle.
Datasana	*n* bittiä dataa, missä n = IP:n dataväylän leveys
IP-lohko	*Intellectual property.* Tässä tapauksessa valmiina hankittava HW-lohko. Esim. muisti tai mikroprosessori.
Järjestelmä	Tässä dokumentissa käytetään merkityksessä, jossa järjestelmä koostuu yhdestä tai useammasta hibiväylästä ja siihen/niihin kytketyistä agenteista.
Liityntälohko	Lohko, jolla toiminnallinen lohko kytketään hibiväylään. Jokaisessa agentissa on liityntälohko.
Master	Agentti joka hallitsee väylää. Ainoa lohko joka voi sillä hetkellä kirjoittaa väylälle.
Master+Slave	Agentti, joka voi toimia sekä masterina että slavena.
Parametrisivu	Liityntälohkoon voi tallettaa monta konfiguraatiota, jokainen konfiguraatio on omalla sivullaan.
Rx, Receive	Vastaanotto-
Slave	Agentti joka ei voi käynnistää operaatioita, ainoastaan vastata niihin.
Toiminnallinen lohko	Laskentayksikkö tms. joka voidaan kytkeä HIBI-väylään liityntälohkon avulla.
Tx, Transmit	Lähetys-
Väylä	Toiminnalliset lohkot yhdistävä kommunikointiarkkitehtuuri. Samat johtimet menevät kaikille lohkoille ja lohkot saavat vuorotellen kirjoittaa väylälle, mutta voivat lukea sitä koko ajan.
VHDL, Verilog	Laitteiston kuvaamiseen käytettäviä kieliä.

Taulukko 4: Dokumentissa käytettävät merkintätavat

Merkintätapa	Merkitys
lihavointi	
kursivointi	Signaalien nimet tekstissä.
ISOILLA KIRJAIMILLA	
hakasuluisssa	viitteet
_n signaalin nimen perässä	alhaalla aktiivinen

Taulukko 4. luettelee dokumentissa käytettävät merkintätavat.

1.4.Yleiskatsaus dokumenttiin

Vaikka dokumentti on kirjoitettu suomeksi, on hibin signaalit ja lohkot nimetty englanniksi.
Ensimmäisessä luvussa on kuvattu dokumentin ja toisessa väylän yleisiä ominaisuuksia. Kolmannessa kuvataan toiminnot, joita väylällä voi suorittaa. Neljännessa esitelllään liityntälohkon ja viidennessä väylän rakenne. Kuudes luku kuvaa siltalohkon toiminnan. Seitsemäs luku käsittelee muita ominaisuuksia kuten arvioitua suorituskykyä. Kahdeksannessa luvussa luetellaan hylätyt ratkaisut ja annetaan lyhyet perustelut hylkäämiselle. Yhdeksännessä luvussa esitellään jatkokehitysajatuksia, joita ei vielä kuitenkaan ole toteutettu ja kymmenennessä luetellaan vielä avoimeksi jätettyjä kohtia. Yhdennessätoista luvussa on määrittelydokumentin lähdeviitteet ja kahdennessatoista luvussa ovat liitteet, mm. järjestelmän ajoituskaavioita.

2. YLEISKUVAUS

2.1.Toiminta

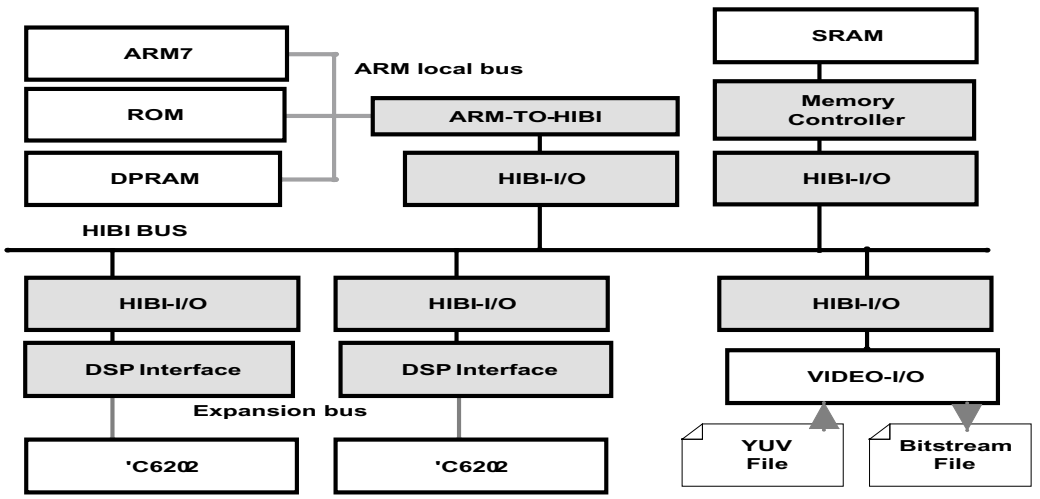
HIBI-väylällä voidaan kytkeä useita väyläagentteja toisiinsa, jolloin ne voivat siirtää tietoa keskenään. Väylän parametreja voi muokata monipuolisesti, jotta monentyyppiset agentit voivat sitä käyttää. Osaa parametreistä voi muokata myös ajon aikana, jolloin kokonaisjärjestelmä voi mukautua sovelluksen mukaan.
Kuva 1. esittelee esimerkkinä hibin ensimmäisen version avulla suunnitellun videokoodausjärjestelmän. Kuvan keskellä hibiväylä yhdistää järjestelmän agentit toisiinsa. Jokainen agentti sisältää liityntälohkon (HIBI-I/O) ja IP-lohkoja tai muuta logiikkaa. Jokainen agentti voi vuorollaan varata väylän käyttöönsä siirtääkseen dataa.

2.2.Väylän signaalit

Suunnittelussa on ollut tavoitteena minimoida käytettävien signaalijohtimien määrä.Kaikki väyläsignaalit menevät kaikille väylään liitetyille agenteille.

2.3.Agenttien rajapinta liityntälohkoon

Kaikilla liityntälohkoilla on samanlainen rajapinta IP-lohkoon. Rajapinnan signaalit esittellään Taulukkossa 17. kappaleessa 12.1.



Kuva 1.Videokoodausjärjetelmä

3. TOIMINNOT

Väylän perustoiminto on siirtää yksi datasana agentilta A agentille B. Koska väylä on jaettu resurssi, tulee varmistua ettei voi syntyä konflikteja, eli että kaksi agenttia yrittää kirjoittaa yhtä aikaa. Arbitrointi on tapa, jolla päätetään mikä lohko milloinkin voi hallita väylää (= kirjoittaa väylälle). Kaikki yksittäiset signaalit ovat ylhäällä aktiivisia paitsi asynkroninen reset-signaali Rst_n (_n=alhaalla aktiivinen).

3.1.Komennot

Hibissä käytetyt komennot esitellään Taulukkossa 5.
Kirjoitusoperaatioita on kolmea tyyppiä data, viesti tai konfigurointidatan kirjoitus. Kirjoitus voi suuntautua joko yhdelle agentille tai multicast-operaationa useammalle agentille. Tarvittaessa kirjoitukset ohjautuvat myös väyliä yhdistävien siltojen ylitse.
Lukuoperaatio suoritetaan kahdessa osassa : ensin pyydetään dataa ja toinen lohko siirtää sen tavallisella kirjoituskomennonlla, kun data on valmis ja se saa väylän käyttöönsä. Myös liityntälohkojen konfigurointiarvoja voi lukea.
Sekä osoitteen että datan siirto tapahtuu samalla väylällä. Siirron alussa (ja kohteen vaihtuessa) siirretään osoite ja sen jälkeen dataa.Käytetyt komennot esitellään tarkemmin seuraavissa luvuissa.

3.2.Siirtotavat

Siirron alussa liityntälohko lähettää vastaanottavan agentin osoitteen. Osoitetta ei lähetetä ellei voida varmistua, että ainakin yksi data saadaan lisäksi siirrettyä ennen kuin väylästä joudutaan luopumaan. Kaikki agentit, joiden osoiteavaruuteen osoite kuuluu, vastaanottavat datan. Kun liityntälohko siirtää osoitetta, se asettaa *AddrValid*-signaalin aktiiviseksi (“1”). Yhden osoiteen siirto voi kestää useamman kellojakson. Tämä on tarpeen lähinnä kun käytetään kapeata (esim. 8 bittinen) dataväylää ja järjestelmässä on paljon agentteja/konfigurointiparameterejä. Synteesin aikana päätetään mikä on osoitteen pituus. Kellojaksojen määrä lasketaan järjestelmän rajoitteista, kuten maksimi agenttien määrä, maksimi väylien määrä, maksimi konfigurointisivujen määrä jne. Osoitteen koko on aina joku väylänleveyden monikerta. Esim. siirrosta

- kellojakso 1 : dataväylä = osoitteen 1. puolikas, Addr_Valid =1, komento = Write_Data
- kellojakso 2 : dataväylä = osoitteen 2. puolikas, Addr_Valid =1, komento = Write_Data
- kellojakso 3 : dataväylä = 1. data, Addr_Valid =0, komento = Write_Data
- kellojakso 4 : dataväylä= 2. data, Addr_Valid =0, komento = Write_Data
- ...
- kellojakso n : dataväylä =n-2. data, AddrValid=0, komento = Write_Data

Taulukko 5: Käytettävät komennot

Komento	Koodi	Merkitys
Idle	000	Väylä vapaa
Write_Config_Data	001	Konfigurointi. Broadcast => agent ID = 0
Write_Data	010	Datan siirto
Write_Message	011	Viestin siirto
Read_RQ	100	Lukupyyntö
Read_Config	101	Konfiguroinnin lukupyyntö
Multicast_Data	110	Datan siirto monelle kohteelle
Multicast_Message	111	Viestin siirto monelle kohteelle

Siirrettävä data ja osoite talletetaan väliaikaisesti liityntälohkon sisälle fifopuskureihin ennen sen kirjoittamista hibiväylälle. Lähettävä agentti voi määritellä siirrettävän datan *viestiksi* jolloin se ohjataan viesteille varattuun omaan puskuriin. Viesteillä on suurempi prioriteetti kuin datalla, joten viestipuskurista kirjoitetaan väylälle ennen kuin datapuskurista. Vastaanottava toiminnallinen lohko voi myös halutessaan lukea viestit ennen dataa. Kaikki siirrot HIBI-väylällä tehdään kelloon sidottuna eli synkronisesti.

IP-lohkot voivat käyttää erilevyistä dataväylää kuin HIBI-väylä. Kummankaan väylän leveyttä ei kuitenkaan voi valita vapaasti, vaan niiden täytyy olla kahden potensseja (muotoa *2^i* bittiä), esim. 8/16/32/64 bittiä. Liityntälohkojen lähetyspuskurit ovat samanlevyisiä kuin IP-lohkon dataväylä ja niiden pituus valitaan synteesin aikana. Sen sijaan vastaanottopuskurit ovat saman levyisiä kuin hibiväylä. Muuten olisi vaikeata lukea väylältä dataa viivästämättä lähettäjää. Jokainen datan siirto väylällä vie korkeintaan yhden kellojakson. Kannattaa huomata, että jos järjestelmässä on yksi muita leveämpi lohko, ei yksikään muu lohko pysty vastaanottamaan tai lähettämään yhtä leveää dataa. Tällöin ei pystytä hyödyntämään leveän lohkon koko siirtokapasiteettia, vaikka laskennassa voidaankin. Kun vastaanottavalla liityntälohkolla on hibiväylää kapeampi puskuri, datan ylimmät tavut jätetään huomioimatta. Lähettävä agentti ei voi kertoa ilman erillistä viestiä montako tavua datasta on hyodyllistä. Tämän tyyppinen tieto kuuluu ylemmälle tasolle, joten sitä varten ei ole varattu omia väyläsignaaleja vaan tieto välitetään tarvittaessa erillisellä viestillä.

3.3.Uudelleen konfigurointi

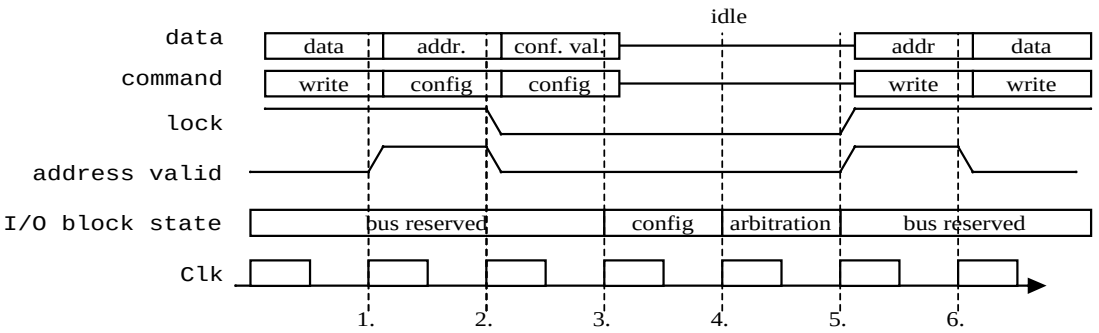
Liityntälohkon konfigurointiparametrejä on mahdollista muuttaa ajon aikana yksi kerrallaan tai vaihtaa käytettävää parametrisivua. Sivua vaihdettaessa kaikki parametrit vaihtuvat kerralla. Resetin aikana kaikki parametrisivut saavat samat oletusarvot. Kaikille lohkoille yhteiset tiedot, kuten aikaslottien tiedot, kannattaa kirjoittaa broadcast-operaatiolla.

Konfigurointiparametrien asettamisessa on kaksi tapaa

- Muutetaan käytössä olevan parametrisivun parametrejä. Esim. kehyksen pituutta, aikaslottien tietoja. Tällöin tulee ehdottomasti varmistua, että ei aiheudu ristiriitoja, kuten että kahdella agentilla olisi sama prioriteetti tai päällekkäiset aikaslotit!
- Kirjoitetaan parametrit sivulle, joka ei ole käytössä. Kun kaikki arvot on asetettu vaihdetaan parametrisivua. Tämä on huomattavasti turvallisempi tapa hoitaa konfigurointi. Konfigurointisivua vaihdettaessa nollataan kellojaksolaskuri.

Uudelleen konfiguroinnissa

- Komento on *write config* tai *multicast config*
- Konfiguroinnissa osoite tulkitaan eri tavalla kuin datan/viestien siirrossa
- Osoite jakautuu kenttiin kohdeagentti, parametrisivun numero ja parametrin numero
- Kenttien leveys bitteinä määritellään synteesiaikana. Parametrin osoittamiseen on varattava vähintään neljä bittiä. Samalla määritellään käytetäänkö konfiguroinnissa normaaleja vai pitkiä osoitteita
- Pitkien osoitteiden kanssa kentät lähetetään järjestyksessä kohteen parametrin numero, parametrisivu ja id. Eli alin osa ensin
- Parametriarvo voi olla esim. agentin prioriteetti tai seuraavaksi käytössä olevan parametrisivun numero



Kuva 2.Konfiguroinnin ajoituskaavio, kun osoitteen siirto tapahtuu yhdessä kellojaksossa.

- Kirjoitus voi tapahtua myös parametrisivulle joka ei parhaillaan ole käytössä. Näin voidaan uusi konfiguraatio kirjoittaa valmiiksi kaikille agenteille ja sen jälkeen vaihtaa parametrisivua.
- Esim. 32 -bittinen osoite => agentti =5 (16b) , parametritsivu= 1 (8b), parametri= 2 (8b) => osoite 0x00050102
- Jos kaikkia kenttiä ei voi siirtää kerralla, pitää osoite siirtää useamman kellojakson aikana. Vaadittujen kellojaksojen määrä pysyy kuitenkin suorituksen ajan vakiona mutta ei välttämättä ole sama kuin data- tai viestisiirroissa.
- Broadcast-operaatiossa kohdeagentin IDksi merkitään 0, jolloin komento menee kaikille oman väylän agenteille. Konfigurointi sillan ylitse tapahtuu yksi agentti kerrallaan.

Molemmilla tavoilla kuluu kaksi kellojaksoa, jolloin mikään lohko ei voi kirjoittaa väylälle. Aluksi vaihdetaan konfiguraatioarvoa tai -sivua ja seuraavalla kellojaksolla suoritetaan arbitrointi uuden konfiguraation mukaisesti. Kuva 2. esittää yhden konfigurointikirjoituksen ajoitusta, kun osoitteen siirto tapahtuu yhdessä kellojaksossa. Kellojaksolla 1 tapahtuu tavallinen datan kirjoitus ja toisella kellojaksolla väylälle ilmestyy komento *write config* ja konfiguroinnin osoite. Kellojaksolla kolme tulee varsinainen konfigurointidata. Kellojaksolla 4 liityntälohkot suorittavat komennon mukaisesti kofigurointiparametrin päivityksen, kellojaksolla 5 suoritetaan arbitrointi uusien konfigurointiparametrien avulla ja kellojaksolla 6 voi uusi siirto alkaa.

3.4.Arbitrointi

Arbitrointi ja osoitteen dekoodaus on molemmat hajautettu eli jokainen liityntälohko hoitaa ne itse. Koska väylällä ei ole yhtä keskitettyä arbitrointilohkoa, tulee varmistua että kaikki liityntälohkot ovat samassa vaiheessa ts. synkronoituja. Jos osa liityntälohkoista sammutetaan tehokulutuksen pienentämiseksi, täytyy liityntälohkot synkronoida, kun sammutetut lohkot kytketään uudestaan päälle.
Hajautuksen tavoitteena on, että järjestelmään voi liittää uusia agentteja ilman että aiempaan logiikkasuunnitteluun tai sijoitteluun (layout) tarvitsee tehdä muutoksia edellyttäen, etta väylän signaaleja ajavat puskurit ovat riittävän suuria. Ainoastaan liityntälohkojen parametrejä tarvitsee muuttaa.

Arbitroinnissa käytetään TDMA-pohjaista (Time Division Multiple Access) menetelmää. Menetelmässä määritellään aluksi *time frame*n eli kehyksen pituus. Kehyksen sisällä määritellään kellojaksot (aikaslotti), jolloin väylä oletusarvoisesti on tietyn agentin käytettävissä.
Esim. kehyksen pituus 40, aikaslottien määrittely 1-10(1), 11-15(2), 16-20 (1), 21-30 (3)

- Agentti 1 saa väylän jaksoilla 1-10
- Agentti 2 jaksoilla 11-15
- Agentti 1 uudestaan jaksoilla 16-20
- Lopuksi agentti 3 jaksoilla 21-30
- Kehys alkaa alusta jaksolla 31
- Agentti 1 uudestaan 10 jaksoksi, sitten agentti 2 jne.

Jos kellojaksolle ei ole määritelty omistajaa tai jos agentti luovuttaa TDMA-vuoronsa muille, väylän varaus tapahtuu kilpailemalla. Väylästä kilpailtaessa korkeimmalla prioriteetilla oleva agentti voi halutessaan varata väylän ensimmäiselllä kellojaksolla. Jos korkein prioriteetti ei varaa väylää, voi seuraavaksi korkein varata väylän toisella kellojaksolla jne. Jos agentti on keskeyttänyt tai jättänyt käyttämättä aikaslottinsa, mutta saa väylän kilpailemalla, siirrytään ’takaisin’ aikaslottivaraukseen. Aikaslotti jatkuu siitä kohtaa mistä kilpavaraus alkaa. Tällä tavalla agentti saa edes osan aikaslotistaan käyttöönsä, vaikka myöhästyisi aikaslotin alusta. Varausjärjestys määräytyy joko round-robin -periaatteella tai prioritettijärjestyksessä. Todellisessa prioriteettijärjestyksessä aloitetaan kilpailu aina korkeimmasta prioriteetista. Jos se ei varaa väylää, voi toiseksi korkein varata sen. Tämä tapa yksistään käytettynä johtaa helposti pieniprioriteettisten agenttien nälkiintymiseen, koska ne saavat väylän vain jos mikään korkeampiprioriteettinen ei sitä tarvitse. Round-robinissa käydään vuorotellen läpi kaikki prioriteetit ja täten jokainen agentti pääsee vuorollaan väylälle. Kyseessä on siis nk. reilu arbitrointi, jossa ei voi tapahtua nälkiintymistä. Tällöin agenttien prioriteeteillä ei ole periaatteessa mitään merkitystä, koska kaikki käydään vuorotellen läpi. Tällöin ongelmaksi voi muodostua, että kiireisten agenttien latenssi kasvaa, koska ne joutuvat odottamaan täyden kierroksen ennen kuin ne saavat väylän uudestaan. Round-robinissa on myös mahdollista tehdä pieni variaatio (*palauttava round-robin*), jolloin aina aikaslottien jälkeen kierros aloitetaan suurimmasta prioriteetista. Tämäkin tapa saattaa johtaa nälkiintymiseen, jos aikaslotteja on niin tiheästi että round-robin -kierros ei ehdi pieniin prioritetteihin asti ennen seuraavaa aikaslottia.

Prioriteettipohjaista kilpailua voi muuttaa reilummaksi myöntämällä pieniprioriteettisille agenteille aikaslotteja. Riippumatta siitä kumpaa kilpailutapaa käytetään, jokaiselle agentille määritellään aika (*MaxSend*), jonka se voi pisimmillään kilpailemalla varata väylää. Näin voidaan varmistua, ettei mikään agentti varaa väylää kohtuuttoman pitkään. Aikaslotit voivat kuitenkin olla pidempiä kuin *MaxSend* ja silti agentti voi siirtää koko aikaslotin ajan dataa. Kun kaikkien agenttien *MaxSend* ja aikaslottiasetukset on tiedossa voidaan laskea pahimman tapauksen viive, jonka tietty agentti joutuu odottamaan väylän saantia. Näin voidaan varmistua reaaliaikavaatimusten täyttymisestä.
Jos agentti on luovuttanut aikaslottinsa pois ja saa väylän kilpailemalla, siirrytään takaisin ko. lohkon aikaslottiin. Näin se voi loppuajan aikaslotista käyttää välittämättä *MaxSend*-arvosta, joka pitäisi kilpailuvarauksessa huomioida.

Kilpailemalla väylän haltijaksi tullut agentti joutuu kuitenkin luovuttamaan väylän, kun alkuperäinen vuoro loppuu (paitsi jos sille on varattu seuraava aikaslotti).
Kun agentti on saanut väylän haltuunsa, se ei vapauta sitä ennen kuin

- sen aikaslotti loppuu (tdma)
- seuraava aikaslotti alkaa (kilpailu)
- siltä loppuu siirrettävä data (mol. tavat)
- sille määritelty siirtoraja *MaxSend* täyttyy (kilpailu)

Toisin sanoen korkeampiprioriteettinen agentti ei voi kaapata väylää alemmalta prioriteetilta, vaan sen täytyy odottaa että matalaprioriteettinen ensin vapauttaa väylän.

3.5.Hierakiset väylärakenteet

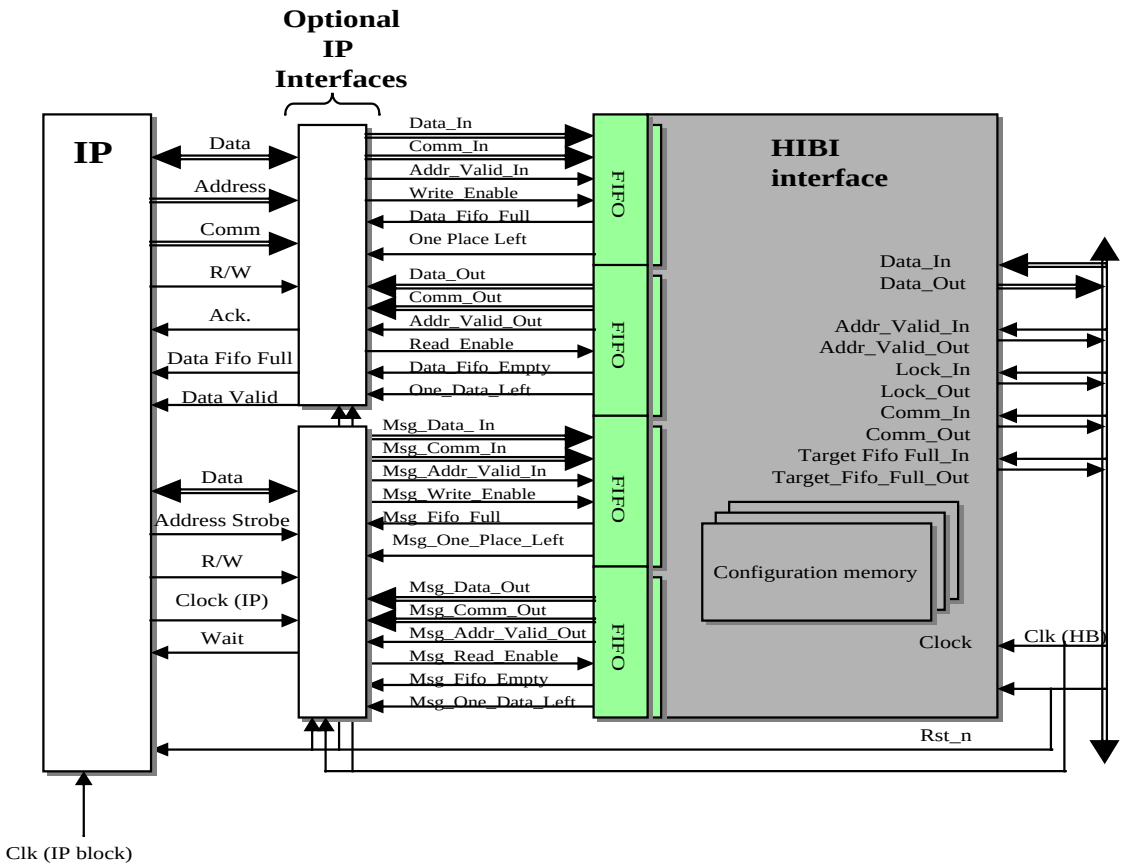
On mahdollista kytkeä monta hibiväylää toisiinsa silloilla. Tällöin jokainen väylä voi toimia itsenäisesti ja sillat ohjaavat tarvittaessa kirjoitukset toiselle väylälle. Data voi kulkea useamman kuin yhden sillan ylitse ennen kuin se pääsee kohteeseensa. Sillat toimivat molempiin suuntiin, jolloin jokainen järjestelmän agentti voi kirjoittaa mille tahansa agentille riippumatta mihin väylään lähde- ja kohdeagentti on kytketty. Myös järjestelmän konfigurointi yhdestä pisteestä on mahdollista. Kappale 5 kuvaa siltojen toiminnan ja rakenteen tarkemmin.

4. LIITYNTÄLOHKON RAKENNE

Jokaisella agentilla on rakenteeltaan samanlainen liityntälohko. Liityntälohkon sisällä olevien fifojen pituus ja leveys ja konfigurointimuiitiin talleteut arvot voivat tosin olla erilaiset eri agenteilla.

4.1.Rajapinnat

Kuva 3. esittää liityntälohkon rajapinnat. Vasemmalla puolella on esitetty IP-lohko ja oikealla hibirajapinta. Fifojen signaalit näkyvät suoraan liityntälohkon ulostuloissa ja ovat samat joka liityntälohkossa. Niihin voidaan kytkeä IP-lohko joko suoraan tai erillisen rajapintalohkon avulla. Esimerkiksi Kuva 3. esittää tilannetta, jossa viestejä käsitellään erilaisen rajapinnan avulla kuin dataa. Kuvassa datarajapinta mahdollistaa kaksisuuntaisen datalinjan käytön, mutta siitä seuraa että vain toista fifoa (joko Rx tai Tx) voidaan käsitellä kerrallaan.
Alempi viestirajapinta taas sopii esimerkiksi C6000-signaaliprosessorin laajennusväylään. Tämän tyyppisessä rajapintalohkossa täytyy tehdä sovitus kahden eri kellon välillä. Tässä esimerkissä prosessorille menevä *wait*-signaali hoitaa tarvittavan kättelyn. On myös mahdollista, että IP käyttää liityntälohkoa yhden ainoan rajapinnan kautta, joka hoitaa sekä viestien että datan välittämisen.
Hibiväylän rajapinta on esitelty Taulukkossa 6. IP-rajapinta esitellään Taulukkossa 17. (ks. liittteet) Koska fifopuskurit toimivat toisistaan riippumatta, voivat IP-lohkon luku- ja kirjoitusoperaatiot tapahtua yhtä aikaa. Lisäksi on mahdollista käsitellä viestejä ja dataa yhtä aikaa. Fifopuskurit rajapintoineen kuvataan luvussa 4.3.



Kuva 3. Liityntälohkon rakenne.

Taulukko 6: HIBI-väylän signaalit

Signaalin nimi	Leveys bitteinä	Merkitys	Aktiivinen
Addr_Valid	1	Kertoo onko dataväylällä osoite (1) vai data (0)	1
Clk	1	Kellolinja	1
Comm	3	Komento ks. Taulukko 5.	-
Data	8/16/32/64	Multipleksattu osoite ja data	-
Lock	1	Onko väylä varattu	1
Rst_n	1	Asynkroninen reset. Alhaalla aktiivinen.	0
Target_Fifo_Full	1	Kohteen (agentti/silta) Rx-fifo täynnä	1

4.2.Liityntälohkon sisäinen rakenne

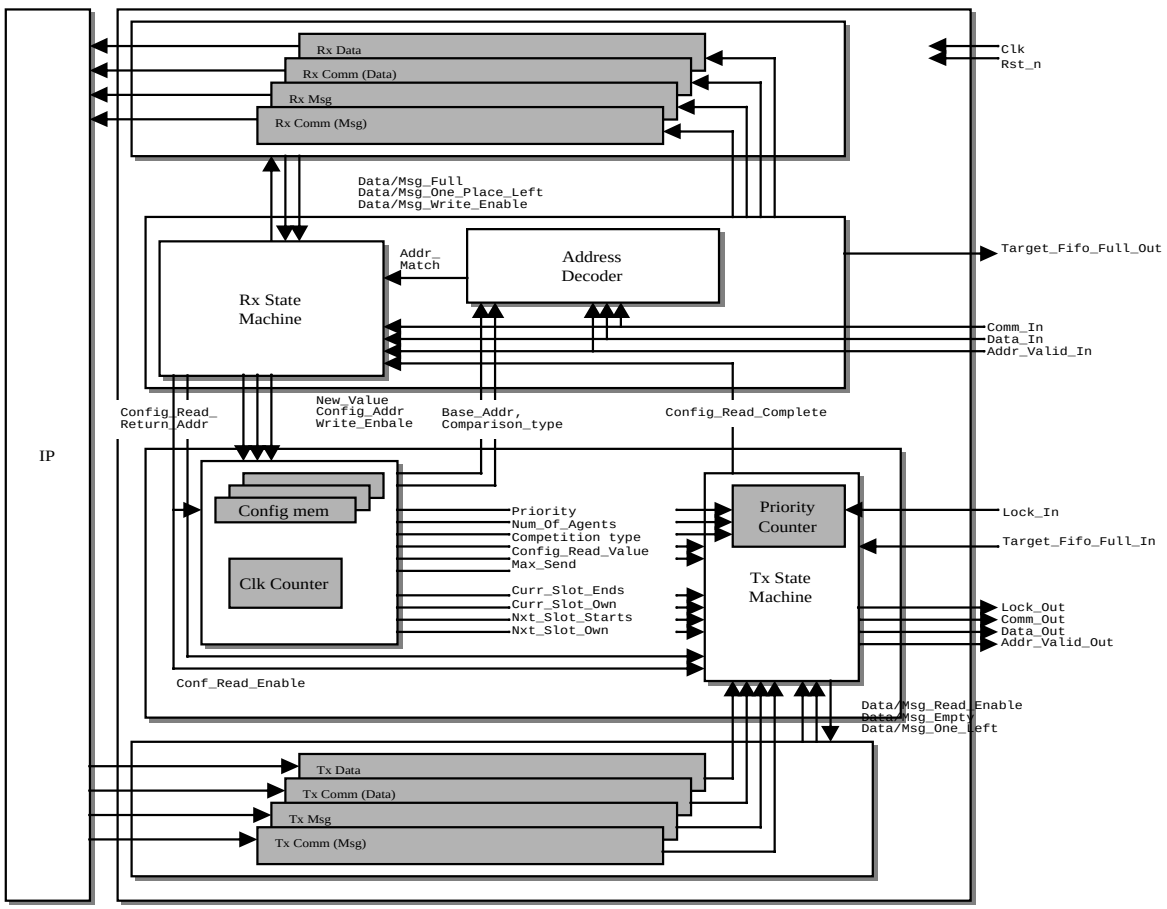
Liityntälohko koostuu 4 alilohkosta kuten Kuva 4. esittää. Lähetykselle ja vastaanotolle on omat lohkonsa. Lisäksi tarvitaan kaksi puskurilohkoa, jotka sisältävät fifopuskurit datalle ja viesteille. Kuvan yläosassa on vastaanotto- eli Rx-fifo ja sen alla vastaanottoa ohjaava lohko. Vastaanottolohko sisältää tilakoneen lisäksi asynkronisen osoitteen dekoodauslogiikan Alimmaisena näkyy lähetys- eli Tx-fifo ja sen yläpuolella lähetystä ohjaava lohko, joka sisältää tilakoneen lisäksi myös liityntälohkon konfigurointimuistin. Kaikki lohkot esitellään omissa kappaleissaan.

4.3.Fifopuskurit

Liityntälohkot sisältävät fifopuskurit datan, viestien ja komentojen puskurointiin. Kaikilla fifopuskureilla on samanlainen rajapinta, jonka signaalit ja niiden suunnat on listattu Taulukkossa 7. Datan ja viestien lähettämiseen

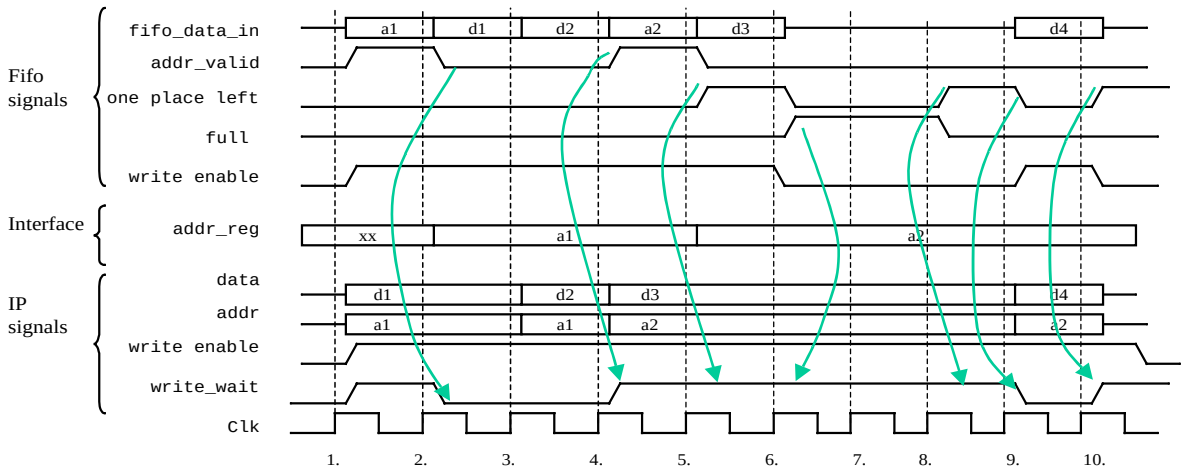
Taulukko 7: Fifopuskurin rajapinta

Nimi	Suunta	Puoli	Leveys
Clk	In	-	1
Rst_n	In	-	1
Data in	In	A	8/16/32/64
Write enable	In	A	1
Fifo full	Out	A	1
One place left	Out	A	1
Data out	Out	B	8/16/32/64
Read enable	In	B	1
Fifo empty	Out	B	1
One data left	Out	B	1



Kuva 4.Liityntälohkon rakenne.

käytetään fifojen A-rajapintaa ja vastaanottamiseen rajapintaa B. Data- ja viestipuskurien leveys on sama kuin IP-väylän dataleveys ja molemmat komentopuskurit ovat hibiväylän komentoväylän levyisiä. Hienona optimointina voi samaan puskuriin talletaa datan komennon ja address valid -tiedon. Tällöin tarvittava talletustila pysyy samana, mutta ei tarvita kuin yksi fifon ohajuslogiikka.
On mahdollista että vastaanotonpuskurit ovat kapeampia kuin hibiväylä ja tällöin sinne hibiväylältä luetaan vain alimmat tavut. Kun hibiväylälle kirjoitetaan kapeasta fifosta, ylimpiin bitteihin kirjoitetaan nollia. Lisäksi puskureihin sisältyy 1 bitin levyinen *AddrValid* -kenttä, joka kertoo onko fifon kyseiseen kohtaan talletettu data vai osoite. Puskurien pituus kiinnitetään synteesivaiheessa ja on vapaasti valittavissa (pituus kuitenkin vähintään 1).Data- ja viestififot voivat olla eripituisia. Fifot ovat täysin synkronisia.
Kuvassa 5. kuvataan ajoitus kun IP-lohko kirjoittaa fifoon. Kappaleessa 12.2 on kuvattu fifojen muut ajoitustapaukset. Fifoissa on neljä tilasignaalia. Signaali *Full* on aktiivinen, kun fifo on täysi. Jos yritetään kirjoittaa täyteen fifoon, sisääntuleva data jätetään täysin huomioimatta ja fifon tila pysyy ennallaan. Signaali *Empty* on aktiivinen, kun fifo on tyhjä. Jos yritetään lukea tyhjästä fifosta, on ulostulodatan arvo määrittelemätön ja fifon tila pysyy ennallaan. (Fifon rekistereitä ei nollata sieltä luettaessa. Tästä seuraa, että tyhjän fifon ulostuloissa saattaa olla jokin vanha arvo.) Tilasignaalit *One_Data_Left* ja *One_Place_Left*, kertovat koska fifossa on yksi data tai paikka jäljellä. Fifosta on mahdollista lukea samaan aikaan kuin sinne kirjotietaan. Tulee kuitenkin huomioida erikoistapaukset, jolloin fifon on joko tyhjä tai täysi. Jos tyhjästä fifosta luetaan ja sinne samaan aikaan kirjoitetaan, on luetty arvo jokin vanha arvo ja siten määrittelemätön. Tämän jälkeen fifossa on yksi data. Kirjoitus onnistuu normaalisti. Täyden fifon tapauksessa, lukeminen onnistuu ja kirjoitettu arvo jätetään huomiotta. Tämän jälkeen fifossa on yksi tyhjä paikka.



Kuva 5. Esimerkki fifopuskurin ajoituksesta, IP-lohko kirjoittaa fifoon.

Kun käytetään IP-lohkoa, joka toimii eri kellolla kuin hibiväylä (ja liityntälohkot), tarvitaan erillinen rajapinta kahden kellodomainin välille (ks Kuva 3.).
Koska osoitteelle ei ole varattu omaa fifoa, täytyy osoitteet tallettaa samaan fifoon datan kanssa. Fifoon talletetaan kuitenkin vain yksi osoite kutakin datalohkoa kohti. Datalohko voi olla useamman sanan mittainen. Kun agentti siirtää suuren datalohkon toiselle agentille, se saattaa joutua *MaxSendi*sta tai arbitroinnista johtuen kirjoittamaan lohkon pienemmissä paloissa. Tätä varten on olemassa vastaanottavan fifon edessä MUX-lohko, joka päättelee, koska osoite pitää kirjoittaa fifoon. Osoitetta ei tarvitse kirjoittaa fifoon, jos mikään muu agenti ei ole kirjoittanut samalle agentille kahden siirron välissä. Vastaavan tyyppinen MUX-lohko on myös fifon IP-puolella.

Esim. agentti A siirtää 50 dataa, 5 datan paloissa agentille B. Jos kahden peräkkäisen 5 datan lohkon välissä joku muu lohko kuin A kirjoittaa myös lohkolle B, pitää lohkon A kirjoittama osoite kirjoittaa uudestaan B-fifoon (luonnollisesti myös välissä tullut osoite ja siihen liittyvä data on talletettu fifoon). Jos A:lta tulevat lohkot tulevat peräkkäin, ei osoitetta tarvitse joka kerta tallettaa. Jotta voidaan välttää sekaannus kahdesta eri lähteestä saapuvan datan välillä, tulee huolehtia, että agentin A lisäksi muut eivät kirjoita samaan agentin B osoitteeseen.

Tilan säästämiseksi voidaan käyttää liityntälohkoa, jossa ei ole omaa puskuria viesteille vaan sekä data että viestit talletetaan samaan puskuriin. Tällöin luonnollisesti menetetään viestien nopeushyöty, mutta säilytetään yhteensopivuus. Agentti voi lähettää viestejä, mutta agentin oma liityntälohko ei käsittele niitä viesteinä. Sen sijaan vastaanottava agentti ja mahdolliset sillat kahden agentin välissä käsittelevät viestit normaalisti. Vastaavasti viestit voidaan myös vastaanottaa, vaikka ei olisikaan erillistä viestipuskuria. Molemmissa tapauksissa liityntälohkon IP-rajapinta muuttuu (toinen tai molemmat viestiportit puuttuvat), mutta puskurilohkon rajapinta lähetys- ja vastaanottolohkoihin pysyy samana. Tästä seuraa, että näihin lohkoihin ei tarvitse tehdä muutoksia vaikka viestiominaisuuksia karsittaisiinkin. Tämän toteuttaminen vaatinee kuitenkin kaksi ylimmän tason arkkitehtuuria.

4.4. Lähetyslohko

Lähetyslohkon signallit on kuvattu. Taulukkossa 8. ja lähetyksen tilakoneen signaalit on kuvattu Taulukkossa 9. Lohko sisältää kaksi rekisteriä. Osoiterekisteri on saman levyinen kuin levein järjestelmässä sallittu osoite. Lisäksi lohkossa on väliaikainen rekisteri datalle. Kuva 6. esittää lähetyslohkon tilakonetta kun osoitteen pituus on yksi. Mahdolliset tilasiirtymät on lisäksi lueteltu Taulukkossa 10.

Lähetyksen tilakone kun pitkät osoitteet käytössä

Ylimpänä näkyy tila *PopAddrFromFifo*, joka on lohkon perustila. Tässä tilassa siirretään kaikki peräkkäiset osoitteen palaset fifosta osoiterekisteriin. Kun koko osoite on siirretty, luetaan fifossa oleva data sille varattuun väliaikaiseen

Taulukko 8: Lähetyslohkon signaalit

In	Leveys	Out	Leveys
Clk	1	Conf_Read_Complete	1
Rst_n	1	Comparison_Type	1
Conf_Read_Enable	1	Base_Addr	n * (8/16/32/64)
Conf_Write_Enable	1	Data_Out	8/16/32/64
Conf_Addr	n * (8/16/32/64)	Addr_Valid_Out	1
Conf_Return_Addr	n * (8/16/32/64)	Comm_Out	3
Conf_New_Value	8/16/32/64	Lock_Out	1
Target_Fifo_Full_In	1	Read_Enable	1
Lock_In	1	Msg_Read_Enable	1

Taulukko 10: Lähetyslohkon mahdolliset tilasiirtymat

Tila	Idle	Slot Res	Re tx A	Re tx D	Re tx Last	Config A	Config D	Config Last	Write	Write Last
Idle	i5	i1	i2	-	-	i3	-	-	i4	-
Own Slot	o1	-	o2, o4	-	-	o3	-	-	o5	-
Re tx A	-	-	-	ra4	ra3	-	-	-	ra2	ra1
Re tx D	rd1	-	-	rd4	rd2	rd3	-	-	rd5	-
Re tx Last	aina	-	-	-	-	-	-	-	-	-
Config A	-	-	-	-	-	-	ca2	ca1	-	-
Config D	cd1	-	cd2	-	-	-	-	-	cd3 ??	-
Config Last	aina	-	-	-	-	-	-	-	-	-
Write Data	w1	-	-	-	-	w4	-	-	w3	w2
Write Last	aina	-	-	-	-	-	-	-	-	-

rekisteriin. Samalla talletetaan luetun osoitteen pituus omaan rekisteriin. Osoitteiden kokoaminen tehdään sekä datalle että viesteille Jos lohkolle on varattu seuraava aikaslotti ja jommassa kummassa fifossa (tai reksiterissä) on dataa, siirrytään tilaan *OwnSlotReservation*. Tässä tilassa asetetaan ainoastaan *Lock*=1 ja siirrytään automaatisesti tilaan *WriteAddr*. Tilaan *WriteAddr* voidaan tulla myös suoraan perustilasta, jos mikään lohko ei varaa väylää, liityntälohkon prioriteetti on seuraavana vuorossa ja sillä on lähetettävää dataa. Ennen siirtymistä kirjoitustilaan tulee varmistua, että agentilla on aikaa (aikaslottiasetukset + MaxSend) lähettää koko osoite ja vähintään yksi data/viesti.

Taulukko 9: Lähetyksen tilakoneen signaalit

In	Leveys	Out	Leveys
Clk	1	Comm_Out	3
Rst_n	1	Lock_out	1
Empty	1	Addr_Valid_Out	1
One_Data_Left	1	Data_Out	8/16/32/64
Msg_Empty	1	Read_Enable	1
Msg_One_Data_Left	1	Msg_Read_Enable	1
Config_Read_Value	8/16/32/64	Config_Read_Complete	1
Config_Return_Addr	n *(8/16/32/64)		
Config_Read_Enable	1		
Priority			
Num_Of_Agents			
Max_send			
Competition_Type	2		
Next_Slot_own	1		
Next_Slot_starts	1		
Curr_slot_own	1		
Curr_slot_Ends	1		

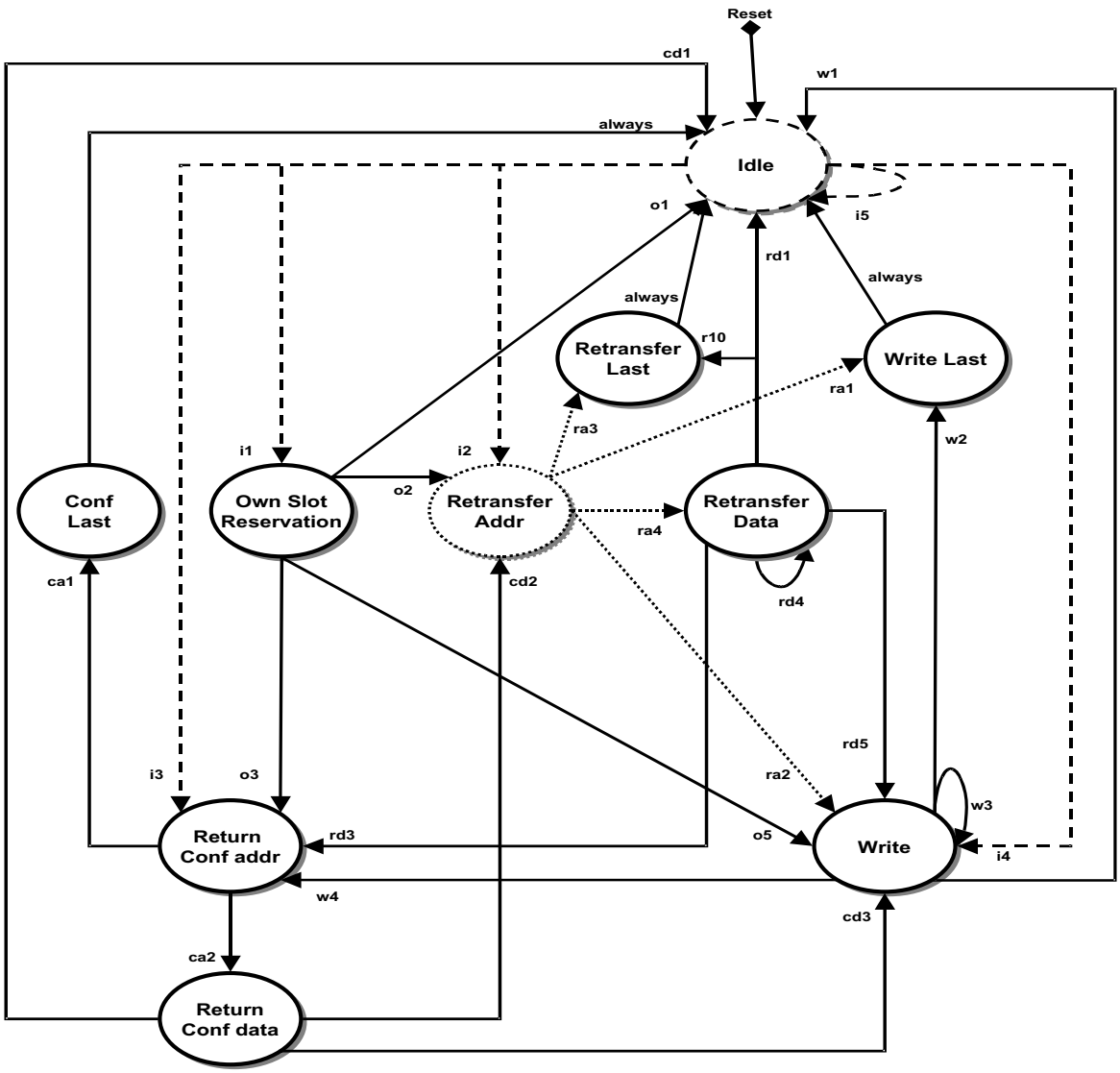
Tilassa *WriteData* kirjoitetaan osoite palanen kerrallaan väylälle ja pidetään *Lock* ja *AddrValid* aktiivisina. Tässä tilassa pysytään kunnes koko osoite ja fifossa ollut data on siirretty. Data kirjoitetaan väylälle väliaikaisesta datarekisteristä ja samalla luetaan rekisteriin seuraava arvo fifosta. Aluksi siirretään viestififossa olevat datat ja sen jälkeen datafifossa. Kun siirretään 1. data, asetetaan *AddrValid* -signaali ei-aktiiviseksi. Tilasta *WriteData* siirrytään tilaan *WriteLastData*, jos oma aikaslotti loppuu, jonkun toisen agentin aikaslotti alkaa tai lähetysraja täyttyy. Jos vastaanottajan puskurista loppuu tila, siirrytään suoraan tilaan *PopAddrFromFifo,* jossa vapautetaan väylä.
Jos oltaessa *WriteData* -tilassa fifossa on osoite, kirjoitetaan se osoitteenkoostamisrekisteriin ja väylälle kirjoitetaan data väliaikaisesta rekisteristä. Jos fifossa on seuraavanakin osoite, luetaan se koostamisrekisteriin. Samalla luovutetaan väylä (*Lock* =0) ja siirrytään perustilaan. Jos osoite oli nk. normaali (1*väylän levyinen), voidaan datan siirron jälkeen siirtää osoite ilman katkoa. Jos datalle varattaisiin leveimmän mahdollisen osoitteen levyinen väliaikainen puskuri, voitaisiin siirtoa jatkaa keskeytyksettä myös, kun kesken siirron tulee fifosta leveä osoite. Ainakin alussa datarekisterin on vain yhden datan kokoinen toteutuksen helpottamiseksi.
Jos huomataan, että oma aikaslotti on loppumassa, jonkun muun aikaslotti on alkamassa, lähetysraja täyttyy tai että vastaanottajan fifo on täyttymässä siirrytään tilaan *WriteLastData*. Siinä siirretään viimeinen data väylälle ja vapautetaan väylä (*Lock* =0) ja siirrytään perustilaan.

Lähetyksen tilakone, kun osoitteen pituus on yksi

Lähetyksen tilakone helpottuu huomattavasti, jos käytössä on vain tavalliset osoitteet. Tällöin perustilana on Idle.
lpapalapalapa...

4.5. Konfigurointimuisti

Muistin signaalit on lueteltu Taulukkossa 11. Konfigurointimuisti koostuu monesta sivusta, joihin kuhunkin voi tallettaa yhden konfiguraation (Taulukko 12.). Taulukko 12. :n pituus riippuu aikaslottien määrästä. Esim. kahdella aikaslotilla taulukon koko on 10+2*3=16 parametriä, kolmella aikaslotilla 10+3*3=19 parametriä jne.
Parametrit ovat oletuksena dataväylän levyisiä. Näin muistinrakenne yksinkertaistuu, vaikka samalla tosin haaskataan tallennustilaa, kun kaikki parametrit eivät tarvitse läheskään niin paljoa tilaa. Tarvittaessa hibi2 tukee kuitenkin osoitteita, jotka ovat dataa leveämpiä. Tällöin tulee osoitteen tallettamiseen varataan useampi peräkkäinen muistipaikka.
Jotta aikaslotteja voidaan asettaa vapaasti, täytyy kellojaksolaskuria verrata rinnakkain aikaslottimuistin kaikkiin aikaslotteihin, jotta saadaan selville minkä lohkon aikaslotti on parhaillaan voimassa. Rinnakkainen vertailu vienee reilusti pinta-alaa, mutta kellojakson vapaa asettaminen katsotaan riittävän suureksi hyödyksi.
On mahdollista tallettaa useita konfiguraatioita. Tällöin konfiguraatiomuistia voi ajatella 2-ulotteisena taulukkona. Tallettamalla useita konfiguraatioita, niiden vaihtaminen käy nopeasti yhdellä konfigurointioperaatiolla. Lisäksi uusi



Kuva 6. Lähetyslohkon tilakone, kun osoitteen pituus on 1.

konfiguraatio voidaan kirjoittaa sivulle, joka ei ole käytössä. Kun sivu on valmis, voidaan kaikki parametrit muuttaa uusiin sivua vaihtamalla. Suurena muutoksena aiempaan hibiin, konfigurointiparameterejä voi myös lukea ulkopuolelta *ConfigRead*-käskyllä. Tällöin luettu arvo talletetaan lähetyslohkon sisälle ja lähetetään pyydettyyn osoitteeseen, kun agentti seuraavan kerran saa väylän haltuunsa. Pitkät osoitteet täytyy lukea monessa osassa jokainen osa erikseen. Konfigurointimuistiin voi myös tallettaa sovelluskohtaisia parametrejä. Liityntälohko ei käytä niitä mihinkään, mutta niihin voi tallettaa arvoja ja niitä voi lukea. Näiden parametrien lukumäärä päätetään synteesissä.

Taulukko 11: Konfiguraatiomuistin signaalit

In	Leveys	Out	Leveys
Clk	1	Read_Value	8/16/32/64
Rst_n	1	Base_Addr	n (*8/16/32/64)
Read_Enable	1	Num_Of_Agents	
Write_Enable	1	Max_Send	
Conf_Addr	n *(8/16/32/64)	Priority	
New_Value	8/16/32/64	Next_Slot_Starts	1
		Next_Slot_Own	1
ID (generic-parametri)	ID_Width	Curr_Slot_Ends	1
Base_ID (generic-parametri)	ID_Width	Curr_Slot_Own	1
		Comparison_Type	1
		Competition_Type	2

Taulukko 12: Konfigurointimuistin rakenne

Parametrin nro	Parametrin nimi	Mahd. arvoja	Merkitys
Sivu = 0, param nro = 0	Page number	2^Page_Addr _Width	Käytössä olevan parametrisivun numero. Talletettu vain yhteen kertaan.
Sivu = 0, param nro =1	ID	2^ ID_Width	Liityntälohkon tunnus. Vakioarvo (generic-parametri)
Sivu = 0, param nro =2	Base_ID	2^ ID_Width	Osoittaa silloilla ID-avaruutta, tavallisilla lohkoilla 0xff...f. Vakioarvo (generic-parametri)
	-		*-Seuraavat kentät ovat joka sivulla erikseen-*
0	Clock cycle counter	Max_Frame	Kellojakson numero aikakehyksen sisällä. Voidaan asettaa vapaasti synkronointia varten.
1	Priority	Num_of_A	Lohkon prioriteetti kilpailutilanteessa.
2	Number of agents	Num_of_A	Agenttien lukumäärä. Tarvitaan kilpailutilanteessa.
3	Competition style	3 kpl	Prioriteetit, round-robin vai palauttava round-robin käytössä
4 (oli 6)	Power state	3 kpl	Kertoo missä tilassa lohko on *Full power/Slowed/ Off*. Tieto välitetään myös IPlle.
5 (oli 7)	MaxSend	Max_Frame?	Montako kellojaksoa lohko voi pitää väylää varattuna kilpailutilanteessa.
6 (oli 8)	Time frame length	Max_Frame	Toistettavan kehyksen pituus kellojaksoina. Kehys on jaettu aikaslottien ja kilpailun kesken
7 (oli 5)	Addr_Comparison	2 kpl	Silloilla voi olla joko Normal/Negated. Liityntälohkoilla ainoastaan Normal.
8 (oli 4)	Base address		Lohkon osoiteavaruuden ylimmät bitit.
8+Addr_Width	Slot #1 Start Cycle	Max_Frame	Ensimmäisen aikaslotin alku
+1	Slot #1 End Cycle	Max_Frame	Ensimmäisen aikaslotin loppu
+1	Slot #1 Owner	Num_of_A	Ensimmäisen aikaslotin haltija
+1	Slot #2 Start Cycle	Max_Frame	Toisen aikaslotin alku
	...		
8+ Addrw + (n-1)3 -1	Slot #n Owner	Num_of_A	N.aikaslotin haltija
8+ AddrW + (n-1)3	Other 1		Riippuu sovelluksesta.
	...		
+1	Other m		Riippuu sovelluksesta.

4.6. Vastaanottolohko

Vastaanottolohko koostuu tilakoneesta ja osoiteenvertailulohkosta. Vastaanottolohkon signaalit on lueteltu Taulukkossa 13. Tilakoneesta on kaksi eri versiota, pitkille ja tavallisille (yhden levyisille) osoitteile. Molemmilla tilakoneilla on samanlainen rajanpinta, joka on esitetty Taulukkossa 14.

Tilakoneen toiminta pitkillä osoitteilla

Kuva 7. esittää pseudokoodina dataa vastaanottavaa lohkoa. Kun väylällä ilmestyy osoite, luetaan se osoitteenkoostamisrekisterin alimpiin bitteihin. Jos seuraavallakin jaksolla väylällä on osoite, luetaan se rekisterin seuraaviin bitteihin. Näin jatketaan kunnes väylälle ilmestyy data (eli *Addr_Valid = 0*). Jos väylä jostain syystä vapautuu osoitteen siirtämisen jälkeen, hylätään osoiterekisteriin kerätty osoite.
Kun väylälle ilmestyy data, vertaillaan osoiterekisteriin kerättyä osoitetta liityntälohkon *BaseAddr*:iin. Jos osoite ei täsmää, ei tarvitse tehdä mitään ja jäädään odottelemaan seuraavaa osoitetta. Jos osoite täsmää, siirretään ensimmäisenä luettu osoitteen palanen (koosterekisterin alimmat bitit) fifoon. Samalla luetaan data väylältä väliaikaiseen siirtorekisteriin. Seuraavilla kellojaksoilla siirretään aina yksi osoitteen palanen fifoon ja luetaan data siirtorekisteriin ja siirretään siellä ennestään olleita datoja yksi pykälä eteenpäin. Kun koko osoite on siirretty fifoon, aletaan siirtää dataa siirtorekisteristä fifoon. Jos väylältä edelleen saapuu dataa, luetaan sitä siirtorekisteriin sisään. Jos väylälle ilmestyy uusi osoite ennen, kuin edellinen osoite on kokonaan siirretty fifoon, luetaan uusi osoite tavalliseen tapaan osoiterekisterin alimpaan kohtaan. Osoitteenvertailulohkolle sen sijaan ohjataan ainoastaan juuri luettu osoitteen pala ja muut osoitteenvertailulohkolle menevät bitit nollataan. Osoiterekisterin ylimpiä bittejä ei kuitenkaan nollata, koska siellä on vielä edellistä osoitetta jäljellä.
Osoitteenkoostamisrekisterin on oltava yhtä leveä kuin levein sallittu osoite järjestelmässä (n*väylän leveys). Vastaavasti datalle varatun siirtorekisterin tulee olla saman kokoinen ylivuotojen välttämiseksi (n *väylän leveys). Siirtorekisterin nimitys saattaa olla harhaanjohtava, koska dataa saatetaan siirtää fifoon mistä tahansa rekisterin

Taulukko 13: Vastaanottolohkon signaalit

In	Leveys	Out	Leveys
Clk	1	Target_Fifo_Full_Out	1
Rst_n	1	Conf_Write_Enable	1
Comm_In	3	Conf_New_Value	8/16/32/64
Addr_Valid_In	1	Conf_Addr	n *(8/16/32/64)
Data_In	8/16/32/64	Conf_Read_Enable	1
Base_Addr	n *(8/16/32/64)	Write_Enable	1
Conf_Read_Complete	1	Msg_Write_Enable	1
Comparison_Type	1		
Full	1		
One_Place_Left	1		
Msg_Full	1		
Msg_One_Place_Left	1		

Taulukko 14: Vastaanoton tilakoneen signaalit

In	Leveys	Out	Leveys
Clk	1	Target_Fifo_Full_Out	1
Rst_n	1	Write_Enable	1
Addr_Match	1	Msg_Write_Enable	1
Base_ID_Match	1	Config_Write_Enable	1
Comm_in	3	Config_Read_Enable	1
Addr_Valid_in	1	Config_New_Value	8/16/32/64
Data_in	8/16/32/64	Config_Return_Addr	n *(8/16/32/64)
Config_Read_complete	1	Config_Addr	n * (8/16/32/64)
Full	1	Comm_To_Decode	3
One_Place_Left	1	Addr_To_Decode	n * (8/16/32/64)
Msg_Full	1		
Msg_One_Place_Left	1		

alkiosta (eikä pelkästään toisesta päädystä) riippuen siitä miten pitkä osoite siirtoon kuului. Jos kyseessä oli pisin mahdollinen osoite, tulee väliaikainen datarekisterikin täyteen, ennen kuin osoite saatiin fifoon.

Tilakoneen toiminta kun osoitteet ovat yhden sanan levyisiä

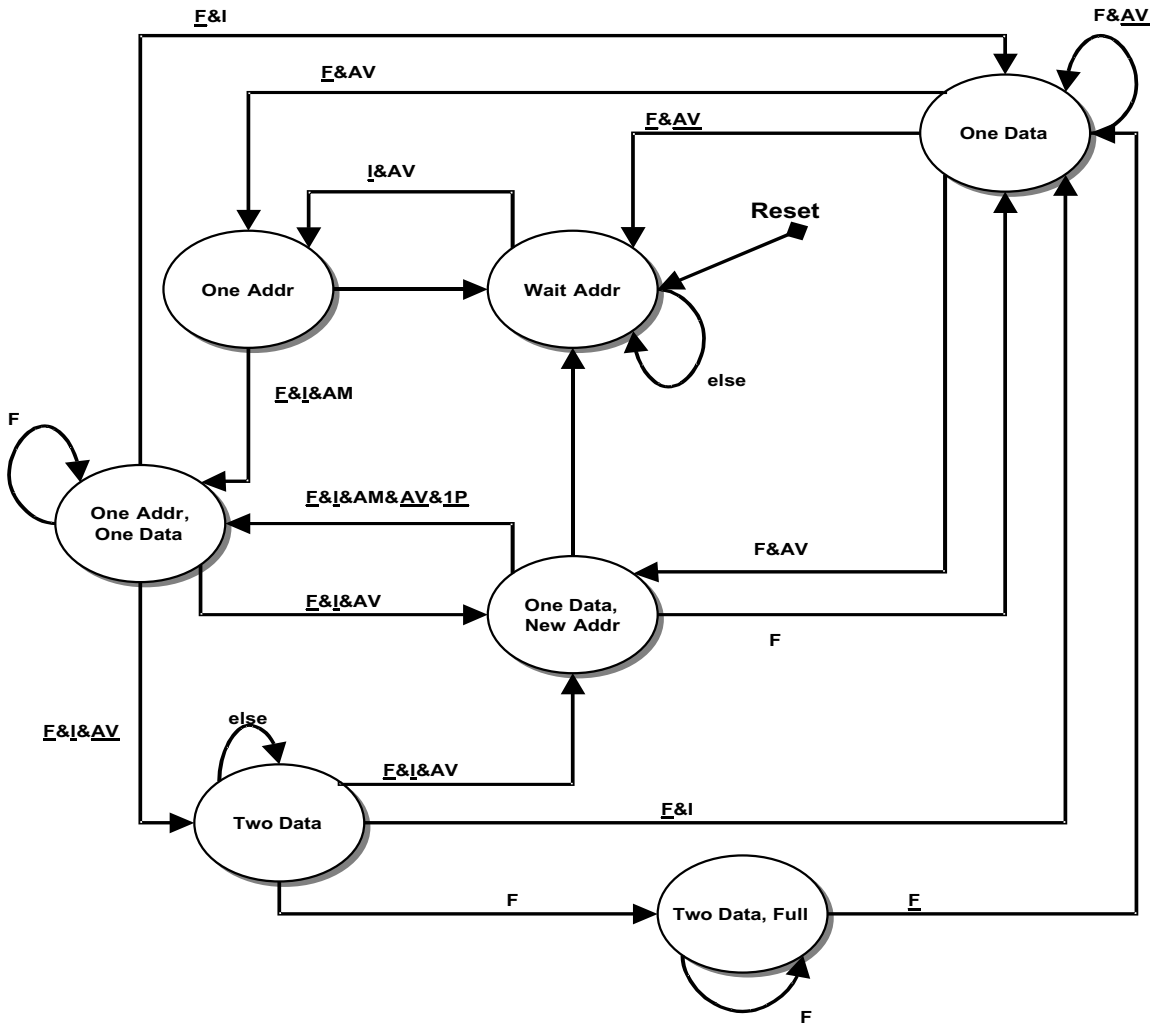
Tilakoneen toiminta helpottuu huomattavasti, jos käytettävät osoittet ovat yhden sana levyisiä (tavallisia verattuna pitkiin osoitteisiin). Kuva 8. esittää kyseisen tilakoneen toiminnan.Kannattaa huomata, että data ja viestit käsitellään samoissa tiloissa. Erottelu tehdään kyseisen tilan sisällä.
Aluksi tilakone on tilassa *Read Addr* ja pysyy siinä kunnes väylälle ilmestyy osoite. Jos komento on konfiguroinnin kirjoitus tai luku, siirrytään tilaan *Reconfiguration*. Muilla komennoilla, siirrytään tilaan *Write Addr.*
Tilasta *Write Addr* siirrytään aina pois seuraavalla kellojaksolla. Jos osoite ei täsmännyt tai fifo on täynnä tai täyttymässä, siirrytään tilaan *Read Addr.* Muutoin siirrytään tilaan *Read Write Data*
Tilasta Read Write Data siirrytään tilaan Read Addr, jos tulee uusi osoite joka ei täsmää tai datan tulo loppuu, tai fifo on täynnä tai täyttymässä. Jos tulee uusi täsmäävä osoite ja komento on joko konfiguroinnin kirjoitus tai luku, siirrytään tilaan *Reconfigutation*. Jos kumpikaan edellisistä ehdoista ei täsmää pysytään samassa tilassa.
Tilasta Reconfiguration siirrytään tilaan Read Addr, jos osoite ei täsmää tai datan tulo loppuu. Tilaan Write Addr siirrytään, jos tulee uusi täsmäävä osoite ja komento on datan tai viestin käsittelyä. Muutoin pysytyään samassa tilassa.

```
SYNKR process
if Comm == 0
     väylä vapaa, ei tehä mitn
else
     if AddrValid ==1
          -- Väylällä osoite, luetaan talteen
          AddrTmp (i) <= BusDataIn
          i <= i+1
          j <= 0 l <= 0 k <= 0
     else
          --i <= 0 -- Seur osoite alkaa alusta
          if AddrTmp matches BaseAddr
               -- Osoite täsmää
               if j < i
                    -- Osoitetta ei ole vielä kokonaan kirjoitettu fifoon
                    FifoIn <= AddrTmp (j)
                    WE_Fifo <= 1
                    j++
                    l <= k
               else
                    -- Joten talletetaan datakin fifoon
                    FifoIn <= DataTmp (l)
                    l <= (l+1) mod MaxAddrLengthInBytes
               end if

               -- Luetaan data väliaikaseen paikkaan, että
               -- ehditään kirjoittaa osoite kok, fifoon
               DataTmp (k) <= Data
               k <= (k+1) mod MaxAddrLengthInBytes


          else
               Osoite ei täsmää, ei tehä mitn
          end if
     end if
end if
```

Kuva 7.Vastaanottolohkon toiminta.



Kuva 8.Vastaanoton tilakone, kun osoitteen pituus on yksi sana.

Tilakoneen muu toiminnallisuus

Fifo, johon saapunut data ohjataan, määräytyy väylällä olleen komennon perusteella. Viestit (*Write_Message* ja *Multicast_Message*) ohjataan viestififoon, kaikki loput datafifoon. Konfigurointikomennot ohjataan fifon sijasta konfigurointimuistilohkolle.

Jos vastaanottofifo on täynnä, *Target_Fifo_Full* -signaali on aktiivinen eli ylhäällä. Kun *Target_Fifo_Full* on aktiivinen, vastaanottaja ei pysty lukemaan dataa, ja lähettäjän pitää lähettää se uudelleen, kun se seuraavan kerran saa väylän haltuunsa. Jos kyseessä oli multicast-operaatio, se keskeytetään jos yhdenkin kohteen fifo tulee täyteen.

Kun vastaanotetaan käsky Conf_Write tai Conf_Read, mukana seuraavasta osoitteesta vertaillaan ID-kenttää. Kirjoituksen tapauksessa osoitetta seuraava data on uusi konfiguraatioarvo. Parametriä luettaessa, se tulkitaan kohdeosoitteeksi, johon luettu arvo lähetetään. Kirjoituksessa vastaanottolohko asettaa konfiguraatiomuistille menevät signaalit *Write_Enable*, *Conf_Addr* ja *New_Value* aktiivisiksi. Jos liityntälohkosta luetaan konfiguraatiota, se asettaa signalit *Conf_Read_Enable*, *Conf_Return_Addr* ja *Conf_Addr* aktiiviseksi, jos *Conf_Read_Complete* on aktiivinen. Tällöin lähetyslohko tallettaa konfiguraatiomuistin parametriarvon ja paluuosoitteen omiin rekistereihinsä ja asettaa ulostulonsa *Conf_Read_Complete* ei-aktiiviseksi. Tämä signaali pysyy ei-aktiivisena kunnes luettu konfiguraatioarvo on lähetetty pyydettyyn osoitteeseen. Kerrallaan voi siis olla vain yksi konfiguraatioluku kesken. Kun tulee konfiguraation lukupyyntö ja edellistä luettua konfiguraatioarvoa ei ole viela ehditty lähettää (*Conf_Read_Complete* = 0), Target_Fifo_Full asetetaan aktiiviseksi. Jos joku lohko siis lähettää toisen konfiguraatiolukupyynnän, kun edellinens on vielä kesken, se keskeytyy *Target_Fifo_Full*iin.

4.7. Osoitteen vertailu

Osoitteenvertailulogiikka vertailee kuuluuko väylän osoite liityntälohkon osoiteavaruuteen Lohkon signaalit on esitelty Taulukkossa 15. Datan ja viestien siirrossa osoitteen dekoodauslogiikka vertailee konfigurointimuistiin talletettua bittikuviota saapuvan osoitteen ylimpiin bitteihin. *Base_Addrin* alin ykkösbitti määrää vertailtavien bittien määrän. Kun osoitteen koodaus on huomannut sopivan osoitteen se nostaa asynkronisen ulostulosignaalin *Addr_Match* ylös. *Addr_Match* nousee siis samalla kellojaksolla, kun ensimmäinen agetille tuleva data on väylällä.

Esim. Jos 8b osoitteen bitti nro 4 on yksi ja kaikki alimmat ovat nollia (vaikkapa 0xDA70), verrataan bittejä siitä ylöspäin (bitit 4-7). Tällöin osoiteavaruus on 4b=16 osoitetta (0xDA70-0xDA7F).
Agentin osoiteavaruude koko on siis 2^ (alimman ykkösen indeksi). Yllä olevassa esimerkissä siis, 2^4 =16 osoitetta.

Multicastissa vertailtavien bittien määrä kerrotaan osoitteen kahdella alimmalla bitillä. Vertailuun otetaan joko puolet (alimmat bitit 00) , neljäsosa (01), kahdeksasosa (10) tai kuudestoistaosa (11) bittiä.
Esim1. Väylän leveys 32b. Mutlicast 00 => vertailuun bitit (31-16), 01 => (31-24), 10 => (31-28), 11 => (31-30).
Esim2. Jos väylän leveys on ainoastaan 8 bittiä, multicast 10 vertailee vain ylintä bittiä. Sen sijaan multicast 11 vastaa toiminnaltaan broadcastia, koska silloin ei vertailla yhtäkään bittiä. 00 => bitit (07-04), 01 => (07-06), 10 => (07), 11 => (-).
.
Konfiguroinnissa osoitteen koodaus toimii eri tavalla. Vastaanottolohkon generic-parametreinä määritellään suurin mahdollinen agenttien määrä yhdessä segmentissä, konfigurointisivujen ja parametrien määrä. Näistä saadaan määriteltyä, montako bittiä konfigurointiosoitteesta tarvitaan millekin kentälle. Kenttä osoitteen alussa kertoo kohteen ID-numeron. Jos konfigurointiosoite on useamman kellojakson pituinen, siirretään ensimmäisenä ID-kenttä. Jos ID on 0, kyseessä on broadcast-operaatio, jonka kaikki agentit ottavat vastaan.

PITKÄT OSOITTEET JA ERILEVYISET AGENTIT JA SEGMENTIT
Jotta voidaan samassa järjestelmässä käyttää erilevyisiä väyliä, täytyy leveämmän väylän osoite siirtää monessa palassa kapean väylän läpi. Samalla periaatteella voi yksittäisessä väylässä käyttää useamman kellojakson pituisia osoitteita.

Esim. 1) 32b väylältä kirjoitetaan 16b väylälle => 32b osoite kirjoitetaan kahtena 16b osoitteen puolikkaana. Sen sijaan 32b datasta siirretään ainoastaan 16 alinta bittiä ja ylimmät bitit hylätään kylmän viileästi.

Taulukko 15: Osoitteen vertailulohkon signaalit

Nimi	Leveys	Suunta
Base_Addr_In	n *(8/16/32/64)	In
Addr _in	n * (8/16/32/64)	In
Addr_Valid	1	In
Comm	3	In
Comparison_Type	1	In
Addr_Match	1	Out
Base_ID_Match	1	Out
ID	ID_Width	"in"(generic-parametri)
Base_ID	ID_Width	"in"(generic-parametri)

Esim. 2) 16b väylältä kirjoitetaan 32b väylälle => kaksi peräkkäistä 16b osoitteen puolikasta yhdistetään sillassa yhdeksi 32b osoitteeksi. Jos tulee vain yksi 16b osoite, se täytetään nollilla 32b osoitteeksi. Sen sijaan 16b data siirretään aina nollilla täytettynä eikä sitä ruveta yhdistelemään. Tämä johtuu siitä että ei ole mitään takeita, että dataa tulisi sopiva tasamäärä (tässä tapuksessa parillinen määrä) ja koska ei ole tavunosoitusta (byte enable) käytössä.

Lähettävässä päässä tarvitaan leveimmän mahdollisen osoitteen levyinen rekisteri, johon osoitteet luetaan fifosta. Ensimmäisenä fifosta luettu osoite muodostaa lähetettävän osoitteen alimmat bitit, seuraavaksi luettu seuraavat jne. Osoitteita kerätään kunnes fifossa on tulossa seuraavana dataa. Sen jälkeen rekisteriä puretaan väylän levyisissä kaistaleissa. Jos osoitekaistale ei tullut täyteen, se täytetään nollilla väylän leveyteen asti. Jos rekisteri ei tullut täyteen (lähetettävä osoite ei ole levein mahdollinen), siitä tarvitsee lähettää vain sinne luettu osuus (nollilla täydennettynä). Jos datan siirto jostain syystä katkeaa, voidaan siirtoa jatkettaessa lukea osoite rekisteristä.

Esim3) 32b väylä, maksimi osoitteen leveys 128b, lähettävä agentti 16b. Lähettäjän fifossa on 3kpl 16b osoitteita ja joku määrä dataa. Osoitteet kerätään rekisteriin, yhteensä siis 48b. Ensin lähetetään rekisterin 32b alinta bittiä. Seuraavaksi lähtetään bitit 32-47 siten että ne ovat siirrettävän osoitteen alimmat 16 bittiä ja ylimmät on täytetty nollilla. Sitten siirretään lähetettävä data.

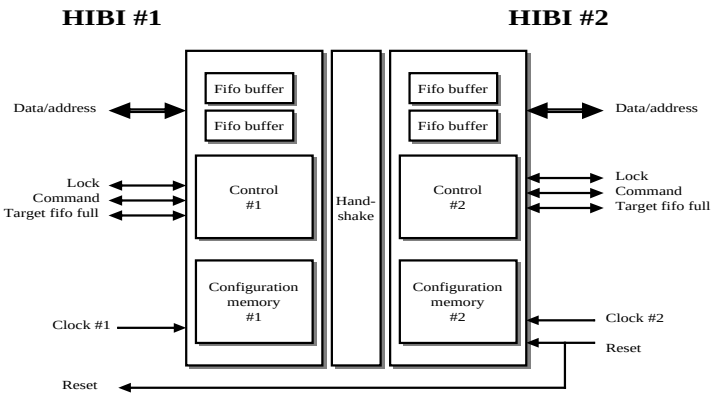
Pitkiä osoitteita vastaanotettaessa tarvitaan myöskin leveimmän mahdollisen osoitteen levyinen rekisteri. Lisäksi tarvitaan datalle siirtorekisteri, jonka pituus on sama kuin osoiterekisterin leveys jaettuna väylän leveydellä. Esim3ssa siirotekisterin pituudeksi tulisi siis 4. Siirtorekisterin leveys on sama kuin väylän leveys. Ensimmäisenä saapuva osoite talletetaan osoiterekisterin alimpiin bitteihin, seuraava osoite seuraaviin jne. Kun väylälle ilmestyy data, vertaillaan osoiterekisteriin kerättyä osoitetta liityntälohkon omaan osoitteeseen ja luetaan data väliaikaiseen datarekisteriin. Jos osoite täsmäsi, talletetaan osoitteen alimmat bitit fifoon. Seuraavalla kellojaksolla talletetaan osoitteen seuraava kaistale fifoon. Lisäksi talletetaan väylällä oleva data siirtorekisteriin ja siirretään siellä ollutta dataa yksi pykälä eteenpäin. Seuraavilla kellojaksoilla talletetaan saapuneet osoitteen palaset fifoon ja ruvetaan sen jälkeen tallettamaan dataa siirtorekisteristä. Osoitteiden talletus vie korkeintaan (osoiterekisterin leveys/väylän leveys) kellojaksoa, joten siirtorekisterissä ei voi tapahtua ylivuotoa.
Kannattaa huomata, että dataa ja osoitteita vastaanotettaessa ne talletetaan väylän levyiseen fifoon. Jos IP-lohko käyttää kapeampaa väylää, tehdään sovitus vasta kun data on kulkenut fifon läpi. Syynä on että väylältä täytyy pystyä vastaanottamaan joka kellojaksolla. Sovitus voidaan hoitaa IP-lohkon haluamalla tavalla. Esim tiettyihin osoitteisiin saapuva leveä data jaetaan useaksi kapeammaksi, kun taas toisiin osoitteisiin saapuvasta karsitaan vain tietyt bitit ja loput hylätään. Lähetysfifo sen sijaan voi olla tilan säästämiseksi saman levyinen kuin IP, koska osoitteita varteen on yllä selitetty kokoamismenkanismi. Jos halutaan suurempaa siirtotehokkuutta, kannattaa luonnollisesti käyttää väylän levyistä fifoa ja hoitaa sovitus ja kokoaminen IPn ja fifon välissä.

5. VÄYLÄN RAKENNE

Hibissä ei käytetä 3-tilalogiikkaa vaan väyläsignaalit on yhdistetty toisiinsa OR-porteilla. Kun liityntälohkolla ei ole mitään kirjoitettavaa, se kirjoittaa ulostuloihinsa nollia ja aktiivisen agentin ykköset jäävät voimaan.

6. SILTALOHKO

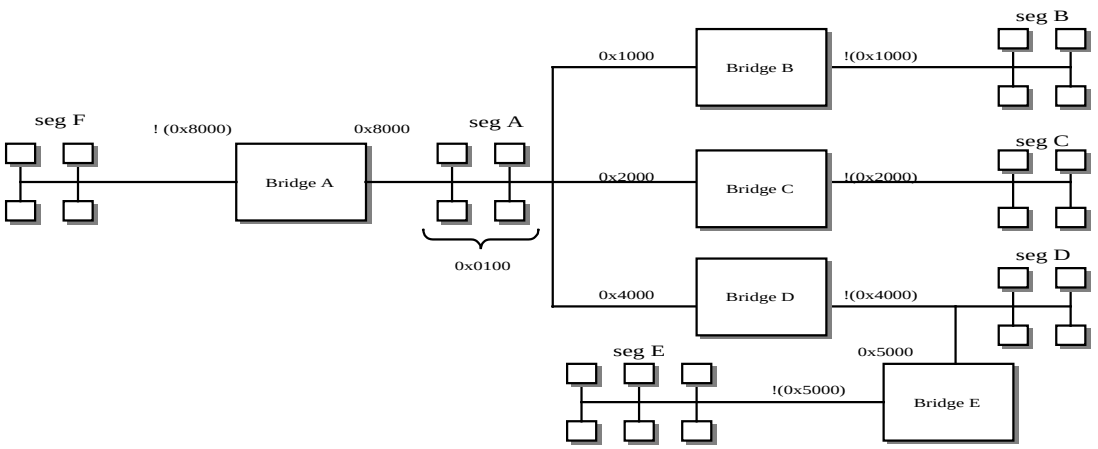
On mahdollista, etta järjestelmässä on useita hibiväyliä. Tällöin ne liitetään toisiinsa siltalohkon avulla. Kytkettävät väylät voivat toimia eri kelloilla. Siltalohkon rakenne on hyvin samantapainen kuin liityntälohkoilla. Siltalohkossa on kaksi hibiväylärajapintaa, muttei yhtään IP-rajapintaa, kuten Kuva 9. esittää.



Kuva 9.Siltalohkon rakenne

6.1.Monen väylän järjestelmät

Väylät voivat olla erilevyisiä ja toimia eri kelloilla. Sovitus tapahtuu siltalohkon sisällä. Kuva 10. esittää järjestelmää, jossa on monta väylää kytketty toisiinsa silloilla. Kuvassa on myös esitetty jokaisen väylän osoiteavaruus. Jotta jokaiselta väylältä olisi mahdollista kirjoittaa mille tahansa IP-lohkolle, täytyy sillan osoiteavaruus määritellä toiselta puolelta negaation avulla. Toisin sanoen, toiseen suuntaan välitetään osoitteet esim. välillä 10-20 ja toiseen suuntaan osoitteet jotka eivät ole välillä 10-20. Konfiguroinnin takia siltalohkoihin pitää samalla tavalla määritellä myös ID-avaruus eli mille puolelle tietyt agenttien IDt kuuluvat. Lisäksi siltalohkolla itsellään on kaksi IDtä, yksi molemmille hibirajapinnoille. Taulukko 16. ja Kuva 11. havainnollistavat muistin käyttöä.



Kuva 10. Monesta väylästä rakennettu systeemi.

Taulukko 16: Esimerkkiärjestelmän muistikartta

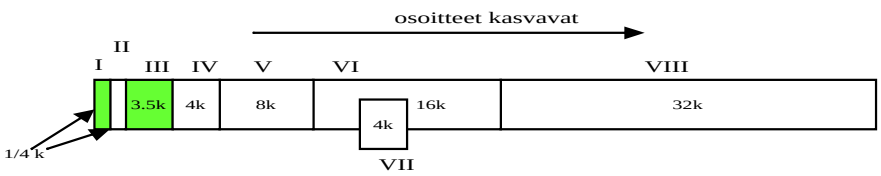
	Alku (hex)	Alku (b)	Loppu (hex)	Status	Vertailtavia bittejä	Osoiteavaruus
I	0x 00 00	0b 0...0	0x 00 FF	vapaa	0b !	8b = 256 = 1/4 k
II	0x 01 00	0b 0000 0001 0...0	0x 01 FF	varattu A	8b	8b = 256 = 1/4 k
III	0x 02 00	0b 0000 0010 0...0	0x 0F FF	vapaa	4b	12b-8b = 4k - 1/2k = 3.5 k
IV	0x 10 00	0b 0001 0...0	0x 1F FF	varattu B	12b	12b = 4k
V	0x 20 00	0b 0010 0...0	0x 3F FF	varattu C	3b	13b = 8k
VI	0x 40 00	0b 0100 0...0	0x 7F FF	varattu D+E	2b	14b = 16k
VII	0x 50 00	0b 0101 0...0	0x 5F FF	varattu E (sis. ed)	4b	12b = 4k
VIII	0x 80 00	0b 1000 0...0	0x FF FF	varattu F	1b	15b = 32k

Kaikki siirrot menevät myös siltojen ylitse, jos osoite/ID täsmäävät. Myöskin siis multicast- ja konfigurointikirjoitukset.Tarvittaessa useita väyliä sisältävä järjestelmä voidaan konfiguroida kokonaan yhdestä pisteestä käsin.
Kun siltalohko saa konfigurointikomennon, se välittää sen eteenpäin. Jos käsky oli broadcast, pitää se lukea sisään toiselta puolelta, kun se on sinne välitetty (ja tehdä toisellekin puolelle haluttu konfigurointi). Tällöin sitä ei kuitenkaan saa missään nimessä, enää välittää eteenpäin, koska se jää silloin oskilloimaan
Haluttu toiminta

- luetaan jonkun muun aloittama broadcastkonfigurointi väylältä #1
- tehdään oma konfigurointi väylän #1 konfigurointimuistiin
- välitetään käsky väylälle #2
- luetaan komento väylältä #2
- tehdään oma konfigurointi väylän #2 konfigurointimuistiin
- liityntälohko lukee väylältä (konfigurointi-)datan, jos se itse on kirjoituksen kohteena. Se ei kuitenkaan talleta (konfigurointi-)dataa fifoon (ja välitä sitä IP-lohkolle), jos se itse on sen väylälle kirjoittanut. Konfigurointimuistin päivitys suoritetaan normaalisti.

6.2.Konfiguraatiomuistit

Siltalohko sisältää 2 kpl konfiguraatiomuisteja (joiden rakenne on kuvattu kohdassa 4.5 Taulukkossa 12.). Muistit toimivat toisistaan riippumatta. Kumpikin ohjaa yhtä hibirajapintaa.



Kuva 11. Järjestelmän muistikartta.

6.3.Fifopuskurit

Siltalohkossa on neljä fifopuskuria kumpaankin suuntaan. Väylät voivat toimia eri kelloilla, joten fifopuskurin käsittelyssa on huolehdittava kättelystä.
Väylän 1 vastaaanottopuskuri on kytketty väylän 2 lähetyspuskuriin ja toisin päin.

7. MUUT OMINAISUUDET

7.1.Suorituskyky ja vasteajat

Aivan älytön suorituskyky. Riippuu käytetystä teknologiasta ja synteesistä.

7.2.Käytettävyys, toipuminen, turvallisuus, suojaukset

Väylä ei ota mitään kantaa virheentunnistukseen tai -käsittelyyn vaan niiden hoitaminen jätetään IP-lohkojen vastuulle.

7.3.Siirrettävyys/yhteensopivuus

Hibin määrittely ei ota mitään kantaa teknologiaan, jolla lopullinen piiri toteutetaan. Tavoitteena on myös, että liityntälohkojen koodaus tapahtuu niin korkealla tasolla, että teknologiaan tai layoutiin yms. ei oteta mitään kantaa. Jos vain mahdollista, toteutuksessa ei käytetä teknologiaan sidottuja muisteja.

8. HYLÄTYT RATKAISUVAIHTOEHDOT

Tässä kappaleessa esitetään mietityt, mutta hylätyt ratkaisuvaihtoehdot.

8.1.HYLÄTTY : Datan lähettäminen rinnakkain

Kapeata dataa (IPn eli datan leveys < hibin leveys) siirretään rinnakkain leveällä väylällä

- pitäisi tietää vastaanottajan leveys => tarvitaan osoitteen vertailu ja iso taulukko
- dataa pitäisi olla sopiva määrä, esim. 1x8b, 2x8b ja 3x8b olisi huonoja 32b väylälle

8.2. HYLÄTTY : Aikasloteissa määritellään joka kellojaksolle omistaja

- Uudelleen konfigurointi hidasta
- + taulukon käsittely olisi muuten kyllä helppoa
- kun määritellään alku- ja loppuaika ja omistaja, on uudelleenkonfigurointi helppoa. Sen sijaan jos on tarvetta hakea satunnaisesta kohtaa konfigurointimuistia, tarvitaan koko muistin levyinen rinnakkainen vertailu => varsin paljon rautaa

8.3. HYLÄTTY : Luovutettujen aikaslotien pyytäminen takaisin

- Hyöty jäänee vähäiseksi, koska agentti voi kuitenkin saada väylän kilpailemalla

8.4. HYLÄTTY : Kilpailutilanteessa edellisen/seuraavan aikaslotin haltijat saavat tavallista suuremman prioriteetin

- Ilmeisen hankala toteuttaa hajautetusti verrattuna saavutettavaan hyötyyn.

8.5.HYLÄTTY : Konfigurointimuistissa eri levyisiä muistipaikkoja

- Kaikki parametrit eivät tarvitse yhtä paljon talletustilaa. Olisi tuhlausta varata kaikille yhtä paljon tilaa. Esim. base ja top addr ovat väylän levyisiä (vaikkapa 32b), mutta connected vain 1b, num of agents 8b (=>agenttien max. määrä 256/väylä).
- Yksi toteutusvaihtoehto on jakaa konfigurointimuisti 8b tavuihin ja yksi arvo varaa sitten n tavua. Tällöin tulee varmistua, että kirjoitettaessa ei kirjoiteta viereisen parametriarvon päälle. Olisi kovasti hienoa, jos kirjoitus onnistuisi purskeena. Tyyliin ensimmäisen muutettavan parametrin osoite, ensimmäinen arvo, seuraava arvo, kolmas arvo jne.
-
- 02.01.2002
- Varataan kaikille sama tila ainakin alkuvaiheessa.

8.6. HYLÄTTY : Vastaanottofifot IPn levyisiä

- datan vastaaottaminen riittävän nopeasti hankalaa
- yhden siirron saa suhteellisen helposti hoidettua väliaikaisella puskuroinnilla (jolloin fifon kaventamisesta seuraava pinta-alan säästö tietenkin vähenee), mutta silloin seuraava siirto samalle agentille ei saisi alkaa ennen tiettyä aikarajaa (joka määräytyy käytetyistä väylien ja osoitteiden leveyksistä).

8.7. HYLÄTTY : IP-rajapinta on VCI-yhteensopiva

- Kaikki IPt eivät vielä moneen jouluun ole VCI yhteensopivia, varsinkaan itse kötöstetyt
- Silloin ei kannata tehdä väliin ylimääräistä rajanpintaa, jos IP voi suoraan käsitellä fifoja
- 4 rinnakkaista fifoa on tässä tapauksessa nopein mahdollinen rajapinta
- VCI-yhteensopivuus hoidetaan erillisellä wrapperilla, joka muuttaa VCI-rajapinnan hibin fiforajapinnaksi
- Samaten sovitus kaikkiin muihinkin rajapintoihin tehdään liityntälohkon ulkopuolella

9. JATKOKEHITYSAJATUKSIA

Matkan varrella mieleen tulleita (jatkorahastus ;-)) ajatuksia, joita ei tämän projektin puitteissa kuitenkaan määritellä tarkemmin tai toteuteta. Esimerkiksi ajan puutteen, rahan puutteen, resurssien puutteen tai taitojen ja osaamisen takia.

Nämä voisi luetteloida (esim. numeroida) jollakin tavalla, jotta niihin mahdollisesti viittaaminen olisi helpompaa myöhemmin.

Päiväys ja ehdottajan nimi(kirjaimet) (varsinkin jos lähde on ryhmän ulkopuolinen) auttavat jälkitarkastelussa, jos vuoden kuluttua projekti saakin yllättäen rahoitusta jatkokehitystä varten. Projektin lopussa nämä kerätään projektisuunnitelman loppuun.

10. VIELÄ AVOIMET ASIAT

10.1.Tehonsäästö (ei mikään kohta vielä)

Dynamic voltage scaling.

The Salaiset Asiat. Kysy Timolta.
Osa siirroista menee ehkä negaationa, jos sillä tavalla tulee vähemmän tilanvaihtoja signaaleihin. Tarvinnee ylimääräisen signaalin. Voitaneen toteuttaa erillisellä lohkolla joka kytketään liityntölohkon ja hibiväylän väliin.

11. VIITTEET

[1] K. Kuusilinna, T. Hämäläinen, P. Liimatainen, and J. Saarinen, " Low-latency interconnection for IP-block based multimedia chips", The Second IASTED International Conference on Parallel and Distributed Computing and Networks, Brisbane, Queensland, Australia,14-16 December, 1998, pp. 411-416.

[2] K. Kuusilinna, P. Liimatainen, T. Hämäläinen, and J. Saarinen, "Reconfiguration Mechanism for an IP Block Based Interconnection", Proceedings of the 25th EUROMICRO99 Conference, Milan, Italy, September 8-10,1999, pp.42-45.

[3] P. Liimatainen, "Low Latency Interconnection for IP Blocks", Master of Science Thesis, Tampere Univeristy of Techology, 1999.

12. LIITTEET

12.1.Liityntälohkon rajapinnat

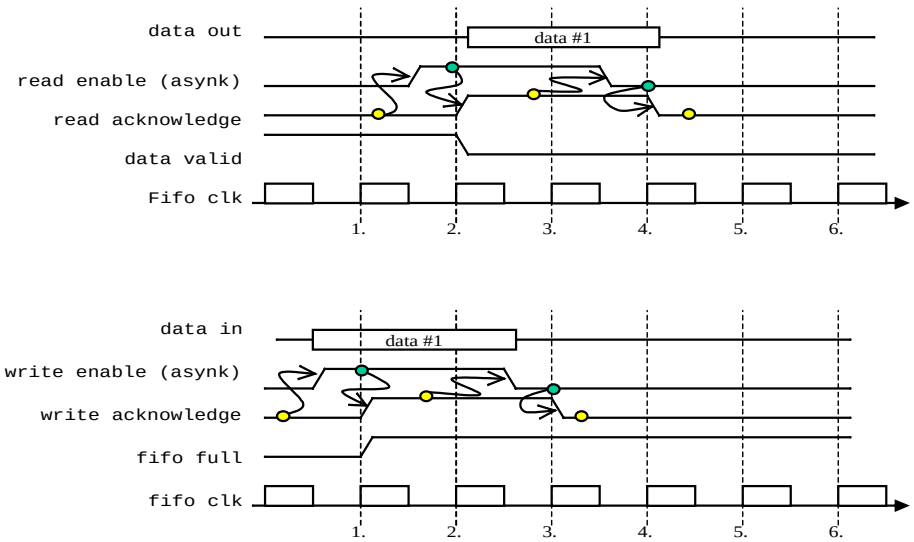
IP-rajapinta esitellään Taulukkossa 17. Kuten sanottua IP-rajapinta koostuu useasta fiforajapinnasta. Taulukossa on esitelty täydellinen rajapinta. On myös mahdollista käyttää suppeampaa rajapintaa, jolloin toinen tai molemmat viestiportit jäävät pois. Tällöinkin viestejä voi käyttää, mutta ne kulkevat datapuskurin kautta. Koska kaikkia portteja vastaa oma fifopuskuri, voidaan kaikkia portteja käsitellä yhtä aikaa toisistaan riippumatta.

Taulukko 17: Liityntälohkon ja IP-lohkon väliset signaalit

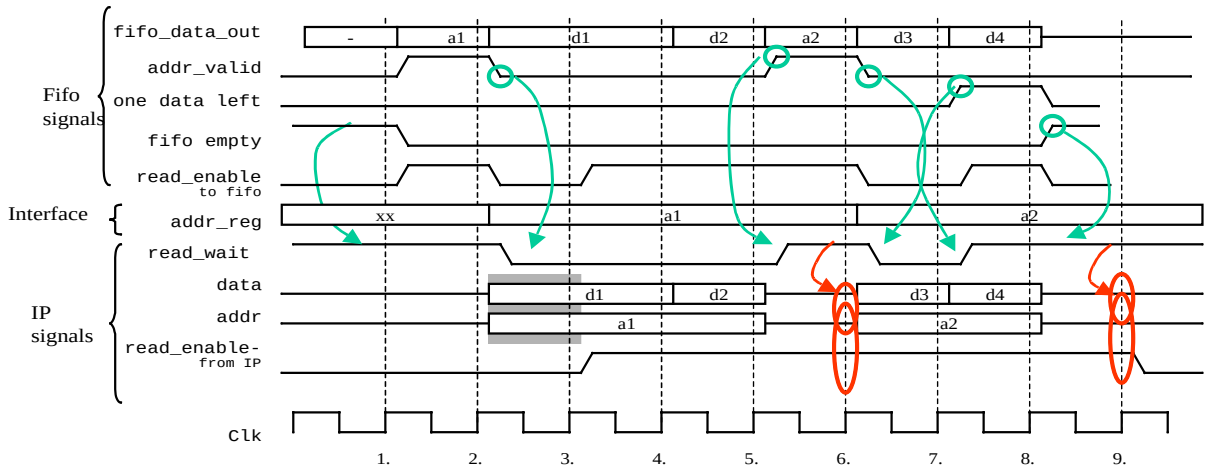
Signaalin nimi	Leveys	Suunta liityntä-lohkon kannalta		Signaalin nimi	Leveys	Suunta liityntä-lohkon kannalta
Data_In	8/16/32/64	I		Data_Out	8/16/32/64	O
Addr_Valid_In	1	I		Addr_Valid_Out	1	O
Comm_In	3	I		Comm_Out	3	O
Full	1	O		Empty	1	O
One_Place_Left	1	O		One_Data_Left	1	O
Write_Enable	1	I		Read_Enable	1	I
Msg_Data_In	8/16/32/64	I		Msg_Data_Out	8/16/32/64	O
Msg_Addr_Valid_ In	1	I		Msg_Addr_Data_Out	1	O
Msg_Comm_In	3	I		Msg_Comm_Out	3	O
Msg_Full	1	O		Msg_Empty	1	O
Msg_One_Place_Left	1	O		Msg_One_Data_Left	1	O
Msg_Write_Enable		I		Msg_Read_Enable	1	I

12.2.Fifojen ajoituskaaviot

Kuvassa 12. kuvataan fifopuskurien ajoitus asynkronista kättelyä käyttäen, kun halutaan käyttää IP-lohkoa, joka ei toimi samalla kellolla kuin hibiväylä (ja liityntälohkot). Ylemmässä kuvassa luetaan fifosta. Aluksi lukeva lohko asettaa *Read_Enable*n ykköseksi, koska fifossa on dataa se kirjoittaa sen ulostuloportteihinsa seuraavalla kellojaksolla ja asettaa *Read_Acknowledge*-signaalin ykköseksi. Samalla *Data Valid* laskee, mikä tarkoittaa että fifo on nyt tyhjä. Seuraavan luku-operaatio ei voi valmistua ennen kuin sinne saapuu dataa. Kellojaksojen 3 ja 4 välissä IP-lohko on saanut luettua datan ja se laskee *Read_Enable*-signaalin. Kellojaksolla 4. fifo huomaa tämän ja laskee *Read_Acknowledge*n.
Alemmassa kuvassa kirjoitetaan fifoon. Aluksi IP laitttaa kirjoitettavan datan datalinjoilla ja *Write_Enable*n aktiiviseksi. Kun fifo vastaa *Write_Acknowledge*llä, voidaan data poistaa ja asettaa *Write Enable* ei-aktiiviseksi.



Kuva 12. 4-vaiheinen kättely.

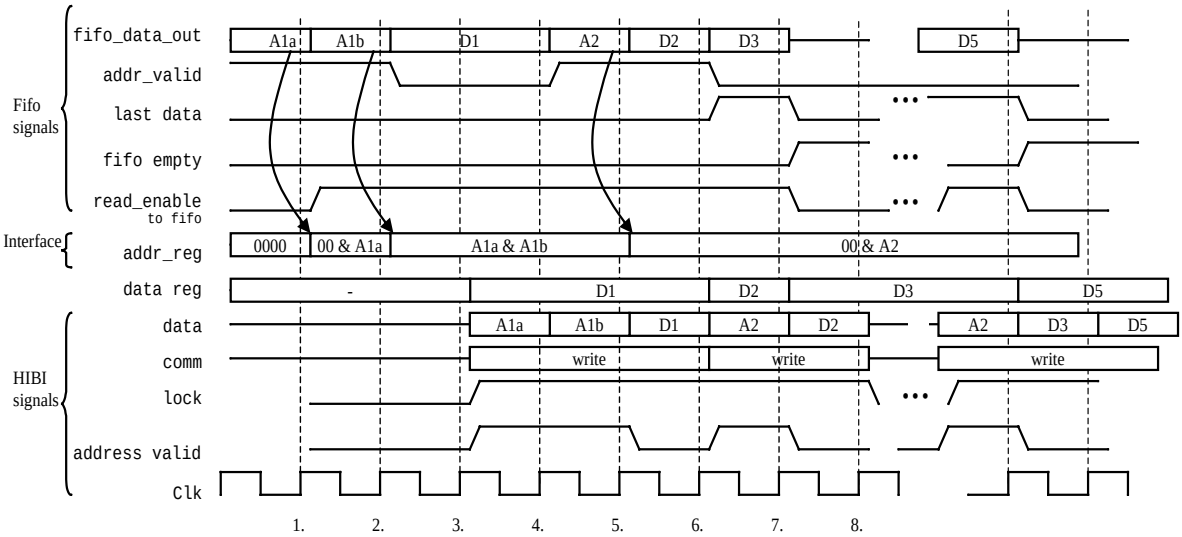


Kuva 13.Rx-fifossa ollut data siirretään IP-lohkolle.

Kuvassa tämä tapahtuu fifon kellojaksojen 2 ja 3 välissä, joten fifo laskee *Write_Acknowledge* -signaalin kellojaksolla 3. Seuraava kirjoitus voi alkaa kun *Write_Acknowledge* on laskenut.

Kuvassa 13. Rx-fifossa oleva data siirretään IP-lohkolle. Harmaat laatikot tarkoittavat kellojaksoja jolloin data olisi jo saatavilla, mutta IP ei asettanut *Read_Enable*a aktiiviseksi. Kellojaksoilla 6 ja 9 *Read_Wait* on aktiivinen, joten niillä kellojaksoilla data ei ole saatavilla vaikka *Read_Enable* on aktiivinen. Nuolet näyttävät minkä signaalien perusteella *Read_Wait* -signaalia ohjataan. Eli *Read_Wait* saadaan OR-operaatiolla signaaleista *Addr_Valid*, *One_Data_Left*, *Fifo_Empty.*

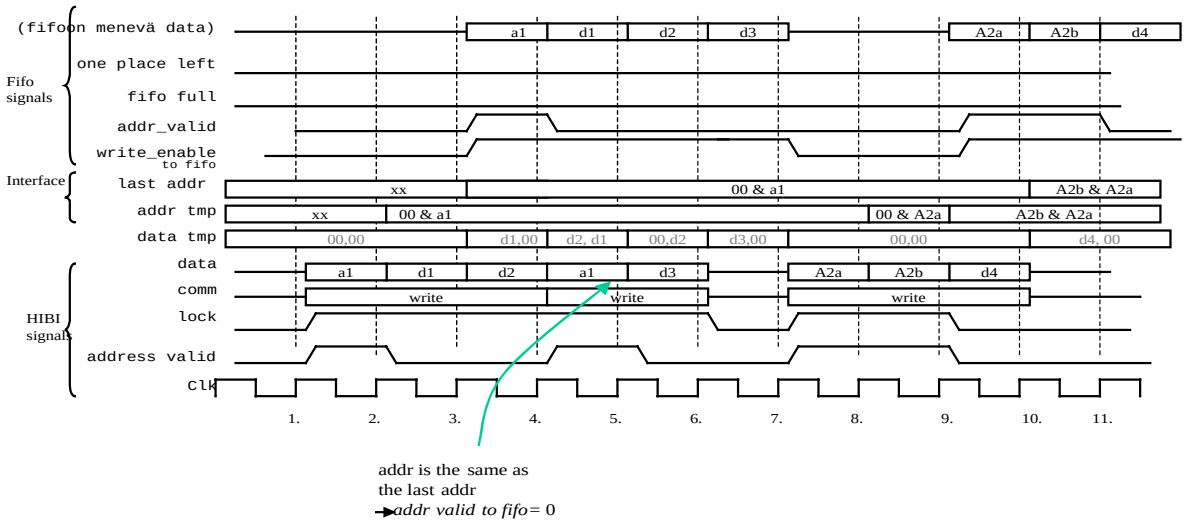
Kuvassa 14. esitetään fifon ulostulon ajoitus hibiväylälle kirjoitettaessa. Kellojaksolla 1 luetaan fifosta osoite joka ohjataan *Addr_Reg* -rekisterin alimpiin bitteihin. Kellojaksolla kaksi kirjoitetaan fifossa ollut seuraava osoite *Addr_Reg*iin. Seuraavalla kellojaksolla (3) fifossa on dataa, joka luetaan rekisteriin *Data_Reg*. Koska dataa on saatavilla, voidaan kellojaksolla 4 voidaan siirtää ensimmäinen osoite puolikas väylälle. Osoitteen siirtoa ei aloiteta



Kuva 14.Data kirjoitetaan transmitfifosta hibiväylälle.

ennen kuin on ainakin yksi data saatavilla. Samalla väylän *Addr_Valid* -signaali asetetaan aktiiviseksi. Osoitteen 1. puolikas luettavissa väylältä jaksolla 4 ja toinen puolikas jaksolla 5.
Jaksolla 5 fifossa on osoite, joka luetaan *Addr_Reg*in alimpiin bitteihin ja ylimmät nollataan. Jaksolla 6 ensimmäinen data D1 on väylällä ja fifossa on data D2. Nyt osoite on vain yhden sanan mittainen, joten sen siirtämiseen ei kulu kuin yksi kellojakso (7). Samalla kellojaksolla kirjoitetaan data D3 rekisteriin ja todetaan sen olevan viimeinen data fifossa. Kellojaksolla 8 siirretään data D2. Kellojaksolla 9 siirretään viimeinen data D3 ja *Lock* on laskenut. Koska *Lock*=0 jo kellojaksolla 8, voi joku muu agentti varata väylän jo seur jaksolla 9.
Kun fifoon saapuu joskus myöhemmin data D5, luetaan se saman tien rekisteriin. Kun agentti seuraavan kerran saa väylän haltuunsa, se siirtää aluksi *Addr_Reg*istä osoitteen (1 sana) ja sen jälkeen datarekisteristä arvon D5. Koska sillä ei ole muuta dataa, se voi jo osoitteen jälkeen laskea *Lock*in. Kun siirto lopetetaan aikaslottien (nykyinen aikaslotti loppuu tai seuraava alkaa) tai *MaxSend* -laskurin vuoksi, voidaan *Lock* laskea samaan tapaan jo samalla kellojaksolla kun viimeinen data siirretään (eli jaksoilla 9 ja n+1 *Lock* olisi jo 0). Tällä tavalla seuraava siirto voi alkaa välittömästi edellisen loputtua (jaksoilla 10 ja n+2). Kuvasta puuttuu rekisteri, jossa säilytetään tieto siitä, montako palasta osoitetta on luettu rekisteriin *Addr_Reg*.

Kuva 15. esittää tilannetta jolloin hibiväylältä luetaan Rx-fifoon dataa. Kellojaksolla 2 väylällä on osoite a1 (*Lock* =1, *Addr_Valid* = 1). Liityntälohko kirjoittaa osoitteen *Addr_Reg* -rekisterin alimpiin bitteihin. Tämän jälkeen tulevat osoitteet luetaan osoiterekisterin seuraaviin kohtiin. Esimerkissä tulee aluksi yhden sanan mittainen osoite. Jaksolla 3 väylällä on data D1, joka luetaan väliaikaiseen fifoon *Data_Reg*. Nyt tiedetään, että osoitetta ei ole tulossa lisää ja voidaan suorittaa vertailu *Base_Addri*n ja *Addr_Reg*in välillä. Koska osoite täsmää, jaksolla 4 voidaan osoite A1 kirjoittaa fifoon. Lisäksi osoite talletetaan rekisteriin *Last_Addr*, joka kertoo fifoon viimeksi talletetun osoitteen. Samalla luetaan väylältä data D2 väliaikaiseen fifoon ja siirretään siellä ennestään ollutta dataa yksi pykälä eteenpäin. Fifon pituus sama kuin osoiterekisterin leveys sanan moninkertoina. (Esim. 32b väylä, max osoite 128 b => osoite reksiteri 128b leveä ja väliaikaisen fifon koko (128/32) *32b). Liityntälohko ottaa kaikki datat vastaan kunnes väylälle imestyy seuraava osoite A1 jaksolla 5. Tämäkin osoite osuu liityntälohkon osoiteavaruuteen, joten se ottaa vastaan myös seuraavat datat (d3 ja d4). Tässä tapauksessa osoite A1 on kuitenkin sama kuin rekisterin *Last_Addr* arvo, joten sitä ei tarvitse tallettaa uudestaan fifoon. Datan d3 jälkeen siirto katkeaa ja sitä jatketaan jaksolla 8. Tällöin osoite a2 koostuu kahdesta osasta A2a ja A2b. Kun väylällä on data D4 jaksolla 10 suoritetaan osoitteen vertailu ja kirjoitetaan ensimmäinen puolikas fifoon. Samalla kirjoitetaan koko osoite rekisteriin *Last_Addr.* Kellojaksolla 11 kirjoitetaan osoite A2b fifoon ja seuraavalla jaksolla data D4.



Kuva 15.Hibiväylältä luetaan dataa receivefifoon.

12.3.Multiplekserit fifojen sisäänmenoissa

IP-puolella MUX-lohkon sisäänmenot ovat : DataIn, Addr_In, Write_Enable, Clk, Rst. Lohko sisältää yhden rekisterin Last_Addr.
Esim. IP-puolen MUX

12.4.Arbitroinnin ajoituskaavio

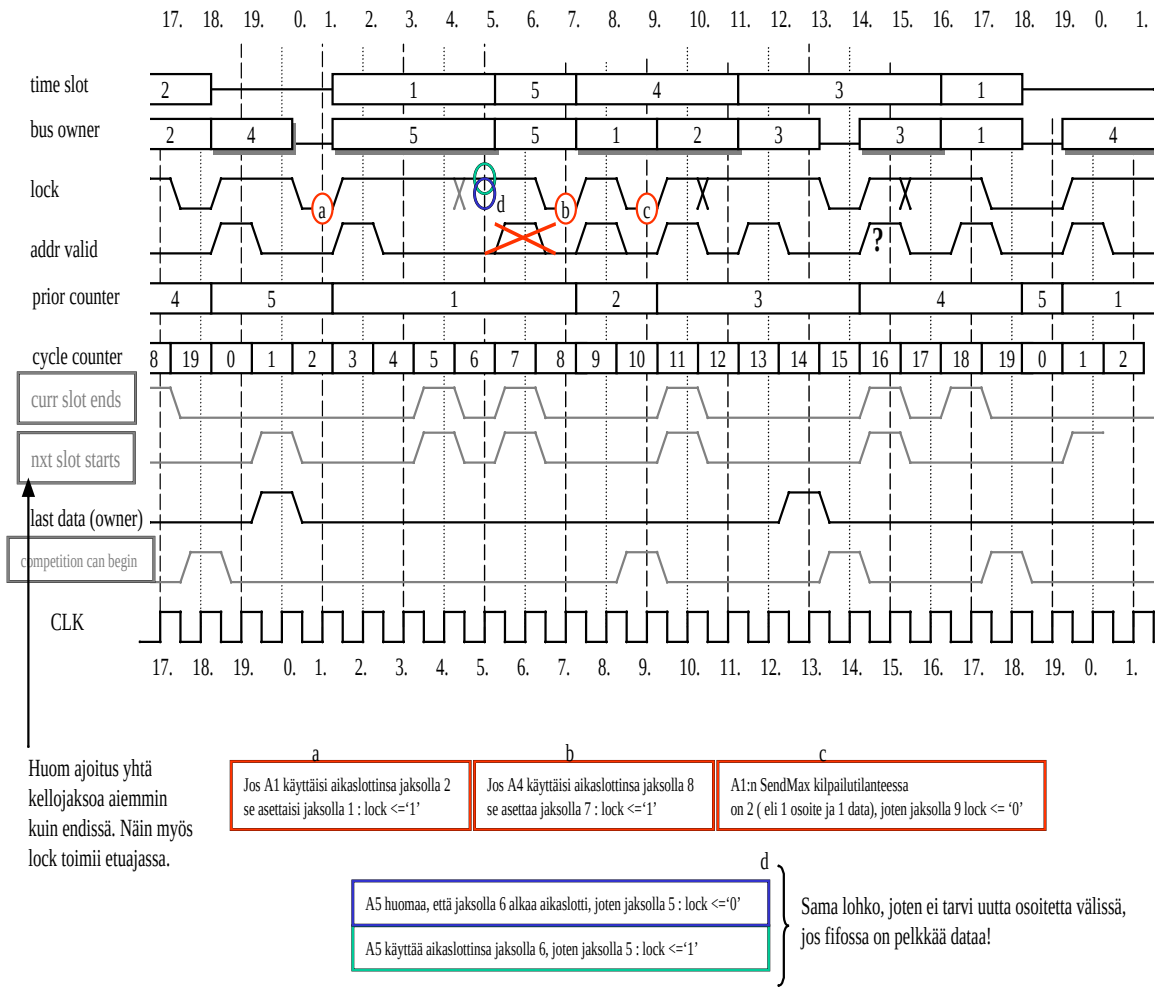
Kuva 17. esittää arbitroinnin ajoitusta hibiväylällä.

```
Reset:
Last_Addr <= (others => ‘0’);

Asynkroninen prosessi :
if Addr_In /= Last_Addr then
     New_transfer <=1;
     Data_out <= Addr;
else
     New_Transfer <= 0;
     Data_Out <= Data;
end if;

Synkroninen prosessi:
if New_transfer==1 then
     Last_Addr <= Addr_In;
else
     Last_Addr <= Last_Addr;
end if;
```

Kuva 16. MUX-lohkon pseudokoodi.



Kuva 17.Arbitroinnin ajoituskaavio.

Lock-signaali on aktiivinen (1)

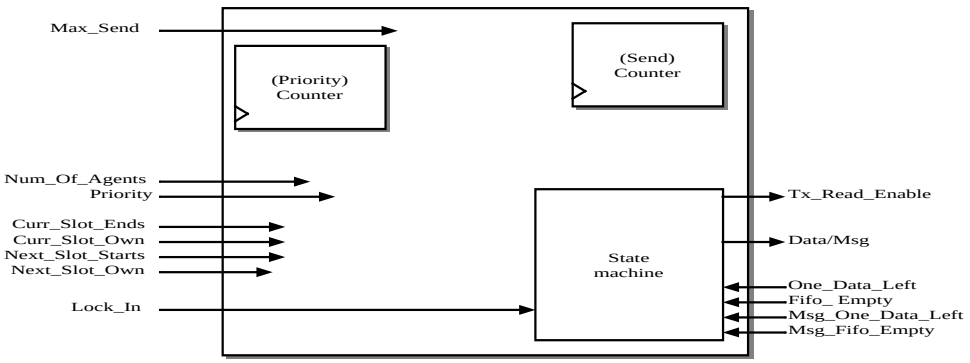
- jo yhtä kellojaksoa ennen aikaslotin alkua, jos sen omistaja haluaa slottinsa käyttää
- kilpavarauksen ensimmäisellä kellojaksolla

Lock on ei-aktiivinen (0)

- yhtä kellojaksoa ennen kuin aikaslotti loppuu
- yhtä kellojaksoa ennen kuin siirtoraja *MaxSend* tulee täyteen
- seuraavalla kellojaksolla kun fifosta on loppunut data

Tästä seuraa

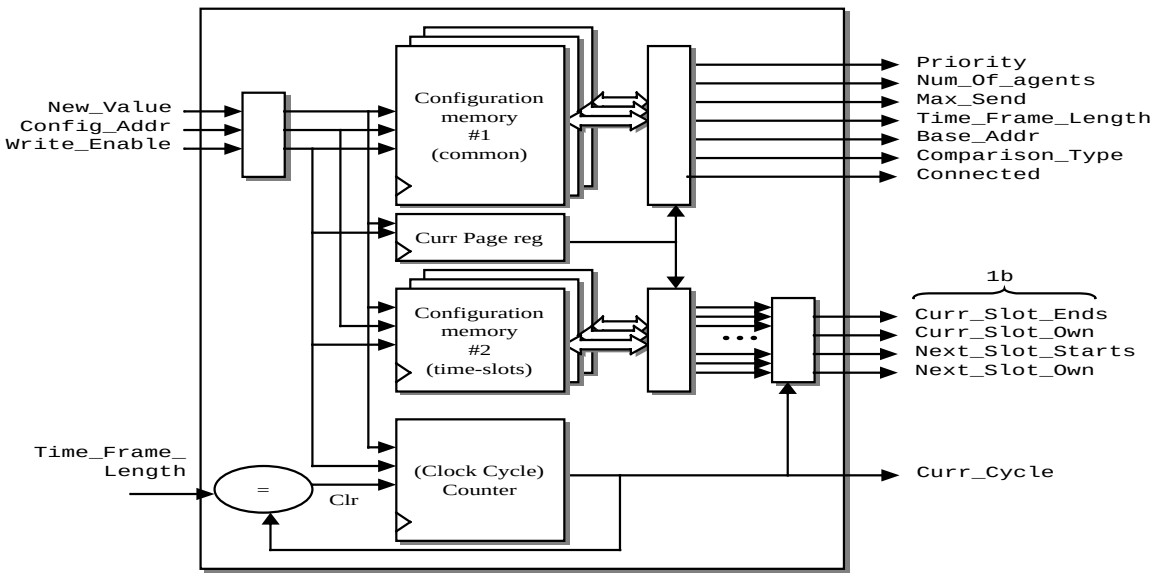
- aikaslotti voi alkaa heti edellisen loputtua
- kilpailuvuoro voi alkaa heti kokonaan käytetyn aikaslotin jälkeen
- kilpailuvuoro voi alkaa heti kokonaan käytetyn (send == *MaxSend*) kilpailuvuoron jälkeen
- kilpailuvuoro voi alkaa heti vastaanottajan täyttymiseen (lock=1 and target_full = 1) päättyneen siirron jälkeen
- jos data loppuu kesken, kestää yksi kellojakso ennen kuin seuraava siirto voi alkaa



Kuva 18.Arbitrointi. Lähetyksen tilakone ja arbitroinnissa tarvittavat laskurit.

12.5. Lähetyslohko

Tässä kappaleessa kuvataan lähetyslohkon rakenne. Lähetyslohkon tilakone on kuvattu kohdassa 4.4 (Kuva 6.). esittää ja esittää.
Tilakoneen ehdot taulukoissa Taulukko 18. ,Taulukko 19. ,



Kuva 19.Konfigurointimuisti.

Taulukko 18: Ehdot joilla tilasiirtymat tapahtuvat, osa 1

Ehdon numero	Curr state	Next state	Ehto
i1	Idle	Own slot	Own_starts (Prior = don't care)
i2	Idle	Re tx A	Own_Starts & Prior & Conf_Compl & ((Msg_AV & Msg_Empty) | (AV & Empty))
i3	Idle	Conf A	Own_Starts & Prior & Conf_Compl
i4	Idle	Write	Own_Starts & Prior & Conf_Compl & ((Msg_AV & Msg_1_Left) | (AV & 1_Left))
i5	Idle	Idle	Own_Starts & Prior | (Msg_Empty & Empty) & Conf_Complete & Retx_amount = 0)
o1	Own slot	Idle	
o2, o4	Own slot	Re tx A	Conf_Compl & ((Msg_AV & Msg_1_Left)| (AV & Empty))
o3	Own slot	Conf A	Conf_Compl
o5	Own slot	Write data	Conf_Compl & ((Msg_AV & Msg_1_Left) | (AV & 1_Left))
w1	Write data	Idle	Full & | (Curr=data&One_Left & (Msg_Empty | Msg_AV & Msg_1_left)) | (Curr=Msg & Msg_One_Left & (Empty | AV & One_Left))
w2	Write data	Write Last	Full & Conf_compl & Turn_Cont
w3	Write data	Write	Full & Conf_compl & Turn_Cont & (Curr=msg & Msg_One_Left) | (Curr=data & ((Msg_AV & Msg_1_Left) | (Msg_AV & Msg_Empty) | (AV & 1_Left) | (AV & Empty))
w4	Write data	Conf A	Full & Conf_compl & Turn_Cont

Taulukko 19: Ehdot joilla tilasiirtymat tapahtuvat, osa 2

Ehdon numero	Curr state	Next state	Ehto
ca1	conf A	conf last	Full & Turn_Cont
ca2	conf A	conf D	Full & Turn_Cont
??	conf A	Idle	Full MAHDOTON???
cd1	conf D	Idle	Full | Turn_Cont | (((Msg_AV & Msg_1_Left) | Msg_Empty) & (Empty | AV & 1_Left))
cd2	Conf D	Re tx A	Full & Turn_Cont & (Msg_AV & Msg_Empty) | (Msg_Empty & AV & Empty))
cd3 ??	conf D	Write	Full & Turn_Cont & (Msg_AV | (Msg_Empty & AV & 1_Left))
ra1	Re tx A	Write Last	Full & Turn_Count & Retx_amount = 0
ra2	Re tx A	Write	Full & Turn_Count & Retx_amount = 0
ra3	Re tx A	Re tx Last	Full & Turn_Cont & Retx_amount != 0
ra4	Re tx A	Re tx D	Full & Turn_Cont & Retx_amount != 0
??	Re tx A	Idle	Full MAHDOTON ????
rd1	Re tx D	Idle	Full
rd2	Re tx D	Retx Last	Full & Turn_Cont
rd3	Re tx D	Conf A	Full & Turn_Count & Retx_amount = 0 & Conf_Compl
rd4	Re tx D	Re tx D	Full & Turn_Cont & Retx_amount != 0
rd5	Re tx D	Write	Full & Turn_Count & Retx_amount = 0 & Conf_Compl

Taulukko 20: Tilasiirtymaehdot, osa 3

Nykyinen tila	Ehto	Toiminto	Seur. tila
Idle	Väylä on varattu. Tai oma aikaslotti alkaa tai prior täsmää, mutta ei ole dataa eikä viestejä, on vastattu konf.lukuun ja edellinen siirto onnistui. Tai oma aikaslotti ei ala, eikä prioriteetti täsmää. Tai jonkun muun aikaslotti alkaa.	Rekisterit säilyttävät edelliset arvonsa,	Idle
	Oma aikaslotti alkaa. Ja on joko dataa tai viestejä tai edellinen siirto epäonnistui tai ei ole vastattu konf.lukuun.		Own Slot
	Ei ala oma eikä kenekään muunkaan aikaslotti, mutta prioriteetti täsmää. On joko dataa tai viestejä ja on vastattu konf.lukuun ja edellinen siirto onnistui.		Write
	Ei ala oma eikä kenekään muunkaan aikaslotti, mutta prioriteetti täsmää ja edellinen siirto onnistui. Ei ole vastattu konf.lukkuun. Fifoilla ei merkitystä.		Conf A
	Ei ala oma eikä kenekään muunkaan aikaslotti, mutta prioriteetti täsmää. On joko dataa tai viestejä ja on vastattu konf.lukuun ja edellinen siirto onnistui. Ei ole kuitenkaan uutta osoite viestiin/dataan.		Re tx A
Own slot	Ed. siirto onnistui ja konf.lukuun on vastattu. (Fifoissa pitäisi olla edelleen dataa, koska oli tähän tilaan tullessakin.)		Write
	Ed. siirto onnistui, mutta konf.lukuun ei ole vastattu. Fifojen datan määrällä ei vaikutusta.		Conf A
	Edellinen siirto epäonnistui tai on viesti ilman uutta osoitetta tai konf.lukuun on vastattu ja on viestejä tai dataa ilman uutta osoitetta.		Re tx A
Write	Kohde täysi. Tai data ja viestit loppuvat ja on vastattu konf. lukuun.		Idle
	Kohde ei ole täysi. Konf. lukuun on vastattu. Oma vuoro jatkuu. Ja on viestejä tai dataa jäljellä		Write

13. Testaus

Tässä kappaleessa kuvaillaan testipenkkien toiminta

13.1.Fifo

Taulukko 21: Fifon testaaminen

Kuvaus	Tulos	Testataanko, jos pituus =1
Testataan fifon pituuksilla 1, 2 ja 2+		
Kirjoitetaan tyhjään fifoon	Fifossa 1 data	K
Kirjoitetaan fifoon joka ei ole tyhjä eikä täysi. Fifossa on yli_2/tasan_2 tyhjänä.	Fifossa 1 data enemmän eikä se ole tyhjä eikä täysi. Fifossa on ainakin_2/ tasan_1 tyhjää paikaa.	E
Kirjoitetaan täyteen fifoon	Fifon tila ei muutu.	K
Kirjoitetaan yhtä vaille täyteen fifoon	Fifo täysi.	sama kuin tyhjään
Lutetaan tyhjästä fifosta	Fifon tila ei muutu.	K
Lutetaan yhtä vaille tyhjästä fifosta	Fifo tyhjä.	sama kuin taysi
Luetaan fifosta joka ei ole tyhjä eikä täysi. Fifossa on yli_2/tasan_2 dataa.	Fifossa 1 data vähemmän eikä se ole tyhjä eikä täysi. Fifossa on ainakin_2/ tasan_1 dataa.	E
Luetaan täydestä fifosta	Fifossa on yksi tyhjä paikka.	K
Luetaan ja kirjoitetaan kun fifo on tyhjä	Fifossa on yksi data. Luettaessa saadaan määrittelemätön vanha arvo.	K
Luetaan ja kirjoitetaan kun fifo on täysi	Fifossa on ykis tyhjä paikka. Kirjoitettu data jätetään huomiotta.	K
Luetaan ja kirjoitetaan kun fifo ei ole tyhjä eikä täysi	Fifossa on sama määrä dataa kuin operaatiota. Sekä luku että kirjoitus onnistuu. Sisäiset laskurit päivittyvät.	E

13.2. Osoitteen vertailu

Taulukko 22: Osoitteen vertailun testitapaukset

Testin kuvaus : komento	Testin kuvaus	Tulos
Idle	Oikea/väärä osoite	
Write Conf	Oikea/väärä osoite	

Taulukko 22: Osoitteen vertailun testitapaukset

Testin kuvaus : komento	Testin kuvaus	Tulos
Write Data	Oikea/väärä osoite	
Write Msg	Oikea/väärä osoite	
Read data	Oikea/väärä osoite	
Read conf	Oikea/väärä osoite	
Multicast data	Oikea/väärä osoite. Alimmat biti “00”, “01” ,”10” ja “11”	
Multicast msg	Oikea/väärä osoite Alimmat biti “00”, “01” ,”10” ja “11”	
ID /Base addr vaihtuu		

13.3.Konfiguraatiomuisti

Taulukko 23: Konfiguraatiomuistin testaaminen

Testin kuvaus	Tulos	
Kirjoitetaan aktiiviselle sivulle yksi parametri	Parametriarvo muuttuu	
Kirjoitetaan ei-aktiiviselle sivulle yksi parametri	Parametriarvo muuttuu	
Luetaan aktiiviselta sivulta yksi parametri		
Luetaan ei-aktiiviselta sivulta yksi parametri		
Vaihdetaan aktiivista sivua		
Kirjoitetaan kaikki parametrit aktiiviselle sivulle		
Kirjoitetaan kaikki parametrit ei-aktiiviselle sivulle		
Synkronoidaan kirjoittamalla kellojaksolaskuriin	Kellojakso muuttuu	
Kirjoitetaan olemattomalle sivulle		
Kirjoitetaan olemattomaan parametripaikkaan		
Luetaan olemattomalta sivulta	Tila ei muutu.Luettu arvo pelkkiä nollia.	
Luetaan olematon parametri	Tila ei muutu. Luettu arvo pelkkiä nollia.	

13.4.Vastaanotto

Taulukko 24: Vastaanoton testaaminen

Testin kuvaus	Kuvaus jatkuu	Tulos
Väylällä komento write, yhden pituinen osoite ja	yksi data	
	kaksi dataa	
	kaksi data. Toistetaan kaksi kertaa peräkkäin	
	kaksi dataa. Toistetaan kolme kertaa peräkkäin	
	dataa kunnes lähetysfifo tyhjä	
	dataa kunnes lähetysraja tulee vastaan	
	dataa kunnes aikaslotti loppuu	
	dataa kunnes seuraava aikaslotti alkaa	
	5D, sitten 1A + 4D joka kuulluu eri kohteelle. Sen jälkeen 1A + 3D testattavalle kohteelle	
	dataa kunnes aikaslotti vaihtuu ja seuraava alkaa saman tien.Toinen siirto 1 A + 3 D	
	dataa kunnes lähetysraja tulee vastaan. Seuraava kilpailusiirto alkaa heti kun mahdollista	
	dataa kunnes vastaanotto fifo täysi	
	50 dataa, s.e. vastaanottava IP lukee fifoa koko ajan	
	aikaslotti loppuu ja seur. alkaa heti perään. Siinä 1A + 5D	
	aikaslotti loppuu ja kilpailuvuoro alkaa heti perään. Siinä 1A + 5D	
	kilpailuvuoro loppuu, eur. alkaa heti perään. Siinä 1A + 5D	
	kilpailuvuoro loppuu, seur. aikaslotti alkaa heti perään. Siinä 1A + 5D	
Samat toistetaan kaikille mahdollisille osoitteen pituuksille sekä laittomalle pituudelle (max. pituus +1). Lisäksi sama kokeillaan mutlicastina.		
Siirretään suoraan dataa ilman osoitetta		
kom = write, osoite 1 A	dataa siten että komento vaihtuu.	Osoite ja data hylätään
kom = write, osoite 1 A	toinen osoite, komento vaihtuu konfiguraatiokomennoksi. Konfiguraatio-osoitteeksi tulkittuna osoite täsmää.	Osoite ja data hylätään

Taulukko 24: Vastaanoton testaaminen

Testin kuvaus	Kuvaus jatkuu	Tulos
kom = konf, 1 osoite ja	1 data	
	2 data	Jälkimmäinen data hylätään??? Miten muuten hoidettaisiin osoitteen lähetys, jos tämän tyyppinen konfigurointi katkeaa.
Toistetaan muilla osoiteen pituuksilla.		

13.5.Lähetys

Taulukko 25: Lähetyksen testaaminen

Testin kuvaus	Kuvaus jatkuu	Kuvaus jatkuu	Kuvaus jatkuu	Tulos
Agentti saa aikaslotin / kilpailuvuoron	Ei dataa, eikä viestejä, ei kesken jääneitä	Ei konf.lukua kesken		
		konf.luku kesken	Vastaanottajalla tilaa/ ei ole tilaa. Vuoro jatkuu / ei jatku.	
	Ei D, ei Msg, ei konf lukua kesken	Keskeytynyt siirto (1) / (2)	Vastaanottajalla tilaa/ ei ole tilaa. Vuoro jatkuu / ei jatku.	
	Ei D	kesk (1)/(2)., konf .luku kesken, ei Msg	Vastaanottajalla tilaa/ ei ole tilaa. Vuoro jatkuu / ei jatku.	
		kesk (1)/(2), konf.luku msg	Vastaanottajalla tilaa/ ei ole tilaa. Vuoro jatkuu / ei jatku.	
		kesk (1)/(2)., ei konf .lukua , msg	Vastaanottajalla tilaa/ ei ole tilaa. Vuoro jatkuu / ei jatku.	
		ei kesk., konf .luku kesken, msg		

Taulukko 25: Lähetyksen testaaminen

Testin kuvaus	Kuvaus jatkuu	Kuvaus jatkuu	Kuvaus jatkuu	Tulos

13.6.Liityntälohko