NORGE

(12) **PATENT**

(19) NO (11) **312926** (13) B1

(51) Int Cl[7] G 06 F 13/362

Patentstyret

(21) Søknadsnr	19993558	(86) Int. inng. dag og søknadsnummer	
(22) Inng. dag	1999.07.20		
(24) Løpedag	1999.07.20	(85) Videreføringsdag	
(41) Alm. tilgj.	2001.01.22	(30) Prioritet	Ingen
(45) Meddelt dato	2002.07.15		

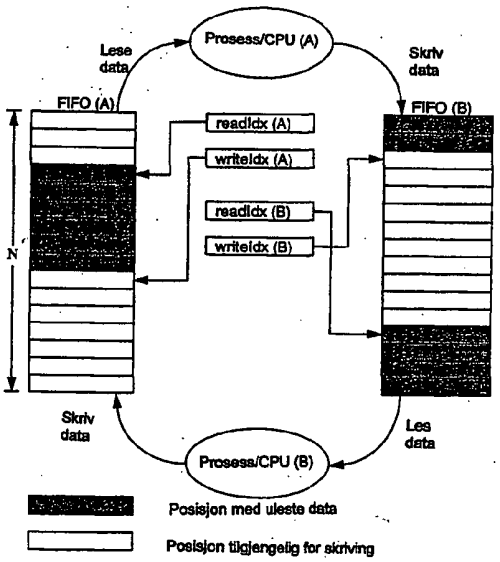
(71) Patenthaver Telefonaktiebolaget L M Ericsson, S-126 25 Stockholm, SE
(72) Oppfinner Geir Robert Svelmoe, 0665 Oslo, NO
(74) Fullmektig Oslo Patentkontor AS, 0306 Oslo

(54) Benevnelse **Kommunikasjon over multimaster buss**

(56) Anførte publikasjoner US 5325492, US 4642755, US 6032267, EP A1 7848, US 5617537, US 5522045, EP A2 360153, WO A1 9700533

(57) Sammendrag

Den foreliggende oppfinnelse vedrører kommunikasjon av data mellom to parter i et multimaster bussystem, slik som PCI-buss eller VME-buss, ved bruk av delt RAM. Partene kan omfatte to CPUer, to SW-prosesser som utføres på samme prosessor, eller en SW-prosess og en avbruddsrutine. I nevnte RAM er det innrettet et FIFO-minne lokalt til hver part. Hver part kan bare lese fra sitt lokale FIFO-minne og skrive til en annen parts FIFO-minne. På denne måte blir bare skriveoperasjoner utført på bussen.



312926

Teknisk område

Denne oppfinnelse kan brukes i et multimaster bussystem (for eksempel PCI, VME og så videre) hvor punkt-til-punkt kommunikasjon mellom to mastere skal utføres ved å bruke delt RAM til overføring av data. Et spesialtilfelle vil være datautveksling mellom to SW-prosesser som utføres på samme prosessor, eller mellom en SW-prosess og en avbruddsrutine.

Teknisk bakgrunn

Problemområde

Når delt minne brukes til å utveksle data mellom to kommuniserende parter som arbeider asynkront i forhold til hverandre, må man sikre at de to ikke oppdaterer samme minneposisjon(er) samtidig.

I noen systemer vil en leseoperasjonen på bussen være langsommere enn en skriveoperasjonen på bussen (adressedekodingen for målet vil være den samme, men for at en leseoperasjon skal fullføres må data hentes fra bussgrensesnittet, mens for en skriveoperasjon kan dataene temporært bufres i grensesnittkretsen for bussen). Derfor ville et "bare skrivende"-system være mer effektivt.

Kjente løsninger og problemer med disse

For å få tilgang til en delt minneressurs har noen prosessorer og bussystemer en atomisk "test-og-sett"-instruksjon som blir en enkelt lese-modifiser-skriv operasjon på bussen. Disse instruksjoner blir ikke understøttet av alle prosessorer/bussystemer, og hvis de blir understøttet må man ha en plattformspesifikk del av SW (vanligvis vil denne delvis være manuelt kodet i assemblyspråk). Hvis man kan unngå felles minneposisjoner som skal oppdateres av mer enn én part, vil man oppnå en mer portabel høynivå SW.

For prosesser som utføres på samme CPU vil det typisk være nødvendig å maskere avbrudd under minneoppdateringen. Vanligvis blir prosesser på brukernivå ikke tillatt å gjøre dette, og de må utstede et systemkall for å utføre oppdateringen. Ved å eliminere behovet for å få minneposisjoner oppdatert av mer enn én part, vil det ikke være noen forskjell i SW hvis de to prosesser utføres på samme prosessor eller på to forskjellige prosessorer som har bussaksess til den annens minne.

Oppfinnelsen

Kortfattet sammenfatning av oppfinnelsen

Hensiktene bak foreliggende oppfinnelse er derfor å tilveiebringe et arrangement for punkt-til-punkt kommunikasjon mellom to mastere eller prosesser over et multimaster bussystem, som tillater kommunikasjon og utføres asynkront og samtidig, som er uavhengig av maskinvare, tillater raskere overføring av data mellom masterne eller prosessene og letter belastningen på buss og CPU(er).

Ytterligere hensikter ved oppfinnelsen er å unngå bruk av felles minneposisjoner oppdatert av mer enn én part, og unngå leseoperasjoner på bussen.

Disse hensikter oppfylles i et arrangement ifølge oppfinnelsen omfattende FIFO-minner for overføring av data mellom de to CPUer/kommunikasjonsparter, hvor en FIFO er tildelt hver CPU/part, hvor hver part tillates å lese bare fra sin tildelte FIFO, og bare skrive FIFOen tildelt den andre part.

Den nøyaktige beskyttelsesomfang for den foreliggende oppfinnelse er definert i de vedlagte patentkrav.

Tegninger

Oppfinnelsen vil nå bli beskrevet med henvisning til de vedlagte tegninger hvor:

Figur 1 viser to FIFO-minner arrangert for overføring av data mellom CPUer/parter, A og B.

Figur 2 viser i prinsippet hvordan en FIFO er innrettet til å motta data fra bare én CPU/part B, og hvordan lese/ skriveindeksene blir oppdatert.

Figur 3 viser det fulle arrangement av FIFOer ifølge oppfinnelsen.

Figur 4 viser et forenklet overblikksbilde av et system hvor oppfinnelsen kan benyttes.

Detaljert beskrivelse av oppfinnelsen

Oppfinnelsen forslår en HW uavhengig løsning for en kommunikasjonsmekanisme over delt minne. De to kommuniserende parter kan være plassert på samme prosessor eller i forskjellige prosessorer i et multimaster bussystem.

Kommunikasjonen mellom de to CPUer/kommuniserende parter utføres gjennom et par med SW FIFOer, én for hver retning av dataflyten. Det spesielle med disse FIFOer er den fysiske plassering av deres komponenter.

For å beskrive denne FIFO-organisering vil vi starte med å tegne to regulære SW FIFOer i et felles fysisk minne, figur 1, og gradvis migrere mot SW FIFO-organiseringen foreslått ved oppfinnelsen.

Leseindeksen og skriveindeksen peker henholdsvis på den neste posisjon som skal leses og skrives. Mengden data i FIFOene må beregnes fra verdiene av lese- og skriveindekse-

ne sammen med FIFO-størrelsen, som i dette tilfellet er N. De kommuniserende prosesser/prosessorer arbeider asynkront og samtidig. Bruk av en teller til å holde antallet dataposter i en FIFO ville medføre behov for en innbyrdes ekskluderingsmekanisme for å oppdatere denne teller.

Når leseindeksen er lik skriveindeksen er FIFOen tom. FIFOen kan høyst inneholde N-1 poster. Hvis FIFOen skulle inneholde N poster, ville leseindeksen være like skriveindeksen og det ville være umulig å sjeldne mellom et fullt og et tomt datafelt.

Prosess B, som sender data til prosess A ved å sette dem inn i FIFO(A), er den eneste prosess som kan skrive til datafeltet i FIFO(A) og til writeIdx(A). Før noen data blir skrevet, må prosess B beregne mengden tilgjengelig rom i FIFO(A). Prosess A leser writeIdx(A), beregner antallet data i FIFOen, leser dataene og oppdaterer readIdx(A) (som kan oppdateres bare ved prosess A). For data i den annen retning er rollene motsatt. Siden de to prosesser er asynkrone i forhold til hverandre, er det viktig at den respektive indeks blir oppdatert etter at lese- eller skriveoperasjonen er fullført.

Hvis FIFO-komponentene er plassert på en slik måte at bare skriveaksesser skal utføres på bussen som knytter sammen de to prosessorer, ville man bare ha nettomengden av bussaksesser til å overføre dataene (det vil si man ville aldri aksessere bussen for å lese en indeks som ikke er oppdatert). Ved å organisere FIFOene på denne måten ville man skrive informasjon til en annen CPU til bussen og lese all den innkomne informasjon i det lokale minnet (som også ville være raskere).

Figur 2 viser den foreslåtte plassering av komponentene for FIFOen benyttet for dataene fra en CPU(B) til en annen CPU(A). Når B ønsker å sende data til A sjekker den i sitt lokale minne for verdien av readIdx(A). Siden A ikke tilla-

tes å oppdatere writeIdx(A) i sitt lokale minne, kan B beregne den tilgjengelige plass i FIFO(A) basert på readIdx(A) og den huskede verdi av writeIdx(B) (som B har skrevet inn i As minne tidligere). B skriver da dataene etterfulgt av en ny writeIdx(A) i As minne.

Det fullstendige bilde av plasseringen av de to FIFO-komponenter er vist i figur 3.

All informasjon (data og indekser) som skal leses av det fjerntliggende system, må skrives inn i dette systems minne, men selvfølgelig må også en lokal kopi av indeksene opprettholdes.

Siden et FIFO-par bare er en punkt-til-punkt kommunikasjonsmekanisme, må det være et par for hvert par av kommuniserende parter.

Figur 4 viser et forenklet overblikk over et system hvor oppfinnelsen kan benyttes.

Fordeler

Portabilitet: Det kan implementeres i høynivå språk uten noen plattformspesifikk assemblyinstruksjon eller OS-systemkall.

De to kommuniserende parter kan utføres på samme prosessor eller på to forskjellige prosessorer med den samme høynivå kode for å ta hånd om kommunikasjonen.

Bussbelastning: Bussen mellom de to CPUer vil være opptatt for et kortere tidsrom siden det bare vil være skrivetransaksjoner og bare nettomengden av de nødvendige data som vil legges på bussen. Jo mindre dataskuren er dess mer effektivt vil "bare skriving" være, sammenlignet med et lese/skrivesystem.

CPU belastning: Siden data kan bufres på alle bussgrense-snitt nedover transaksjonsbanen, vil operasjonen fullføres på mindre tid og derfor gi mer tid for annen prosessering.

En viktig fordel ved oppfinnelsen er at den kan implementeres i programvare alene, det vil si uten å gjøre endringer i maskinvaren i de involverte systemer.

P a t e n t k r a v

1. Arrangement for å tilveiebringe punkt-til-punkt kommunikasjon mellom en første part og en andre part over en intern multimaster buss ved bruk av delt RAM for å overføre data, hvor dataene er i form av en datastrøm og hvor arrangementet omfatter et første og et andre FIFO-minne arrangert i nevnte RAM,
k a r a k t e r i s e r t v e d at det første FIFO-minnet er arrangert lokalt til nevnte første part og det andre FIFO-minnet er arrangert lokalt til nevnte andre part, idet den første part bare tillates å lese fra et datafelt i det første FIFO-minnet og bare skrive til et datafelt i det andre FIFO-minnet.

2. Arrangement ifølge krav 1,
k a r a k t e r i s e r t v e d at den andre part bare tillates å lese fra et datafelt i det andre FIFO-minnet og bare skrive til et datafelt i det første FIFO-minnet.

3. Arrangement ifølge krav 1 eller 2,
k a r a k t e r i s e r t v e d at hvert FIFO-minne inkluderer en skriveindeks for sitt eget datafelt og en leseindeks for datafeltet i det andre FIFO-minnet, hvor indeksene i det første FIFO-minnet bare oppdateres av den andre part og indeksene i det andre FIFO-minnet bare oppdateres av den første part.

4. Arrangement ifølge krav 3,
k a r a k t e r i s e r t v e d at hvert FIFO-minne inkluderer en lokal kopi av indeksene i det andre FIFO-minnet.

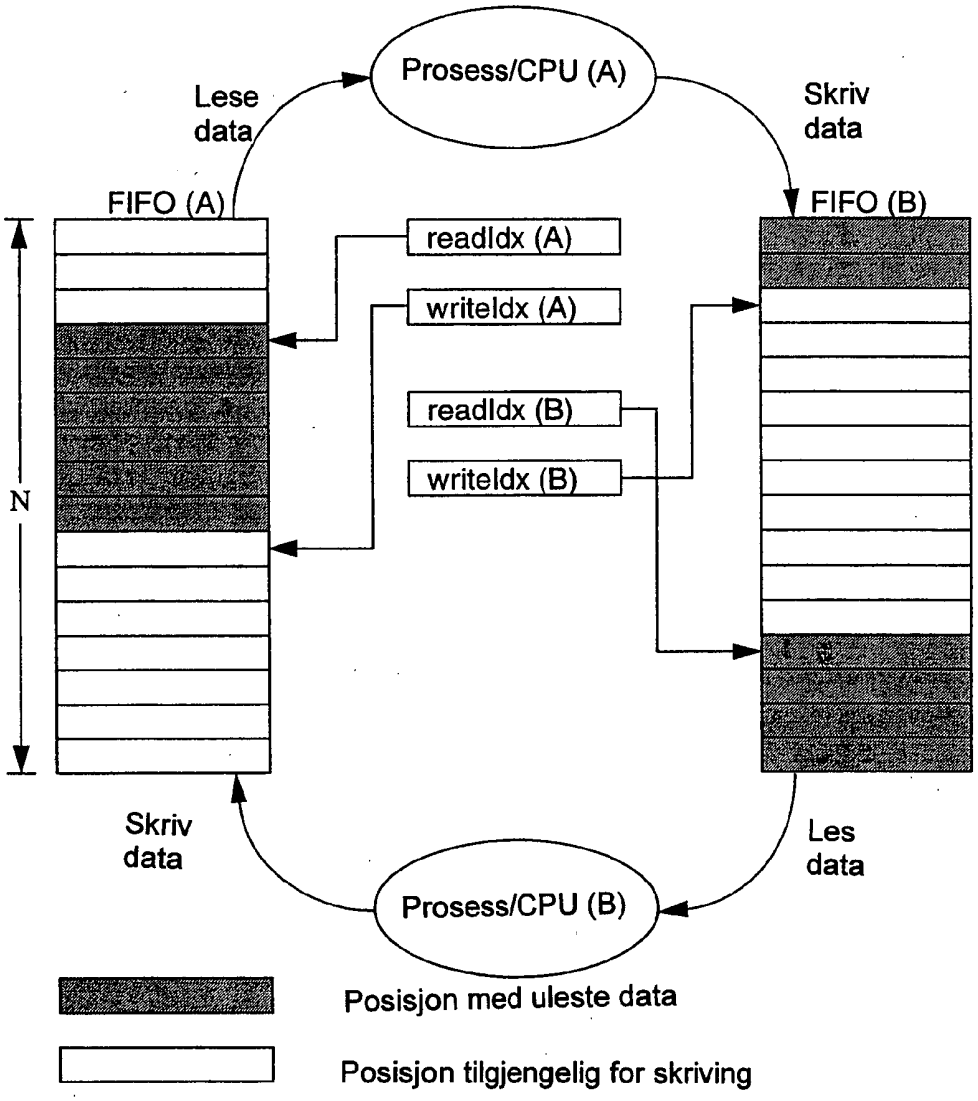
5. Arrangement ifølge krav 1 eller 4,
k a r a k t e r i s e r t v e d at de kommuniserende parter er bussmastere.

6. Arrangement ifølge krav 1 eller 4, k a r a k t e r i s e r t v e d at de kommuniserende parter er SW-prosesser.

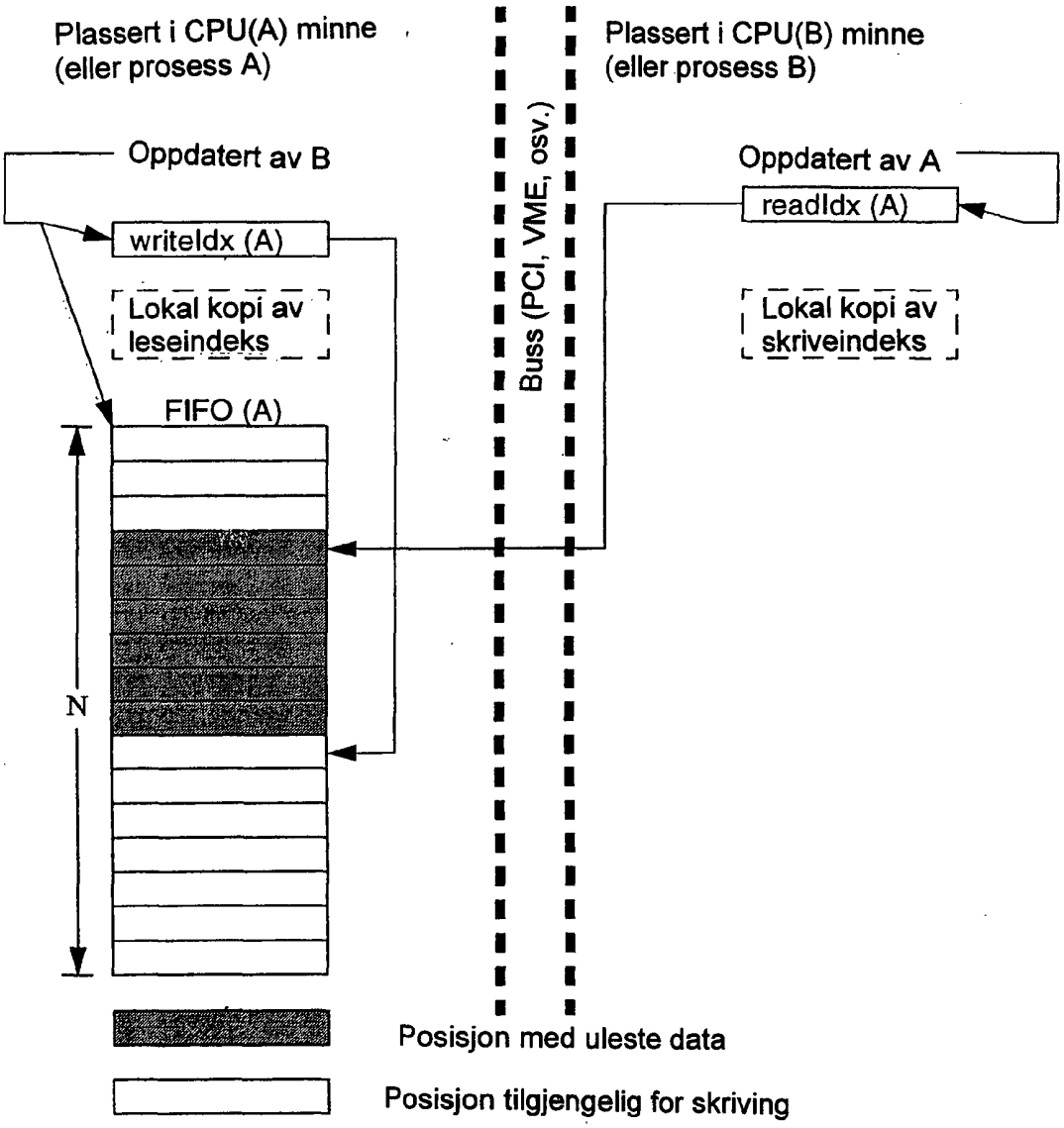
7. Arrangement ifølge krav 1 eller 4, k a r a k t e r i s e r t v e d at de kommuniserende parter omfatter en prosess og en avbruddsrutine.

8. Arrangement ifølge et av de foregående krav, k a r a k t e r i s e r t v e d at bussystemet er en PCI-buss.

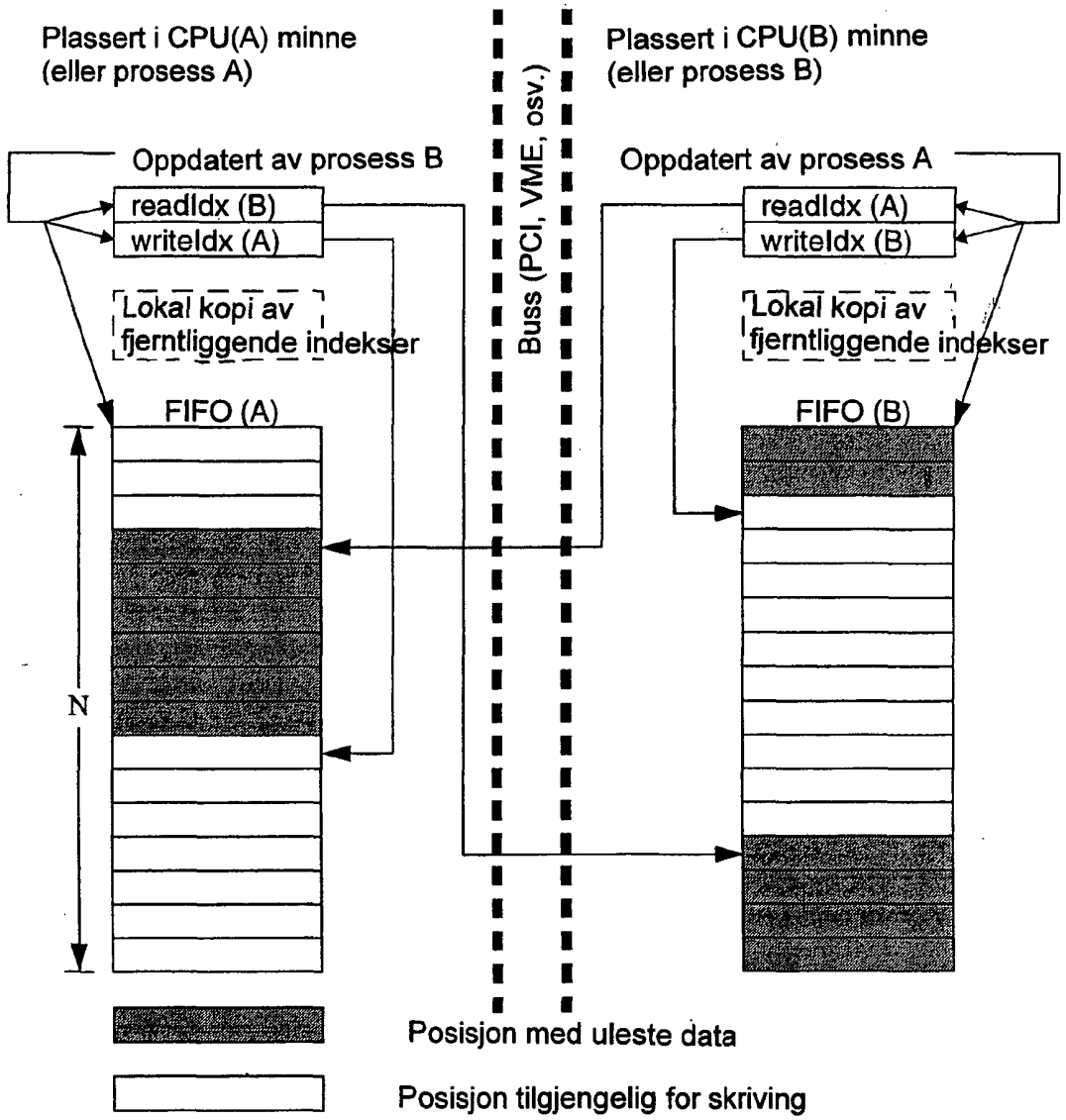
9. Arrangement ifølge et av de foregående krav, k a r a k t e r i s e r t v e d bussystemet er en VME-buss.


Prosess/CPU (A)
Lese data
Skriv data
FIFO (A)
FIFO (B)
readIdx (A)
writeIdx (A)
readIdx (B)
writeIdx (B)
N
Skriv data
Les data
Prosess/CPU (B)
Posisjon med uleste data
Posisjon tilgjengelig for skriving

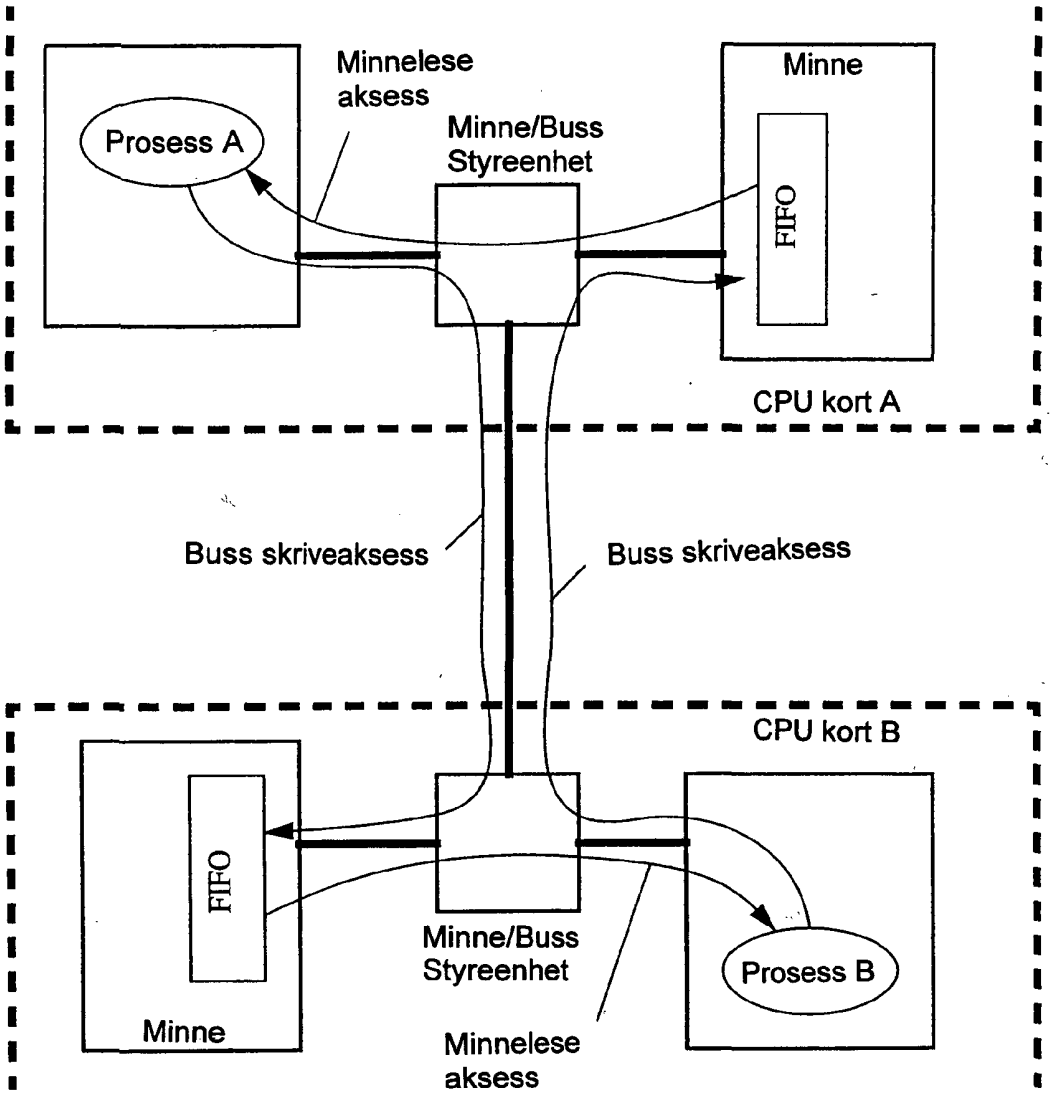
FIGUR 1


Plassert i CPU(A) minne
(eller prosess A)
Plassert i CPU(B) minne
(eller prosess B)
Buss (PCI, VME, osv.)
Oppdatert av B
writeIdx (A)
Lokal kopi av
leseindeks
FIFO (A)
N
Oppdatert av A
readIdx (A)
Lokal kopi av
skriveindeks
Posisjon med uleste data
Posisjon tilgjengelig for skriving

FIGUR 2


Plassert i CPU(A) minne
(eller prosess A)
Buss (PCI, VME, osv.)
Plassert i CPU(B) minne
(eller prosess B)
Oppdatert av prosess B
readIdx (B)
writeIdx (A)
Lokal kopi av
fjerntliggende indekser
FIFO (A)
N
Oppdatert av prosess A
readIdx (A)
writeIdx (B)
Lokal kopi av
fjerntliggende indekser
FIFO (B)
Posisjon med uleste data
Posisjon tilgjengelig for skriving

FIGUR 3


Minnelese
aksess
Minne
Prosess A
Minne/Buss
Styreenhet
FIFO
CPU kort A
Buss skriveaksess
Buss skriveaksess
CPU kort B
FIFO
Minne/Buss
Styreenhet
Prosess B
Minne
Minnelese
aksess

FIGUR 4