

(19)



SUOMI - FINLAND

(FI)

PATENTTI- JA REKISTERIHALLITUS

PATENT- OCH REGISTERSTYRELSEN

FINNISH PATENT AND REGISTRATION OFFICE

(10) **FI 932175 A7**

(12) **JULKISEKSI TULLUT PATENTTIHAKEMUS
PATENTANSÖKAN SOM BLIVIT OFFENTLIG
PATENT APPLICATION MADE AVAILABLE TO THE
PUBLIC**

(21) Patentihakemus - Patentansökan - Patent application 932175

(51) Kansainvälinen patenttiluokitus - Internationell patentklassifikation -
International patent classification
H04Q 11/06

(22) Tekemispäivä - Ingningsdag - Filing date 13.05.1993

(23) Saapumispäivä - Ankomstdag - Reception date 13.05.1993

(41) Tullut julkiseksi - Blivit offentlig - Available to the public 15.11.1993

(43) Julkaisupäivä - Publiceringsdag - Publication date 13.06.2019

(32) (33) (31) Etuoikeus - Prioritet - Priority

14.05.1992 US 882912

(71) Hakija - Sökande - Applicant

1 • Alcatel N. V., Strawinskylaan 341, 1077 XX Amsterdam, ALANKOMAAT, (NL)

(72) Keksijä - Uppfinnare - Inventor

1 • Bowdon, Edward K., USA, AMERIKAN YHDYSVALLAT, (US)

(74) Asiamies - Ombud - Agent

Kolster Oy Ab, Salmisaarenaukio 1, 00180 Helsinki

(54) Keksinnön nimitys - Uppfinningens benämning - Title of the invention

Menetelmä ja järjestelmä optimaalisen uudelleenjärjestelysekvenssin valitsemiseksi ristikytchentätietoliikennematriisille
Förfarande samt system för val av optimal omarrangemangsekvens för en korskopplingskommunikationsmatriis

Menetelmä ja järjestelmä optimaalisen uudelleenjärjestelysekvenssin valitsemiseksi ristikytkentätietoliikennematriisille

5 Esillä oleva keksintö liittyy yleisesti elektro-
niikkaan ja tarkemmin ottaen liikennejärjestelmiin. Vielä
tarkemmin ottaen keksintö koskee menetelmää ja järjestel-
mää optimaalisen uudelleenjärjestelysekvenssin valitsemik-
si tuloporttien kytkemiseksi ristikytkentäliikennematrii-
10 sin läpi lähtöporteille.

 Digitaaliset ristikytkentäliikenneverkon komponent-
tit, sellaiset kuin 1631 SX, valmistajana Alcatel Network
Systems Inc., on suunniteltu suorittamaan kytKentöjä tulo-
kytkimissä olevien tuloporttien ja lähtökytkimissä olevien
15 lähtöporttien välillä. Sellaisten ristikytKentäkomponent-
tien tunnetut konstruktiot hyödyntävät matriiseja tulo-
porttien kytkemisessä lähtöportteihin. Matriisien fyysisen
tilan ja valmistuskustannusten rajoittamiseksi suunnitte-
lijat pyrkivät minimoimaan matriisissa olevien ristikyt-
20 kimien määrän. Tämä tavoite tulee mutkikkaammaksi liikenne-
palveluiden markkinakysynnän kasvaessa. Erityisesti
verkkolaitteiden levitessä suuremmille markkinoille lait-
teiden tulee palvella yhä useampia tulo- ja lähtösignaale-
ja. Nämä tekijät pahentavat entisestään tila- ja kustan-
25 nusrajoituksiin liittyviä ongelmia liikennematriisilait-
teiden osalta.

 Yksi matriisirakenne, jossa käytetään minimaalista
matriisiristikytkimien määrää määrätylle tuloporttien ja
lähtöporttien määrälle ja joka tarjoaa potentiaalisesti
30 optimaalisen ratkaisun, tunnetaan "uudelleenjärjesteltävä-
nä" matriisina. Uudelleenjärjesteltävässä matriisissa vä-
liportaan kytkimien määrän tulee olla yhtä suuri kuin tu-
loporttien määrä kussakin matriisin tulokytkimessä tai
ylittää tämä määrä. Uudelleenjärjesteltävässä matriisissa
35 esiintyy joukko tilanteita siten, että vaikka laite ei

käytä kaikkia tuloportteja ja kaikkia lähtöportteja, yritys käyttää vapaata tuloporttia ja lähtöporttia on estetty, koska olemassa olevat kytkennät sulkevat signaalivirran matriisin läpi. Tämä voi tapahtua esimerkiksi jos olemassa olevat kytkennät jo käyttävät ainakin yhden yhteyden jokaisella mahdollisella polulla kysymyksessä olevien tulo- ja lähtöporttien välillä.

Uudelleenjärjesteltävässä verkossa on aina mahdollista vapauttaa signaalipolku vapaasta tuloportista vapaaseen lähtöporttiin siirtämällä olemassa olevia kytkentöjä verkossa. Termi "uudelleenjärjesteltävä" kuvaa sen vuoksi ominaisuutta, että verkon määrätyn tilan ja minkä tahansa vapaan tulo- ja lähtöporttiparin tapauksessa matriisin olemassa olevat kytkennät voidaan uudelleensijoittaa tarvittaessa uusille poluille vapaan parin kytkemiseksi.

Olemassa olevat menetelmät ja järjestelmät tulojen kytkemiseksi lähtöihin uudelleenjärjesteltävissä matriiseissa käyttävät yleensä standardia uudelleenjärjestelymenetelmää, joka määrittää mitkä matriisin ristikytkimet tulee uudelleenjärjestää signaalin virtauksen sallimiseksi. N.C. Paull kuvaa julkaisussa "Reswitching of Connection Networks", The Bell System Technical Journal, May 1962, ss. 833 - 856, tätä tunnettua menetelmää uudelleenjärjesteltävän matriisin vapauttamiseksi. Tämä menetelmä (nimitetään tämän jälkeen Paullin menetelmäksi) kärsii suuresta rajoituksesta. Paullin menetelmä vaatii joiden ristikytkeiden katkaisun ja joidenkin muiden ristikytkeiden kytkemisen matriisin järjestämiseksi uudelleen. Tämä menettely vaatii aikaa ja johtaa epätoivottaviin palveluviiveisiin tai keskeytyksiin matriisin uudelleenjärjestelyn aikana.

Esillä olevan keksinnön tavoitteena on sen vuoksi tarjota menetelmä ja järjestelmä, joka sallii välittömästi kytkeä vapaat tuloportit vapaisiin lähtöportteihin ja, jos esiintyy uudelleenjärjesteltävästi estetty tila, uudel-

leenjärjestää matriisi välittömän kytkennän jälkeen. Pro-
sessin tärkeä näkökohta on uudelleenjärjestää matriisi
käyttämällä minimaalista määrää olemassa olevien kytkentö-
jen uudelleenjärjestelyjä. Esillä oleva keksintö sen vuok-
5 si saavuttaa tämän tavoitteen minimaalisella lisäpiirien
määrällä ja välttää tunnettujen uudelleenjärjestelymene-
telmien ja järjestelmien palveluviiveet ja keskeytykset.

Esillä olevan keksinnön kohteena on myöskin tarjota
menetelmä ja järjestelmä, joilla määritetään minimimäärä
10 uudelleenjärjestelyjä uudelleenjärjesteltävästi estetyille
liikennematriisille esittämällä liikennematriisi neliömat-
riisilla, jonka ensimmäinen ulottuvuus edustaa liikenne-
matriisiin tuloporrasta ja toinen ulottuvuus edustaa lii-
kennematriisin lähtöporrasta siten, että neliömatriisissa
15 solut edustavat ensimmäisen ulottuvuuden risteyskohtia
toisen ulottuvuuden kanssa ja osoittavat mahdolliset väli-
portaan kytkennät matriisin tuloportaan ja lähtöportaan
välillä, ja jossa neliömatriisi identifioi estetyn väli-
portaan kytkimen. Pari väliportaan kytkimiä, joista toinen
20 ei liity samaan lähtöportaan kytkimeen kuin estetty väli-
portaan kytkin ja joista toinen ei liity samaan tulopor-
taan kytkimeen kuin estetty väliportaan kytkin, identi-
fioidaan ja määritetään uudelleenjärjestelyaskelten, jotka
tarvitaan ensimmäistä uudelleenjärjestelysekvenssiä varten
25 tuloportaasta liikennematriisin läpi lähtöportaaseen käyt-
tämällä ensin parin toista väliportaan kytkintä, määrä, ja
määritetään uudelleenjärjestelyaskelten, jotka tarvitaan
toista uudelleenjärjestelysekvenssiä varten tuloportaasta
liikennematriisin läpi lähtöportaaseen käyttämällä toista
30 väliportaan kytkintä, määrä. Menetelmä suorittaa uudel-
leenjärjestelysekvenssit asteittaisina rinnakkaisina as-
kelina ja päättyy määritettyään ensin lyhyemmän mainituis-
ta kahdesta uudelleenjärjestelysekvenssistä, jotka väli-
portaan kytkinpari tuottaa.

Muut tavoitteet selviävät ja ne ymmärretään parhaiten lukemalla seuraava havainnollistavien toteutusten kuvaus yhdessä oheisten piirrosten kanssa, joissa:

5 kuvio 1 esittää viisiportaista kytkentäverkkoa (TSSST), johon sisältyy kolmeportainen tilamatriisi (SSS);

 kuvio 2 esittää yksityiskohtaisemmin kolmeportaista kytkentäverkkoa, joka voidaan sisällyttää loogisesti kuvion 1 viisiportaiseen TSSST-verkkoon;

10 kuvio 3 on kolmeportainen kytkentäverkko, joka on yleistetty $N(m,n,r)$ -tapaukseksi;

 kuvio 4 esittelee merkitsemistavan edullisen toteutuksen ymmärtämiseksi;

 kuvio 5 tarjoaa edullisen toteutuksen mukaisen menetelmän vuokaavion;

15 kuviot 6 - 14 kuvaavat esimerkkiä edullisten toteutusten mukaisesta sekvenssingenerointiosuudesta; ja

 kuviot 15 - 34 kuvaavat edullisen toteutuksen yksinkertaistettua aika-alueen esimerkkiä;

20 kuviossa 1 on esitetty suuri matriisikonfiguraatio, joka tarjoaa kytkennän ja testisisääntulon korkeintaan 1024 DS3-portille (tai vastaavasti 32678 DS1-portille) 1631 SX -laitetta varten. Tämä on viisiportainen aika-tila-tila-tila-aika-jakoisesti (TSSST) multipleksoitu kytkentämatriisiverkko 50. Matriisikonfiguraatio 50 sisältää
25 aikaportaan 52, joka yhteyksien 54 kautta kytkeytyy kolmeen tilaväliportaaseen (SSS), joita on merkitty yleisesti viitenumerolla 56. Kolme tilaväliporrasta SSS 56 kytkeytyy lähtölinjan 58 kautta aikälähtöportaaseen 60. Tässä konfiguraatiossa verkko 50 sisältää aikaportaiden 52 ja 60
30 aika-alueen ja kolmen tilaväliportaan 56 tila-alueen. Kuvio 1 havainnollistaa loogisesti aikaportaiden 52 ja 60 ja tilaväliportaiden 56 välistä riippuvuutta. Tuloaikaporras 52 käyttää 128 aikaväliä tuloa 62 kohden. Aikavälejä voidaan esittää I/O-hyllyn tulojen 62 kytkennöillä matriisiin
35 56 kuhunkin aikaväliin 66. Tilaväliporras 56 sisältää 128

aikaväliä 66, joista kukin aikaväli 66 sisältää $N(17,16,16)$ matriisin. Aikavälimatriisin 66 numero 17 edustaa väliportaan kytkinten määrää; ensimmäinen numero 16 edustaa tulojen määrää kuhunkin tuloportaan kytkimeen ja toinen 16 edustaa tuloportaan kytkimien määrää. (Symmetrian vuoksi kustakin lähtöportaan kytkimestä on 16 lähtöä ja 16 lähtöportaan kytkintä.) Aikalähtöporras 60 käyttää 128 aikavälikytkentää, sellaista kuin kytkentä 68, kullekin I/O-hyllyyn menevälle matriisin lähdölle 70.

10 Kuviossa 2 tilaväliporras 56 on esitetty matemaattisesti kolmeportaisena matriisien 72 kytkentäverkkokuviona, joille kullekin symboli $N(17,16,16)$ kuvaa täysin kunkin matriiseista. Siten kullekin $N(17,16,16)$ -matriisille 72 kuvion 2 esimerkissä on 16 tulokytkintä, sellaista
15 kuin tulokytkin 74, joista kukin vastaanottaa 16 tuloporttia, sellaista kuin tuloportti 76. Tuloportit 76 muodostavat yhdessä kuvion 1 I/O-hyllyn ja aikavälin väliset kytkennät kullekin aikavälille. Kustakin tulokytkimestä 74 menee tulokytkin-välikytkin-yhteys 78 välikytkimiin, sellaisiin kuin välikytkin 80. Edullisessa toteutuksessa kukin $N(17,16,16)$ -matriisi 72 sisältää 17 väliportaan kytkintä 80. Kustakin väliportaan kytkimestä 80 menee 16 lähtökytkinyhteyttä 82 lähtökytkimiin, sellaisiin kuin lähtökytkin 84. Kukin lähtökytkin 84 tarjoaa 16 lähtöporttia, sellaista kuin lähtöportti 86, jotka kytkeytyvät I/O-hyllyihin, sellaisiin kuin kuvion 1 I/O-hylly 70. Lähtöportit 86 muodostavat yhdessä kuvion 1 aikaväli-I/O-hylly-yhteydet kutakin aikaväliä varten.

Yhteydet muodostetaan tuloportin 76 ja lähtöportin 86 välillä välittömästi käyttämällä esillä olevaa kytkentämenetelmää. Tämä menetelmä muodostaa kytkennän käyttämällä tulokytkimen 74, välikytkimien 80 ja lähtökytkimien 84 välillä olevaa reittiä, jossa yritetään käyttää matriisin 72 eniten käytettyä osaa ensiksi ja matriisin 72 vähiten käytettyä osaa viimeiseksi. Käyttämällä matriisin run-

saasti kuormitettua osaa silloin kun se on mahdollista määrättyä yhteyttä luotaessa, matriisin 72 uudelleenjärjestelyn ollessa välttämätöntä edullisen toteutuksen mukainen uudelleenjärjestelyprosessi kohtaa elementtejä joilla on pienin määrä olemassa olevia kytkentöjä. Tämä minimoi matriisin uudelleenjärjestelyssä tarvittavien las-

5 kutoimitusten määrän.

Edullisessa toteutuksessa käytetään nimettyä aikaväliä, kun uudelleenjärjestely on tarpeen aika-alueella.

10 Esimerkiksi kuviota 1 tarkasteltaessa 128. aikaväli voidaan nimetä ylimääräiseksi aikaväliksi, jota käytetään matriisin 50 uudelleenjärjestelyn aikana. Samalla tavoin tila-alueella uudelleenjärjestelyssä käytetään nimettyä tai ylimääräistä välikytkintä. Esimerkiksi kuviota 2 tarkasteltaessa välikytkin 17 voi olla kyseinen nimetty tai

15 ylimääräinen välikytkin. Esillä oleva kytkentämenetelmä käyttää automaattisesti nimettyä tai ylimääräistä elementtiä (joko nimettyä tai ylimääräistä aikaväliä aika-alueella tai nimettyä väliportaan kytkintä tila-alueella), kun

20 esiintyy uudelleenjärjesteltävästi suljettu tilanne.

Kuvio 3 esittää, että kuvion 2 $N(17,16,16)$ -matriisi 72 voidaan yleistää matriisiksi 88, jolle esitys $N(m,n,r)$ kuvaa konfiguraation. Noudattamalla matriisin 16 esitystapaa $N(17,16,16)$, yleistetty merkintä $N(m,n,r)$ kuvaa

25 yleistetyn matriisin 88. $N(m,n,r)$ -esitysmuodossa m on yhtä kuin rxr väliportaan kytkimien määrä (kuin myös lähtöjen määrä kustakin $n \times m$ tuloportaasta ja tulojen määrä kuhunkin $m \times n$ lähtöportaan kytkimeen); n on yhtä kuin tulojen määrä kuhunkin $n \times m$ tuloportaan kytkimeen (kuin myös lähtöjen

30 määrä kustakin $n \times m$ lähtöportaan kytkimestä); ja r on yhtä kuin $n \times m$ tuloportaan kytkimien määrä (kuin myös $m \times n$ lähtöportaan kytkimien määrä). Huomaa myös, että matriisissa 88 m on ainakin yhtä suuri kuin $n+1$. Tämä $m:n$ rajoitus määrätyn $n:n$ tapauksessa takaa, että on olemassa ainakin

35 yksi ylimääräinen väliportaan kytkin.

Tarkasteltaessa tarkemmin kuviota 3, yleistetyssä matriisissa 88 on n tuloporttia 90, jotka kytkeytyvät tulokytkimiin, sellaisiin kuin tulokytkin 92, joissa on m lähtökytkentää 94. Yleistetyssä matriisissa 88 voi olla

5 esimerkiksi r tulokytkintä 92. Tulokytkimestä välikytkimeen olevat kytkennät 94 kytkeytyvät rXr välikytkimiin, sellaisiin kuin välikytkin 96, jossa kukin välikytkin 96 vastaanottaa r välikytkimen kytkentää 94 ja tarjoaa r lähtökytkentää 98. Kuvion 3 esimerkissä on m välikytkintä 96.

10 Kukin välikytkimestä lähtökytkimeen oleva kytkentä 98 välikytkimestä 96 kytkeytyy lähtökytkimeen 100. Kukin lähtökytkin 100 vastaanottaa m välikytkimestä lähtökytkimeen olevaa kytkentää 98 ja sen lähtönä on n lähtöporttia, sellaista kuin lähtöportti 102.

15 Sekä kuvion 2 $N(17,16,16)$ -matriisin 72 että kuvion 3 $N(m,n,r)$ -matriisin 88 sanotaan olevan uudelleenjärjesteltäviä matriiseja, jos vastaavasti määrätyn matriisin tilan tapauksessa ja minkä tahansa annetun tuloporttien 76 tai 90 ja lähtöporttien 86 tai 102 vapaan parin tapauksessa matriisien sisällä olevat kytkennät voidaan tarvittaessa uudelleensijoittaa uusille poluille, jotta vastaavien vapaiden parien välinen kytkentä voidaan suorittaa. Matriisit ovat uudelleenjärjesteltäviä vain, ja vain silloin, kun $m \geq n$. Merkitsemisen helpottamiseksi esityksellä

20 $N(m,n,r)$ kuvatussa verkossa merkintä $\Delta(m,n,r)$ kuvaa kytkentöjen, jotka tulee uudelleenjärjestää vapaan tulo- ja lähtöportin parin kytkemiseksi toisiinsa, määrää. Paullin menetelmä seuraa ominaisuudesta, että $\Delta(n,n,n) \leq n-1$. Toisin sanoen $N(n,n,n)$ -matriisin tapauksessa on tarpeen

25 siirtää korkeintaan $n-1$ kytkentää vapaan tuloportin kytkemiseksi vapaaseen lähtöporttiin. Tämän ominaisuuden määrätty sovellutus tarkoittaa esimerkiksi, että kuvion 2 matriisin 72 tapauksessa uudelleenjärjestäminen vaatii korkeintaan $n-1 = 17-1 = 16$ kytkentämuutosta vapaan tulo-

30 portin kytkemiseksi vapaaseen lähtöporttiin.

35

Seuraavassa esityksessä kuvio 4 esittelee merkin-
 nän, joka helpottaa edullisen toteutuksen ymmärtämistä.
 Kuviossa 5 esitetään vuokaavio, joka havainnollistaa vä-
 littömän kytkemisen menetelmän peruskäsitteitä ja esillä
 5 olevan keksinnön mukaista järjestelmää. Sen jälkeen ku-
 vioissa 6 - 14 esitetään esimerkki menetelmän ja järjes-
 telmän toiminnasta kun esiintyy kolmen väliportaan tila-
 matriisin 56 matriisin 72 uudelleenjärjesteltävästi estet-
 ty tila. Jotta esillä oleva keksintö voitaisiin ymmärtää
 10 vielä yksityiskohtaisemmin, menetelmän ja järjestelmän
 toiminta kuvataan kuvioilla 15 - 34, joissa käytetään yk-
 sinkertaistettua $N(5,4,4)$ uudelleenjärjesteltävää matrii-
 sia, joilla on kyky suorittaa välitön kytkentä uudelleen-
 järjesteltävästi estetyssä tilanteessa.

15 Kuviossa 4 merkintätavassa, joka on hyödyllinen
 esillä olevan keksinnön ymmärtämisen kannalta, käytetään
 neliömatriisia 110 edustamaan kytkentöjä, jotka esiintyvät
 tila-alueen matriisissa, sellaisessa kuin kuvion 3 matrii-
 si 88. Neliömatriisissa 110 on rivejä 112, jotka merkit-
 20 sevät tulokytкимиä (sellaisia kuin kuvion 3 tulokytkin 92)
 ja sarakkeita 114, jotka edustavat lähtökytkimiä (sellai-
 sia kuin kuvion 3 lähtökytkin 100). Kuvion 4 yksinkertais-
 tettu esimerkki esittää tapauksen, jossa rivien 112 ja
 sarakkeiden 114 määrä on yhtä kuin 8. Huomaa, että jotta
 25 kuvio 5 vastaisi tarkalleen kuvion 2 $N(17,16,16)$ -matrii-
 sia, siinä voitaisiin esittää matriisi jossa on 16 riviä,
 jotka edustavat tulokytкимиä 74, ja 16 saraketta, jotka
 edustavat lähtökytkimiä, sellaisia kuin lähtökytkin 84.
 Kussakin neliömatriisin positiossa, sellaisessa kuin mat-
 30 riisin positio 116, on m mahdollista symbolia jotka voivat
 esiintyä. Nämä m symbolia vastaavat välikytkimiä, sellai-
 sia kuin kuvion 3 väliportaan kytkin 96. Esimerkiksi mat-
 riisipositiossa 116 voi olla m mahdollista arvoa, yksi
 kullekin väliportaan kytkimelle 96, jotka näkyvät kuviossa
 35 3.

Kuviossa 4 järjestetty pari $(3,1)$ merkitsee matriisipositiota 116, jossa 3 on rivin merkintä, so. rivi 3 jolla on viitenumero 118, ja 1 on sarakepositio, so. sarake 1 jolla on viitenumero 120. Arvo B ($1 \leq B \leq m$) matriisipositiossa $(3,1)$ vastaa kytkentää tulokytkimestä 3 välikytkimen B kautta lähtökytkimeen 1. Tyhjä paikka matriisipositiossa, sellainen kuin tyhjä matriisipositio $(3,2)$ johon viitenuoli 122 osoittaa, ilmaisee ettei mitään kytkentää ole tulokytkimen 3 ja lähtökytkimen 2 välillä.

Esimerkiksi kuvion 3 tapauksessa, koska siinä voi olla vain n tuloporttia 90 kuhunkin tulokytkimeen 92 ja n lähtöporttia 102 kustakin lähtökytkimestä 100, vain n symbolia voi esiintyä millä tahansa rivillä 112 tai sarakkeella 114. Koska jokaisella tulokytkimellä 92 on vain yksi kytkentä 94 kuhunkin välikytkimeen 96, mitkään kaksi symbolia millä tahansa rivillä 112 eivät voi olla samoja. Samalla tavoin kullakin lähtökytkimellä 100 on vain yksi kytkentä 98 kustakin välikytkimestä 96. Mitkään kaksi symbolia missä tahansa sarakkeessa 114 eivät saa olla samoja.

Kuvion 4 merkitsemistavalla neliömatriisin 110, joka täyttää nämä rajoitukset, sanotaan olevan "laillinen", koska siinä on vain laillisia merkintöjä.

Huomaa, että triviaali estotapaus sattuu, jos kaikki tulolinjat tulokytkimeen, sellaiseen kuin tulokytkin 1 jota rivi 1 merkitsee, on jo kytketty. Tämä vastaisi tilannetta jossa rivillä 1 on n symbolia (esim. 16 symbolia kuvion 2 tapauksessa). Samalla tavalla jos kaikki tulolinjat esimerkiksi lähtökytkimeen 3 olisi jo kytketty vastaaviin n symboliin sarakkeessa 3 (esim. 16 symboliin kuvion 2 esimerkin tapauksessa), kuvion 2 verkkomatriisi 72 ja kuvion 3 verkkomatriisi 88 on triviaalisti estetty, ja neliömatriisi 110 edustaa tätä tosiasiaa.

Kuviossa 5 vuokaavio 140 kuvaa esillä olevan keksinnön edullisen toteutuksen vaiheet ja toiminnot. Alkaen aloitusaskeleesta 142 tässä menetelmässä on ensin kysely,

kuten kyselylohko 144 osoittaa, onko olemassa porttiparia joiden välillä tulee suorittaa kytkentä. Kuvion 3 esimerkissä askeleessa kysytään, onko vapaata tuloporttia 90, jolla voidaan kytkeytyä vapaaseen lähtöporttiin 102. Jos sellainen pari on olemassa, vuo siirtyy lohkoon 146 aika-alueen kytkentöjen aloittamiseksi. Jos kytkettävää porttiparia ei ole, vuo palaa lohkoon 148, joka kuvataan alla. Aloita aika-alue -askeleessa 146 vuo siirtyy lohkoon 150, jossa menetelmä valitsee numeroltaan alhaisimman aikavälin, joka on käytettävissä kytkentään. Sen jälkeen alkavat tila-alueen operaatiot lohkossa 152, joilla operaatioilla valitaan numeroltaan alhaisin kytkentää varten vapaana oleva väliportaan tulokytkin askeleessa 154. Kun numeroltaan alhaisin väliporras on valittu kytkentää varten, kytkennät suoritetaan lohkossa 156 ja neliömatriisi 110, tai samanlainen kytkentöjä osoittava tilataulu, päivitetään lohkossa 158.

Lohkossa 160 tapahtuu kysely siitä, vaaditaanko tilan uudelleenjärjestely. Jos niin on, vuo palaa lohkoon 162, jossa askeleen tehtävänä on valita kaksi väliportaan kytkintä uudelleenjärjestelyä varten. Seuraavaksi lohkossa 164 askel etsii uudelleenjärjesteltävät kytkennät, ja kytkentöjen perusteella lohkossa 166 suoritetaan kysely siitä, mikä polku vaatii pienimmän määrän uudelleenjärjestelyjä. Jos polku X→A vaatii pienimmän määrän uudelleenjärjestelyjä, niin ohjelmavuo noudattaa polkua 168. Jos X→B vaatii pienimmän määrän uudelleenjärjestelyjä, niin ohjelmavuo noudattaa polkua 170. Riippumatta siitä, mitä polkua ohjelmavuo noudattaa, askeleessa 172 kytkennät uudelleenjärjestelään yksi kerrallaan käyttämällä "törmäyksetöntä rullausta". Alla oleva selvitys määrittelee termin "törmäyksetön rullaus". Vuo palaa askeleeseen 174 aika-alueen operaatioiden päättämiseksi. Huomaa, että jos mitään tilauudelleenjärjestelyä ei tarvittu, niin kyselylohkosta 160

vuo siirtyy suoraan 'päättä aika-alue'-askeleeseen lohkoissa 174.

Kyselyssä 176 kysymyksenä on, vaaditaanko aika-uudelleenjärjestelyä. Jos niin on, vuo palaa askeleeseen 178, jossa menetelmän kohdassa valitaan kaksi aikaväliä uudelleenjärjestelyä varten. Sen jälkeen lohkoissa 180 askeleena on löytää uudelleenjärjesteltävät kytkennät. Lohkoissa 182 tapahtuu kysely siitä, mikä polku vaatii pienimmän määrän uudelleenjärjestelyjä. Jos X→A-polku vaatii pienimmän määrän uudelleenjärjestelyjä, vuo kulkee polkua 184 pitkin. Toisaalta jos X→B-polku vaatii pienimmän määrän uudelleenjärjestelyjä, vuo seuraa polkua 186. Seuraava askel, riippumatta siitä noudattaako ohjelmavuo polkua 184 tai 186, uudelleenjärjestää kytkennät yksi kerrallaan käyttämällä jälleen törmäyksetöntä rullausta aika-alueella askeleessa 188. Ohjelmavuo siirtyy sitten askeleeseen 190 aika-alueen operaatioiden päättämiseksi. Huomaa, että jos kyselylohko 176 havaitsee, ettei mitään aikauudelleenjärjestelyä vaadita, vuo etenee suoraan 'päättä aika-alue'lohkoon 190.

'Päättä aika-alue'-lohkosta 190 vuo siirtyy kyselyyn 148, jossa menetelmä määrittää, onko olemassa porttiparia joiden välinen yhteys puretaan. Jos ei ole, vuo palaa kyselylohkoon 144 edetäkseen kuten aiemmin kuvattiin. Jos on olemassa porttipari joiden välinen yhteys puretaan, tässä kohden poistetaan lohkoissa 192 pyydetyt kytkennät ja päivitetään sen jälkeen datataulukot, jotka kuvaavat matriisin kytkentöjä, askeleessa 194.

Kyselyssä 196 menetelmä määrittää, onko "pakkaa kytkennät purkamisen jälkeen" -toiminne sallittu. Jos niin on, vuo siirtyy kyselyyn 198 tutkiakseen, onko uudelleenjärjestely käynnissä. Jos niin on, vuo kulkee silmukkaa 200 kunnes uudelleenjärjestely ei enää ole käynnissä. Vaikka kuviossa 5 esitetään purkamisen tapahtuvan minkä tahansa tarvittavan uudelleenjärjestelyn loppuun saattami-

sen jälkeen, esillä oleva menetelmä ja järjestelmä voi prosessoida purkamispyynnön rinnakkaisesti uudelleenjärjestelyn kanssa. Sen vuoksi jos esillä olevan keksinnön mukainen järjestelmä pyrkii suuntaamaan kytkennät ensin matriisiin eniten käytettyyn osaan, kutsujen pakkaaminen purkamisen jälkeen, lohkon 198 uudelleenjärjestely käynnissä, estää yritykset pakata uudelleenjärjestelyn aikana.

Jos uudelleenjärjestely ei ole käynnissä, kyselylohkosta 198 seuraava askel on 'määritä tilassa pakattavat kytkennät' askeleessa 202. Vuokaavion 140 seuraava askel on sen jälkeen pakata kytkennät aika-alueella käyttämällä törmäyksetöntä rullausta lohkoissa 204. Ja sen jälkeen lohkoissa 206 on askel jossa määritetään ajassa pakattavat kytkennät, jonka jälkeen kytkennät pakataan lohkoissa 206 ajassa käyttämällä automaattisesti törmäyksetöntä rullausta. Ohjelmavuo palaa sitten pisteeseen 210 jatkaakseen aiemmin kuvattuja vuokaavion 140 operaatioita.

Seuraavassa esityksessä kuvataan kuinka kuvion 5 vuokaavion 140 askeleet liittyvät kuvion 4 neliömatriisiin 110. Muistettakoon, että neliömatriisiin 110, tai samankaltaisen taulukon, päivittämisen jälkeen askeleessa 158 edullisen toteutuksen mukainen menetelmä ilmaisee, esiintyykö uudelleenjärjesteltävästi estetty tila. Siten kun matriisikonfiguraatiossa käytetään nimettyä elementtiä (joko aikaväli aika-alueella tai väliportaan kytkin tila-alueella) tekemään välitön kytkentä vapaan tuloportin ja vapaan lähtöportin välillä, uudelleenjärjesteltävän esto-tilan esiintyminen ilmaistaan automaattisesti.

Askeleet 162 ja 178 suorittavat Paullin menetelmän, jotta löydetään polku joka vaatii vähiten uudelleenjärjestelyjä. Erityisesti estetyn solun ollessa merkitty (r_1, c_1) , kuten kuvion 4 solu 121, menetelmä testaa löytääkseen kaikki symboliparit (A, B) siten, että A on rivillä 1, mutta ei sarakkeessa 1, ja B on sarakkeessa 1, mutta ei rivillä 1. Määrätyille symbolipareille, esimerkiksi (A, B) ,

on kaksi mahdollista uudelleenjärjestelysekvenssiä, joista toinen on lyhyempi. Tämän määrittäminen on osa askelta 16 tila-alueella ja askelta 182 aika-alueella.

Sekvenssi "X→A" merkitsee sekvenssiä, joka alkaa välittömällä kytkennällä nimetyn elementin läpi (esim. kuvion 2 väliportaan kytkin 17 tila-alueella ja kuvion 1 aikaväli 128 aika-alueella) ja päättyy estettyyn soluun, jota ollaan varaamassa elementiksi A. Sekvenssi alkaa esimerkiksi alkaen estetystä solusta (r_1, c_1) kuvion 4 kohdassa 121). Koska yhtään B:tä ei ole rivillä r_1 , tapahtuu A:n haku r_1 :ssä. A:n tulee olla rivillä r_1 , muussa tapauksessa matriisi ei ole estetty. Sen jälkeen kun rivillä r_1 oleva A on löydetty, se ympyröidään tai merkitään.

Oletetaan silloin yleisesti, että ympyröity A on kohdassa (r_j, c_k) , $1 \leq j, k \leq 8$. Silloin tapahtuu B:n haku sarakkeesta c_k . Jos sarakkeessa c_k ei ole yhtään B:tä, sekvenssi päättyy ja sen jälkeen seuraavana askeleena on löytää uudelleenjärjesteltävät kytkennät lohossa 164 tai 180, tilanteen mukaan. Jos B esiintyy sarakkeessa c_k , seuraava askel on ympyröidä B ja jatkaa. Jos ympyröity B on positiossa (r_k, c_j) , tapahtuu A:n haku rivistä r_k . Sekvenssi X→A jatkuu ympyröiden vuorotellen B:tä sarakkeista ja A:ta riveiltä, kunnes esiintyy sarake jossa ei ole yhtään ympyröitävää B:tä tai rivi jossa ei ole yhtään ympyröitävää A:ta. Huomaa, että kaikkia A-kirjaimia ja B-kirjaimia ei ympyröidä, vaan ainoastaan ne jotka kohdataan yllä mainitussa haussa.

X→B-sekvenssin generointi tapahtuu samalla tavalla seuraavasti. Aloitettaessa esimerkiksi kuvion 4 estetystä solusta (r_1, c_1) , sarakkeessa 1 ei ole A:ta, joten alkaa B:n etsintä sarakkeesta c_1 . Sarakkeessa c_1 täytyy olla B, muussa tapauksessa matriisi ei olisi estetty. Sen jälkeen menettelynä on ympyröidä tuo B. Koska ympyröity B on positiossa (r_3, c_1) , alkaa A:n etsintä riviltä r_3 . Jos r_3 :ssa ei ole A:ta, niin tämä sekvenssi päättyy ja ohjelmavuo jatkuu

askeleeseen 164 tai 180, kuten tila- tai aika-alueen osalta on soveliaista. Muussa tapauksessa menettelynä on ympyröidä A ja jatkaa. Tässä esimerkissä ympyröity A on positi-
 5 Jos sarakkeessa c_3 ei ole B:tä, niin sekvenssi päättyy. Muussa tapauksessa sekvenssin tulee ympyröidä B ja jatkaa. Tämä proseduuri jatkuu ympyröimällä vuorotellen B-kirjaimia sarakkeissa ja A-kirjaimia riveillä, kunnes löydetään joko sarake jossa ei ole ympyröitävää B:tä tai rivi jossa
 10 ei ole ympyröitävää A:ta. Huomaa, että kaikki A:t ja kaikki B:t eivät tule ympyröidyiksi, vaan ainoastaan ne jotka löydetään yllä olevassa haussa.

Annetulle symboliparille (A,B) seuraavana askeleena on valita sekvenssin (joko $X \rightarrow A$ -sekvenssi tai $X \rightarrow B$ -sekvenssi) generoima polku, joka sisältää pienimmän määrän ympyröityjä symboleja (so. A:t ja B:t). Kaikille sellaisille symbolipareille askeleen tulee valita pari (A,B), joka
 15 sisältää pienimmän määrän ympyröityjä symboleja (joko A:t tai B:t). Tämä sekvenssi määrää minimimäärän uudelleenjärjestelyjä, jotka tarvitaan polun muodostamiseen uudelleenjärjesteltävän matriisin tulosta sen lähtöön.

Kaikkien $X \rightarrow A$ - ja $X \rightarrow B$ -sekvenssien täydellisen generoinnin sijasta edullinen toteutus generoi lyhyimmän sekvenssin. Tämä suoritetaan lohkoissa 164 tila-alueella ja
 25 lohkoissa 180 aika-alueella käyttäen edullisen toteutuksen menetelmää, jota tämän jälkeen kutsutaan ylivuotoalgoritmiksi. Ylivuotoalgoritmi aloitetaan etsimällä $X \rightarrow A$ -sekvenssin ensimmäinen elementti. Jos $X \rightarrow A$ -sekvenssi ei ole päättynyt, etsitään $X \rightarrow B$ -sekvenssin ensimmäinen elementti. Sen
 30 jälkeen, jos $X \rightarrow B$ -sekvenssi ei ole päättynyt, etsitään $X \rightarrow A$ -sekvenssin seuraava elementti. Tämä sekvenssi jatkuu kunnes generoidaan lyhyempi näistä kahdesta sekvenssistä. Hetkellä, jolloin lyhyempi kahdesta sekvenssistä generoidaan, haku päättyy.

Piste, jossa ylivuotoalgoritmi päättyy, määrää minimimäärän uudelleenjärjestelyjä sisältävän polun, kuten lohkoissa 166 tila-alueella ja lohkoissa 182 aika-alueella on osoitettu.

5 Tila-alueella askeleessa 172 ja aika-alueella askeleessa 188 uudelleenjärjestelyt tapahtuvat käyttämällä törmäyksetöntä rullausta. Ympyröityjen symbolien, joko A:t tai B:t, sekvenssi, jonka yllä olevat askeleet määrittävät, muutetaan sitten vapauttamaan estetty solu käyttäen
10 soveltuvaa $X \rightarrow A$ tai $X \rightarrow B$ rullausta riippuen siitä mikä tuottaa minimimäärän uudelleenjärjestelyjä.

Seuraava taulukko 1 ja siihen liittyvä teksti kuvaavat kytkentöjen suorittamisen askeleet väliportaana kytkimien uudelleenjärjestämiseksi, kun $X \rightarrow A$ johtaa minimimäärään kytkentöjen uudelleenjärjestelyjä:
15

TAULUKKO 1	
ASKEL	TOIMENPIDE
1	$X - (r, c_1)$
2	$\textcircled{B} - \textcircled{B}, X$
3	$\textcircled{B}, X - X$
4	$\textcircled{A} - \textcircled{A}, B$
5	$\textcircled{A}, B - B$
6	$X - X, A$
7	$X, A - A$

Jos $X \rightarrow A$ -sekvenssi tuottaa minimimäärän uudelleenjärjestelyjä, silloin tarkasteltaessa kuvion 4 neliömat-
35

riisia 110 ja kuvion 5 askelta 172 tai 188 tilanteen mukaan, menetelmän tulee laittaa "X" (r_1, c_1)-positioon, mikä osoittaa että kytkentä tehtiin ylimääräisen väliportaan kytkimen kautta (Taulukossa 1 askel 1 osoittaa tämän toimenpiteen). Sitten ympyröityjen B:den kohdalla rullaus lisää ensin X:t (askel 2) ja poistaa sitten ympyröidyt B:t (askel 3). Seuraavaksi ympyröityjen A-kirjaimien kohdalla rullaus lisää ensin B:t (askel 4) ja poistaa sitten ympyröidyt A:t (askel 5). Sitten matriisissa esiintyvien X:ien kohdalle lisätään ensin A:t (askel 6) ja sen jälkeen X:t poistetaan (askel 7).

Seuraava taulukko 2 ja siihen liittyvä teksti kuvaavat askeleet joilla kytkentä tehdään, jotta saataisiin aikaan väliportaan kytkimien uudelleenjärjestely, kun $X \rightarrow B$ johtaa minimimäärään kytkentöjen uudelleenjärjestelyjä:

TAULUKKO 2	
ASKEL	TOIMENPIDE
1	$X-(r_1, c_1)$
2	$\textcircled{A} \rightarrow \textcircled{A}, X$
3	$\textcircled{A}, X-X$
4	$\textcircled{B} \rightarrow \textcircled{B}, A$
5	$\textcircled{B}, A-A$
6	$X-X, B$
7	$X, B-B$

$X \rightarrow B$ -rullaus tapahtuu samalla tavalla kuin $X \rightarrow A$ -rullaus. Esimerkiksi ensimmäinen askel on laittaa X positioon

(r_1, c_1), mikä osoittaa, että kytkentä tehtiin ylimääräisen väliportaan kautta (Taulukko 2, askel 1). Ympyröityjen A-kirjaimien osalta rullaus lisää ensin X:t (askel 2) ja poistaa sitten ympyröidyt A:t (askel 3). Seuraavaksi ympyröityjen B-kirjaimien osalta rullaus lisää A:t (askel 4) ja poistaa sitten ympyröidyt B:t (askel 5). Sen jälkeen X:ien osalta rullaus lisää B:t (askel 6) ja poistaa lopuksi X:t (askel 7).

Uudelleenjärjestely suoritetaan tarpeen mukaan käyttämällä "etupään siltaa" ja "vastaanottopään kytkintä" osumattoman rullauksen suorittamiseksi (sisältäen alkuperäisen kytkennän joka suoritettiin nimetyssä tai ylimääräisessä elementissä) matriisin numeroltaan alemmille elementeille. Termit "etupään silta" ja "vastaanottopään kytkin" määritellään ja selitetään havainnollisesti alla kuvioiden 15 - 34 esimerkkiin liittyen. Nämä askeleet jättävät ylimääräisen elementin vapaaksi ja valmiiksi seuraavaa välitöntä kytkentää varten uudelleenjärjesteltävästi estetyssä tilassa.

Vaikka uskotaan, että esillä oleva määrittely ja piirroksot kuvaavat järjestelmän täydellisesti ja tarkasti ja antavat riittävästi informaatiota siten, että kuka tahansa alan ammattimies voi soveltaa keksinnöllistä periaatetta, "julkaisematon liite A" on sijoitettu mukaan jädäkseen painamatta. Julkaisematon liite A sisältää liskauksen toimivasta lähdekoodista, jotta tietokone voisi voisi toteuttaa ja suorittaa esillä olevan menetelmän ja järjestelmän mukaisen välittömän kytkennän menetelmän.

Toiminta

Selitettyämme yleisesti edullisen toteutuksen mukaisen prosessin, seuraava havainnollistava esimerkki esittää kuinka edullinen toteutus suorittaa välittömän kytkennän uudelleenjärjesteltävästi estetyn tilan esiintyessä tila-alueella. Aika-alueella menetelmä tulee sel-

väksi tila-alueen menetelmän toteutustavan ymmärtämisen jälkeen.

Kuviot 6 - 14 kuvaavat esimerkkiä edullisen toteutuksen mukaisesta välittömän kytkennän menetelmästä. Kuvion 6 neliömatriisissa solu (1,1) on merkitty estetyksi, koska esillä oleva kytkentäalgoritmi havaitsi, että nimetty väliportaan kytkin on ainut väliportaan kytkin, jolla on sekä vapaa yhteys tulokytkimeen 1 että vapaa yhteys lähtökytkimeen 1.

Tämän tilanteen seurauksena kytkentäalgoritmi laittaa X:n kohtaan (r_1, c_1) kytkennän suorittamiseksi, kuten kuvio 6 esittää. Merkin "X" käyttö tässä merkitsee nimetyn tai ylimääräisen väliportaan kytkimen käyttöä uudelleenjärjesteltävästi estetyssä tilassa. Kun kytkentäalgoritmit käyttävät suuren matriisikonfiguraation nimettyä väliportaan kytkintä, uudelleenjärjesteltävä estotilanne havaitaan automaattisesti.

Koska kytkentäpari (1,1) on uudelleenjärjesteltävästi estetty, tulee olla olemassa vapaa yhteys tulokytkimen 1 ja jonkin väliportaan kytkimen, esimerkiksi A, välillä jotta kytkentä olisi mahdollinen. Lisäksi tulee olla olemassa vapaa yhteys lähtökytkimen 1 ja jonkin väliportaan kytkimen, esimerkiksi B, välillä. Pari (A,B) on yksiväliportaan kytkimien pari, jota harkitaan uudelleenjärjestelyä varten. Tässä esimerkissä pari (A,B) on ainut väliportaan kytkimien pari, joka täyttää yllä mainitun kriteerin. Koska pari (A,B) muodostaa etsittävien parien koko joukon, (A,B) on pari joka vaatii vähiten uudelleenjärjestelyjä.

Kuvio 7 esittää edullisen toteutuksen mukaisen välittömän kytkentäalgoritmin $X \rightarrow A$ -sekvenssin ja kuvio 8 esittää $X \rightarrow B$ -sekvenssin. Yhdessä kuviot 7 ja 8 esittävät edullisen toteutuksen mukaisen ylivuotoalgoritmin toiminnan. Ensimmäinen askel esimerkiksi alkaa kuviossa 7 riviltä r_1 ja löytää A:n kohdasta (r_1, c_2) , joka on ympyröity.

Kuvio 8 esittää ylivuotoalgoritmin, joka käyttää $X \rightarrow B$ -sekvenssiä ja alkaa sarakkeesta c_1 löytääkseen ympyröidyn $B:n$ kohdassa (r_3, c_1) , toista askelta. Sen jälkeen kolmas askel suoritetaan jälleen kuviossa 7, jossa saraketta c_2 tutkitaan $B:n$ löytämiseksi. Tämä löydetään positioista (r_2, c_2) ja ympyröidään. Koska rivillä r_2 ei ole yhtään $A:ta$, tämä $X \rightarrow A$ -sekvenssi on päättynyt. Kuviossa 7 olevat kaksi symbolia osoittavat, että vaaditaan kaksi uudelleenjärjestelyä.

10 Vaikka esillä olevan menetelmän mukainen ylivuotoalgoritmi päättyy kolmannessa askeleessa, mikäli se jatkuisi, kuviossa 8 tutkittaisiin rivi r_3 $A:n$ löytämiseksi, joka löydetäisiin positioista (r_3, c_3) . Tämä proseduuri jatkuisi ympyröiden vuorotellen B -kirjaimia sarakkeissa ja
15 A -kirjaimia riveillä, kunnes löytyisi joko sarake ilman ympyröitävää $B:tä$ tai rivi ilman ympyröitävää $A:ta$. Kuviossa 8 olevat neljä ympyröityä symbolia sen vuoksi osoittavat, että neljä uudelleenjärjestelyä on tarpeen esimerkin tapauksessa.

20 Koska $X \rightarrow A$ -sekvenssi vaatii vain kaksi uudelleenjärjestelyä ja $X \rightarrow B$ -sekvenssi vaatii neljä uudelleenjärjestelyä, edullinen toteutus valitsee $X \rightarrow A$ -sekvenssin parille (A, B) . Koska (A, B) on ainoa harkittava väliportaan kytkimien pari, $X \rightarrow A$ -sekvenssi tuottaa minimimäärän uudelleenjärjestelyjä.

25 Seuraavana askeleena on suorittaa uudelleenjärjestely käyttämällä aiemmin mainittua "törmäyksetöntä rullausta". Koska $X \rightarrow A$ -sekvenssi tuottaa minimimäärän uudelleenjärjestelyjä, edullinen toteutus käyttää $X \rightarrow A$ -rullausta
30 tässä esimerkissä. Törmäyksetön rullaus sisältää alkupeiräisen kytkennän, joka sijaitsee nimetyssä väliportaan kytkimessä. Tämä rullaus tapahtuu suorittamalla ensin tarvittava uudelleenjärjestely käyttämällä etupään siltaa ja vastaanottopään kytkintä.

Kuviot 9 - 14 kuvaavat X→A törmäyksettömän rullauksen suorittamista. Menetelmä muuttaa aiemmin ympyröityjen symbolien A ja B sekvenssiä estetyn solun vapauttamiseksi seuraavasti:

5 Ympyröityjen B-kirjaimien kohdalle lisätään X:t kuten kuvio 9 esittää. Tässä esimerkissä tämä askel vastaa etupään sillan muodostamista väliportaan kytkimen läpi ylimääräisessä kytkimessä X olemassa olevan kytkennän osalta, joka on tulokytimestä 2 väliportaan kytkimen B
10 läpi, ja vastaanottopään kytkimen muodostamista välikytkimen X kautta olemassa olevan kytkennän osalta ja väliportaasta B lähtöportaaseen.

Seuraavana askeleena on poistaa ympyröidyt B:t kuten kuvio 10 esittää. Tässä esimerkissä, kun on olemassa
15 kytkentä väliportaan kytkimen X läpi ja lähtökytkin 2 vastaanottaa hyvän signaalin, seuraavana askeleena on ottaa pois käytöstä kytkentä tulokytimestä 2 väliportaan kytkimen B läpi lähtökytkimeen 2.

Sitten, kuten kuvio 11 esittää, ympyröidyille
20 A-kirjaimille lisätään B:t. Tässä esimerkissä tämä askel vastaa etupään sillan muodostamista väliportaan kytkimen B läpi olemassa olevalle kytkennälle tulokytimestä 1 väliportaan kytkimen A läpi, ja vastaanottopään kytkimen muodostamista väliportaan kytkimen B läpi olemassa olevalle
25 kytkennälle väliportaan kytkimestä A lähtöportaalle kytkimeen 2. Sen jälkeen, kuten kuvio 12 esittää, ympyröidyt A:t poistetaan. Tässä esimerkissä, kun on olemassa kytkentä väliportaan kytkimen B läpi ja lähtökytkin 2 vastaanottaa hyvän signaalin, seuraavana askeleena on ottaa pois
30 käytöstä kytkentä tulokytimestä 2 väliportaan kytkimen A kautta lähtökytkimeen 2.

Sitten seuraavana askeleena on lisätä A:t positioihin jotka sisältävät X:t, kuten kuvio 13 esittää. Tässä
35 esimerkissä tämä vastaa etupään sillan muodostamista väliportaan kytkimen A läpi sekä kytkennälle tulokytimestä 2

välikytkimen X läpi lähtökytkimeen 2 että välittömälle kytkennälle tulokytimestä 1 välikytkimen X läpi lähtökytkimeen 1.

Viimeisenä askeleena on poistaa X:t, kuten kuvio 14
 5 kuvaa. Tässä esimerkissä, kun kukin kytkentä väliportaan kytkimen A läpi on olemassa ja vastaava lähtökytkin vastaanottaa hyvän signaalin, vastaava kytkentä tulokytimestä välikytkimen X läpi vastaavaan lähtökytkimeen voidaan ottaa pois käytöstä. Huomaa, että ylimääräinen väliportaan
 10 kytkin X jää vapaaksi, koska kaikki kytkennät puretaan ylimääräisestä väliportaan kytkimestä X kyseessä olevaan väliportaan kytkimeen.

Kuvattuamme yksityiskohtaisesti edullisen menetelmän ja järjestelmän toiminnan, seuraava esitys kuvaa edullisen toteutuksen mukaisen kytkentäpolun valintamenetelmän ja järjestelmän. Esillä olevan keksinnön tämän näkökohdan kuvaamisen yksinkertaistamiseksi esimerkki tarkastelee
 15 kytkentöjä kuvion 15 edelleen pienennetyssä $N(5,4,4)$ -tilamatriisissa 220.

Kuvio 15 kuvaa yksinkertaistetun $N(5,4,4)$ -tilamatriisin 220 tilaa alustuksen jälkeen. Huomaa, että viides väliportaan kytkin 224 on merkitty X:llä, jotta osoitettaisiin että se on nimetty tai ylimääräinen väliportaan kytkin. Kuten selitys 226 esittää, lyhyeksi katkotut vii-
 20 vat 228 osoittavat vapaan yhteyden, jatkuva viiva 230 osoittaa varatun yhteyden ja pitkäksi katkottu viiva 232 osoittaa kytkimen kytkennän. Koska yhtään tuloa ei ole kytketty yhteenkään lähtöön kuviossa 15, ei ole olemassa mitään tuloportaan, väliportaan tai lähtöportaan kytkimien
 25 kytkentöjä, ja kaikki yhteydet kytkimien välillä ovat vapaana.

Yksinkertaistettu $N(5,4,4)$ -tilamatriisi mielessä pitäen on mahdollista kertoa yksityiskohtaisesti kuinka toteuttaa tietokoneohjelmisto, joka suorittaa kytkentäpolun valinnan ja esillä olevan keksinnön muut tavoitteet.
 35

Huomaa kuitenkin, että tilamatriisin 220 koko ei rajoita ohjelmistototeutuksen käytettävyyttä tai soveltuvuutta. Ohjelmistototeutuksen ymmärtämisen helpottamiseksi seuraavassa pohdinnassa esitellään datamäärittelyt ja merkinnät, jotka ovat tarpeellisia kuvion 16 matriisien 222, 234, 236, 238 ja 240 käsittelyssä. Huomaa myöskin, että kytkentöjen ymmärtämisen helpottamiseksi kuviossa 15 käytetään samoja viitenumeroita tuloporteille 90, tuloportaan kytkimille 92 jne., jotka esiintyvät kuvion 3 yleistetyssä $N(m,n,r)$ -matriisissa. Seuraavat datamäärittelyt ovat hyödyllisiä kuvattaessa esillä olevaa kytkentämenetelmää ja järjestelmää.

Ohjelmistototeutuksessa tapahtuu tulo-/lähtöportin, tai linjojen, konversio. Kytkentä voidaan esimerkiksi pyytää muodossa $IL-m \rightarrow OL-n$. Toisin sanoen tulolinja m lähtölinjaan n . Tulolinjan numero konvertoidaan tulokytkinportaan numeroksi i ja tuon tulokytkinportaan j tulonumeroksi siten, että $IL-m \rightarrow I(i,j)$. Samalla tavoin lähtölinjan numero konvertoidaan lähtökytkinportaan numeroksi k ja tuon lähtökytkinportaan l numeroksi siten, että $OL-n \rightarrow O(k,l)$.

Formaalimmin $I(i,j)$ määritetään $IL-m$:stä ja $O(k,l)$ määritetään $OL-n$:stä seuraavasti:

$$\begin{aligned} i &= 1 + \text{Quot}_N(m-1) \\ j &= 1 + \text{Rem}_N(m-1) \\ k &= 1 + \text{Quot}_N(n-1) \\ l &= 1 + \text{Rem}_N(n-1) \end{aligned}$$

jossa

N on ensimmäisen (tulo-) portaan kytkimien määrä, joka on yhtä suuri kuin kolmannen (lähtö-) portaan kytkimien määrä;

Quot_N on kokonaislukuosamäärä argumentista joka on jaettu N :llä; ja

Rem_N on kokonaislukujakojäännös sen jälkeen kun argumentti on jaettu N :llä.

Seuraavat lausekkeet ja datamäärittelyt voidaan ymmärtää tarkastelemalla kuvioita 15 ja 16 tietokoneen generoimien matriisien muodostamiseksi esillä olevan keksinnön tavoitteiden toteuttamiseksi.

5

Datamäärittely: `space_in_mat(in_stage,in_num)`

Kuviossa 16 tilatulomatriisi 222 kuvion 15 N(5,4,4)-esimerkkiä varten on 4x4-matriisi, joka edustaa tuloportaan kytkimien kytkentää SSS:ssä, sellaisessa kuin kytkin 92. Tilatulomatriisin 222 kukin rivi vastaa yhtä tuloportaan kytkimistä 92. Tilatulomatriisin 222 sarakkeet vastaavat tuloportteja 90 kuhunkin ensimmäisen portaan kytkimistä. Tilatulomatriisin arvot vastaavat ensimmäisen portaan kytkimien lähtöjä, sellaisia kuin lähdöt 94. Koska kunkin tuloportaan kytkimen lähtö on kytketty numeroltaan samaan väliportaan kytkimeen, tilatulomatriisin arvoja voidaan pitää väliportaan kytkimenä, johon tuo määrätty tuloportaan kytkin (rivi) ja tulo (sarake) on kytketty.

20

Tämän seurauksena lauseke

Lauseke: `space_in_mat(i,j) = c`

aiheuttaa, että tilatulomatriisi 222 ilmaisee kytkennän i:n ensimmäisen portaan kytkimen j:stä tulosta väliportaan kytkimeen c.

Tilatulomatriisin 222 arvot vaihtelevat nollasta X:ään käytettäessä kuvion 15 yksinkertaistettua esimerkkiä. Nolla osoittaa ettei mitään kytkentää ole asetettu. Arvot 1 - 4 ovat normaaleja kytkentöjä. Matriisin 222 numeerinen arvo 5 vastaa tässä esimerkissä kuvion 6 arvoa "X" ja se osoittaa, että on asetettu ylimääräinen väliportaan kytkin 224 ja että uudelleenjärjestelyprosessia on kutsuttu.

35

Datamäärittely: `space_cnt_mat[cnt_stage,in_stage]`

Tilavälimatriisi 234 on 5x4-matriisi, joka edustaa välilytkimien, sellaista kuin väliportaan kytkimet 96, kytkeäkarttaa tila-alueella. Tilavälimatriisin kukin rivi vastaa yhtä väliportaan kytkimestä 96. Tilavälimatriisin sarakkeet vastaavat tuloja kuhunkin väliportaan kytkimeen, sellaisia kuin tulot 94. Tilavälimatriisin arvot vastaavat väliportaan kytkimien lähtöjä, sellaisia kuin lähtö 98. Koska kunkin väliportaan kytkimen 96 lähtö kytkeytyy lähtöportaan kytkimeen samalla tavalla, tilavälimatriisin 234 arvoja voidaan pitää lähtöportaan kytkimenä, johon tuo määrätty väliportaan kytkin (rivi) ja tulosarake kytkeytyy.

Tämän seurauksen lauseke

Lauseke: `space_cnt_mat(c,j) = k`

aiheuttaa, että tilavälimatriisi 234 ilmaisee kytkennän väliportaan kytkimen "c" j:nestä tulosta k:nteen lähtöportaan kytkimeen. Tilavälimatriisin 234 arvot vaihtelevat välillä 0 - 4. Nolla ilmaisee, että mitään kytkentää ei ole asetettu. Arvot 1 - 4 ovat normaaleja kytkentöjä. Nolla X:nellä rivillä (so. 5. rivi tässä esimerkissä) ilmaisee, että on asetettu ylimääräinen väliportaan kytkin 224 ja että on kutsuttu uudelleenjärjestelyprosessia.

Datamäärittely: `space_out_mat[out_stage,out_num]`

Tilalähtömatrissi 236 on 4x4-matriisi, joka edustaa tilamatriisin lähtöportaan kytkentäkarttaa tila-alueella. Tilalähtömatrissin 236 kukin rivi vastaa yhtä lähtökytkimestä, sellaista kuin lähtökytkin 100. Tilalähtömatrissin 236 sarakkeet vastaavat lähtöä kustakin lähtökytkimestä, sellaisesta kuin lähtöportit 102. Tilalähtömatrissin 236

arvot vastaavat tuloa johon lähtö kytkeytyy, sellaisia kuin kytkennät 98. Koska kunkin lähtökytkimen tulo kytkeytyy saman numeroiseen väliportaan kytkimeen, tilalähtömatriisiin 236 arvoja voidaan pitää väliportaan kytkimenä johon määrätty lähtökytkin 100 (rivi) ja lähtöportti 102 (sarake) kytkeytyy.

Tämän seurauksena lauseke

Lauseke: $\text{space_out_mat}(k,l) = c$

10

aiheuttaa, että tilalähtömatriisi 236 ilmaisee kytkennän väliportaan kytkimestä "c" k:nnen lähtöportaan kytkimen l:nteen lähtöön. Tilalähtömatriisiin 236 arvot vaihtelevat välillä 0 - X. Nolla ilmaisee, että mitään kytkentää ei ole asetettu. Arvo X (so. 5 tässä esimerkissä) ilmaisee, että ylimääräinen väliportaan kytkin 224 on asetettu ja että uudelleenjärjestelyprosessia on kutsuttu.

Jatkamalla kuvioiden 15 ja 16 tarkastelua seuraava käsittely kertoo yksityiskohdat siitä, kuinka toteuttaa edullinen toteutus tuloportin ja lähtöportin välisen vapaan yhteyden esittämiseksi. Normaalin toiminnan aikana esillä oleva kytkentämenetelmä yrittää suorittaa määrätyn kytkennän numeroltaan pienimmän väliportaan kytkimen läpi. Kytkennän suorittamiseksi määrätystä tuloportaan kytkimestä, sellaisesta kuin tulokytkin 92, määrättyyn lähtöportaan kytkimeen, sellaiseen kuin lähtökytkin 100, tulee olla olemassa tietty väliportaan kytkin, sellainen kuin välikytkin 96, jolla on ominaisuutena, että siinä on vapaa tuloyhteys tulokytkimestä 92 välikytkimeen 96 ja vapaa lähtöyhteys välikytkimestä 96 lähtökytkimeen 100.

Esillä oleva menetelmä ja järjestelmä jäljittävät vapaita tuloyhteyksiä ja vapaita lähtöyhteyksiä käyttämällä kahta erilaista bittikarttataulua (yksi tuloyhteyksille ja yksi lähtöyhteyksille). Kummassakin bittitaulukartassa on erillinen rivi tilavälimatriisien kullekin riville.

Menetelmä käyttää primitiivien ryhmää bittikarttojen käsitteilyyn kytkentä- ja purkuoperaatioiden suorittamiseksi. Primitiivit puolestaan kutsuvat uudelleenjärjestelyprosesseja tarvittaessa.

5

Datamäärittely: space_in_link[in_stage,cnt_stage]

Kuvioiden 15 ja 16 esimerkin tapauksessa tilatulon vapaan yhteyden bittikartta 238 on 4x5 bitin taulu, joka edustaa vapaita tuloyhteyksiä 94 tulokytkimien 92 ja välikytkimien 96 välillä tila-alueella. Tilatulon vapaan yhteyden bittikartan 238 kukin tulo vastaa yhtä tuloportaan kytkimistä 92. Tilatulon vapaan yhteyden bittikartan 238 arvot määrätyle riville (tuloporras) ja sarakkeelle (väliporras) osoittavat tuloyhteyden 94 vapaan tilan tuloportaan ja väliportaan välillä. Arvo 1 ilmaisee, että tuloyhteys on vapaa ja käytettävissä. Arvo nolla ilmaisee, että tuloyhteys on käytössä eikä se ole asetettavissa käyttöön. Arvo nolla sarakkeessa X ilmaisee, että ylimääräinen väliportaan kytkin 234 on käytössä ja uudelleenjärjestelymenetelmä on käynnissä.

Datamäärittely: space_out_link[out_stage,cnt_stage]

Tilalähdön vapaan yhteyden bittikartta 240 on 4x5 bitin taulu, joka edustaa vapaita lähtöyhteyksiä lähdön ja välikytkimien 96 välillä tila-alueella. Tilalähdön vapaan yhteyden bittikartan 240 kukin rivi vastaa yhtä lähtöportaan kytkimistä 100. Arvot tilalähdön vapaan yhteyden bittikartassa 240 määrätyle riville (lähtökytkin) ja sarakkeelle (välikytkin) ilmaisevat lähtöyhteyden 98 vapaan tilan tuon lähtökytkimen ja välikytkimen välillä. Arvo 1 ilmaisee, että lähtöyhteys on vapaana ja käytettävissä. Arvo nolla sarakkeessa X ilmaisee, että ylimääräinen väli-

portaan kytkin 224 on käytössä ja uudelleenjärjestelymenetelmä on käynnissä.

Kuvattuamme kartat vapaiden yhteyksien esittämiseksi, seuraavassa kuvataan primitiivioperaatiot, jotka on
5 hyödyllisiä tilassa olevan vapaan yhteyden bittikarttojen
238 ja 240 manipuloinniseksi kytkentä- ja purkuoperaatioita
varten.

```

10      Primitivi:    init_links()

```

Vapaan yhteyden bittikartat alustava primitiivi (käytetään vain käynnistysoperaatioissa) luo sekä tilatun vapaan yhteyden bittikartan 238 että tilalähdön vapaan yhteyden bittikartan 240 edullista toteutusta varten.

15 Kaikki arvot vapaan yhteyden bittikartoissa alustetaan arvoon 1, mikä ilmaisee, että kaikki yhteydet ovat vapaina.

```

Primitiivi:    set_link(i/o, j, k)
20

```

Tämä primitiivi asettaa "i/o"-parametrin määrittelemän bittikartan j:n:n rivin k:n:n bitin arvoon 1. Jos bittikartta sisältää n elementtiä ja j:n arvo ylittää n:n arvon, tietokone palauttaa virhelipun. Muussa tapauksessa k:s elementti asetetaan arvoon 1.

```
Primitiivi: clear link(i/o, j, k)
```

30 Tämä primitiivi tyhjentää (nollaa) i/o-parametrin
määrittelemän bittikartan j:n:n rivin k:n:n bitin arvoon
0. Jos bittikartta sisältää n elementtiä ja j:n arvo ylit-
tää n:n arvon, tietokone palauttaa virhelipun. Muussa ta-
pauksessa k:s elementti asetetaan arvoon 0.

```

35 Primitiivi: pump_link(space_in_link[i],space_out_link
    [k])

```

Edullinen menetelmä ja järjestelmä tekee kytkennän tulokytkinportaan i ja lähtökytkinportaan k välille käyttämällä numeroltaan pienintä saatavilla olevaa väliportaan kytkintä. Tämä toteutetaan prosessilla, jota tämän jälkeen

5 kutsutaan `space_n_link[i]:n` ja `space_out_link[k]:n` "pump-paukseksi". Tämä tarkoittaa, että `space_in_link[i]:lle` suoritetaan looginen "AND" `space_out_link[k]:n` kanssa `c:n`, numeroltaan pienimmän väliportaan kytkimen, jolla on sekä vapaa yhteys tuloportaaseen ja vapaa yhteys lähtöportaaseen

10 `k`, määrittämiseksi. Loogisen AND:in tuloksessa olevan ensimmäisen nollasta poikkeavan arvon indeksi on `c`, numeroltaan pienin väliportaan kytkin, jolla on sekä vapaa yhteys tuloportaaseen i että vapaa yhteys lähtöportaaseen k . Mainitut kaksi yhteyttä `c:en` merkitään sitten "varatuiksi" `space_in_link[i]:ssä` ja `space_out_link[k]:ssa` ja palautetaan väliportaan numero `c`. Taulukko 3 kuvaa tämän primitiivimakron toteutuksen.

15

TAULUKKO 3	
PUMPPULISTAMAKRON VUOKAAVIO	
5	<pre> pump_link(space_in_link[1], space_out_link[k] int c, i, k, x[xtra_stage]; { int j; x = space_in_link[i] && space_out_link[k]; 10 for (j = 1; j <= xtra_stage; j++) { if (X(j) == 1) { c = j; 15 break; } } clear_link(space_in_line, i, c); clear_link(space_out_link, k, c); 20 if (c == xtra_stage) { rearrange = 1; } return(c); 25 { </pre>

Kuvio 16 kuvaa toteutusdatamatriisit alustamisen jälkeen. Vastaavat arvot esimerkiksi kytkentäpolun valintamatriiseissa space_in_mat 222, space_cnt_mat 234 ja space_out_mat 236 on kuvattu kuviossa 16. Koska mitään kytkinyhteyksiä ei ole, kaikki arvot ovat nolliä. Bittikarttojen space_in_link 238 ja space_out_link 240 sisällöt ovat kuten alustuksessa on esitetty. Huomaa, että kussakin tuloportaan kytkimessä ja kussakin lähtöportaan kytkimessä

on elementti 1 jokaista vapaata väliportaan kytkintä kohden. Tässä esimerkissä alustuksen jälkeen kaikki viisi väliportaan kytkintä on merkitty "vapaiksi".

5 Kuvio 17 esittää toteutusdatan ensimmäisen kytkennän jälkeen. Oletettakoon, että ensimmäinen pyydetty kytkentä on kytkeä tulolinja 1 lähtölinjaan 1, tai yksinkertaisemmin IL-1→OL-1. Ensimmäisenä askeleena on muuntaa tulolinjan numero tulokytkinportaan numeroksi ja tulonumeroksi tuota tulokytkinporrasta varten. Tässä tapauksessa
10 IL-1→I(1,1). Samalla tavoin lähtölinjan numero muunnetaan lähtökytkinportaan numeroksi ja lähtönumeroksi tuota lähtökytkinporrasta varten. Tässä tapauksessa OL-1→O(1,1). Siten aiomme tehdä kytkennän tuloportaan 1 ja lähtöportaan 1 välille.

15 Esillä oleva menetelmä ja järjestelmä pyrkivät tekemään kytkennän tuloportaan 1 ja lähtöportaan 1 välillä käyttäen numeroltaan alhaisinta käytettävissä olevaa väliportaan kytkintä, joka tässä tapauksessa on väliporras 1. Tämä suoritetaan käyttämällä taulukon 3 pump_list-makroa
20 space_in_link[1]:n vertaamiseksi space_out_link[1]:n kanssa kuviossa 16 ja asettamalla molempien bittikarttojen kanssa yhteisen alhaisimman väliportaan numeron vapaat kytkennät arvoon "0". Sitten space_in_mat(i,j) ja space_out_mat(k,l) asetetaan samaksi kuin tuo väliportaan
25 numero, ja space_cnt_mat(c,j) asetetaan samaksi kuin k. Vastaavat tilamatriisin data-arvot on esitetty kuviossa 17 ja siitä seuraava tilamatriisin 220 tila on esitetty kuviossa 18.

Seuraavat lausekkeet suorittavat tämän prosessin:

	TAULUKKO 4
	KYTKENTÄLAUSEKKEET
5	<pre> x = pump_list(space_in_link[i], space_out_link[k]); space_in_mat(i,j) = c; space_cnt_mat(c,i) = k; space_out_mat(k,l) = c; jossa tässä tapauksessa i=j=k=l=m=c-1. </pre>
10	

Oletetaan, että kytkentöjä summaava proseduuri jatkuu kunnes n^2-2 kytkentää on tehty (oletettavasti ilman uudelleenjärjestelyä) lopputuloksen olevan tilamatriisin ollessa esitetty kuviossa 19 ja kytkentätien valintamatriisin datan ollessa esitetty kuviossa 20.

Sitten vastaanotetaan pyyntö kytkeä IL-13 OL-7:ään. Pyyntö kytkeä IL-13 OL-7:ään konvertoituu pyynnöksi kytkeä I(4,1)→O(2,3). Esillä oleva menetelmä ja järjestelmä pyrkivät suorittamaan kytkennän tuloportaan 4 ja lähtöportaan 2 välillä käyttämällä numeroltaan alhaisinta käytettävissä olevaa väliportaan kytkintä, joka tässä tapauksessa on väliportaan kytkin numero 4, kuten kuviossa 20 on kuvattu. Taulukon 1 pump_list-makro vertaa space_in_link[4]:ää space_out_link[2]:een, määrittää molemmille bittikartoille yhteisen alhaisimman väliportaan numeron ja asettaa tilatiedon vapaa tiedoksi "varattu" molemmissa bittikartoissa. Sitten space_in_mat(i,j) ja space_out_mat(k,l) asetetaan samaksi kuin tuo väliportaan numero ja space_cnt_mat(c,j) asetetaan yhtä suureksi kuin k.

Vastaavat tilamatriisin data-arvot on esitetty kuviossa 21 ja siitä seuraava tilamatriisin tila on esitetty kuviossa 22. Tämä prosessi suoritetaan käyttämällä seuraavia lausekkeita kytkentärutiinista sellaisina kuin aiemminkin:

TAULUKKO 5

KYTKENTÄLAUSEKKEET

```

x = pump_list(space_in_link[i], space_out_link[k]);
space_in_mat(i,j) = c;
space_cnt_mat(c,i) = k;
space_out_mat(k,l) = c;

```

jossa tässä tapauksessa $i=4$ $j=1$ $k=2$ $l=3$ $m=13$ $n = 7$ ja $c=4$.

10 Oletetaan, että seuraava pyyntö on purkaa IL-13 OL-7:stä (mikä poistaa kytkennän joka juuri tehtiin). Pyyntö purkaa IL-13 OL-7:stä konvertoituu pyynnöksi purkaa $I(4,1) \rightarrow O(2,3)$. Put_back-makro suorittaa purkamisen katkaisemalla yhteyden tuloportaan 4 ja lähtöportaan 2 välillä ja laittamalla vapautetun väliportaan kytkimen, numeroltaan 4, takaisin tulo-/lähtöpinoihin myöhempää käyttöä varten. Tarkemmin ottaen put_back-makro määrittää vapautetun väliportaan kytkimen numeron ja laittaa sen takaisin oikeisiin kohtiin molemmissa järjestetysti ketjutetuista listoista space_in_link[4] ja space_out_link[2]. Sitten space_in_mat(i,j) ja space_cnt_mat(c,j) ja space_out_mat(k,l) asetetaan yhtä suuriksi kuin 0.

25 Vastaavat tilamatriisin data-arvot on esitetty kuviossa 23 ja siitä aiheutuva tilamatriisin tila on sama kuin kuviossa 18 on aiemmin esitetty. Purkaminen suoritetaan käyttämällä seuraavia lausekkeita put_back-makrosta:

TAULUKKO 6	
PURKULAUSEKKEET	
5	<pre> c = space_in_mat (i,j); set_link(space_in_link, i, c); set_link(space_out_link, k, c); space_in_mat(i,j) = 0; space_cnt_mat(c,i) = 0; 10 space_out_mat(k,l) = 0; </pre>
	jossa tässä tapauksessa i=4 j=1 k=2 l=3 m=13, n=7, ja c=4.

Seuraavaksi oletetaan, että vastaanotamme pyynnön kytkeä IL-13 OL-12:een. Pyyntö kytkeä IL-13 OL-12:een kon-
 vertoituu pyynnöksi kytkeä I(4,1)→O(3,4). Esillä oleva
 menetelmä ja järjestelmä pyrkii suorittamaan kytkennän
 tuloportaan 4 ja lähtöportaan 3 välillä käyttämällä nume-
 roltaan alhaisinta käytettävissä olevaa väliportaan kyt-
 kintä. Tässä tapauksessa X:s, tai ylimääräinen väliportaan
 kytkin 224, on numeroltaan alhaisin käytettävissä oleva
 väliportaan kytkin, kuten kuviossa 25 on kuvattu.

Kuten aiemminkin, pump_list-makro suorittaa tämän
 vertaamalla space_in_stack[4]:ää space_out_stack[3]:een
 kuviossa 23 määrittämällä alhaisimman väliportaan numeron,
 joka on yhteinen molemmille bittikartoille (joka tässä
 tapauksessa on väliportaan kytkin "X") ja asettamalla ti-
 latiedon vapaa tilatiedoksi "varattu" molemmissa bittikar-
 toissa. Sitten space_n_mat(i,j) ja space_out_mat(k,l)
 asetetaan samaksi kuin tuo väliportaan numero ja
 space_cnt_mat(c,j) asetetaan yhtä suureksi kuin k. Arvo 5
 ilmaisee, että X:s väliportaan kytkin on asetettu ja uu-
 delleenjärjestelyprosessia kutsutaan automaattisesti. Vas-
 taavat tilamatriisin data-arvot uudelleenjärjestelyn aika-

na on esitetty kuviossa 24 ja siitä aiheutuva tilamatriisin tila järjestelyn aikana on esitetty kuviossa 25.

Seuraavat välittömän kytkennän lausekkeet suoritavat tämän:

5

10

15

TAULUKKO 7	
VÄLITTÖMÄN KYTKENNÄN LAUSEKKEET	
	<pre> x = pump_list(space_in_stack[i],space_out_stack[k]); space_in_mat(i,j) = c; space_cnt_mat(c,i) = k; space_out_mat(k,l) = c; jossa tässä tapauksessa i=4 j=1 k=3 l=4 m=13, n=12, ja c=5.</pre>

Tuloksena uudelleenjärjestelyalgoritmista kytkentä $I(4,3) \rightarrow O(2,1)$, joka näkyy kuviossa 26, "rullataan" väliportaan kytkimestä numero 2 väliportaan kytkimeen numero 4. Tämä käärintä suoritetaan ilman törmäyksiä luomalla etupään silta tuloportaan kytkimeen 4 ja vastaanottopään kytkin lähtöportaan kytkimeen 2, kuten kuviossa 27 on esitetty. Kun oikea polku otetaan vastaan lähtöportaan kytkimeen 2 väliportaan kytkimestä 4, alkuperäinen kytkentä väliportaan kytkimen 2 kautta irroitetaan, kuten kuviossa 28 on esitetty, ja tilamatriisin data päivitetään, kuten kuviossa 29 on esitetty.

Aiempi rullaus sallii nyt välittömän kytkennän $I(4,1) \rightarrow O(3,4)$, kuten kuvio 31 esittää, joka on väliportaan kytkimessä X 224, rullaamisen väliportaan kytkimeen 2.

Tämä rullaus suoritetaan ilman törmäyksiä luomalla etupään silta tuloportaan kytkimeen 4 ja vastaanottopään kytkin lähtöportaan kytkimeen 3, kuten kuviossa 31 on esitetty. Kun oikea polku otetaan vastaan lähtöportaan kyt-

kimeen 3 väliportaan kytkimestä 2, alkuperäinen kytkentä väliportaan X 224:n kautta irrotetaan.

Välittömän kytkennän $I(4,1) \rightarrow O(3,4)$ rullaus pois ylimääräisestä väliportaan kytkimestä päättää tämän kytkennän uudelleenjärjestelyprosessin. Huomaa, että tämä prosessi jättää ylimääräisen väliportaan kytkimen vapaaksi ja valmiiksi seuraava kytkentää varten, kun tapahtuu uudelleenjärjesteltävä estotilanne. Kuvio 32 esittää lopputuloksen aoleva tilamatriisin tilan uudelleenjärjestelyn jälkeen. Kuviossa 33 esitetään vastaavat tilamatriisin data-arvot uudelleenjärjestelyn jälkeen.

Kuviossa 24 edullisen toteutuksen uudelleenjärjestelynäkökohtia sovelletaan yllä olevaa esimerkkiin. Kun kerran edullinen toteutus käyttää nimettyä väliportaan kytkintä 224 välittömän kytkennän suorittamiseen, edullisen toteutuksen mukainen järjestelmä kutsuu uudelleenjärjestelyprosessia automaattisesti. Ensiksi järjestelmä määrittää kaksi väliportaan kytkintä, jotka tulevat osallistumaan uudelleenjärjestelyprosessiin. Tämä suoritetaan käyttämällä Paullin menetelmää, joka esitetään uudelleen seuraavasti:

Jos (r_1, c_1) on estetty, testaa kaikki symboliparit (A, B) siten, että A on rivillä r_1 , mutta ei sarakkeessa c_1 , ja B on c_1 :ssä, mutta ei r_1 :ssä, jotta selvitetäisiin mikä pari vaatii vähiten muutoksia. Menetelmä suorittaa sitten muutosalgoritmin tuolle parille.

Palattaessa kuvion 25 tilamatriisiesimerkkiin on edullista ensin muuntaa `space_cnt_mat`-matriisin data kuviossa 24 kuvion 4 muotoon. Kuviossa 34 esitetään tämän muunnoksen tulokset.

Nimetyn väliportaan kytkimen käyttö ilmaistaa X:llä kuvion 34 kohdassa (r_4, c_3) . Seuraavana askeleena on löytää kaikki symboliparit (A, B) siten, että A on rivillä r_4 , mutta ei sarakkeessa c_3 , ja B on sarakkeessa c_3 , mutta ei rivillä r_4 . Tässä esimerkissä rivillä r_4 on merkintä 2, mutta sarakkeessa c_3 ei ole merkintää 2, joten A:lle asetetaan

arvo 2. Samalla tavoin sarakkeessa c_3 on merkintä 4, mutta rivillä r_4 ei ole merkintää 4, joten B:lle asetetaan arvo 4. Koska A:lle on vain yksi arvo ja B:lle on vain yksi arvo, (A,B)-pari, joka vaatii vähiten muutoksia, on ilmeisesti pari (2,4).

Määrätylle symboliparille (A,B) on olemassa kaksi mahdollista uudelleenjärjestelysekvenssiä. Toinen näistä on yleensä lyhyempi. Seuraavat askeleet generoivat $X \rightarrow A$ - ja $X \rightarrow B$ -sekvenssit tätä esimerkkiä varten.

$X \rightarrow A$ -sekvenssi alkaa välittömällä kytkennällä ylimääräisen elementin kautta ja päättyy estetyn solun ollessa asetettu elementtiin A, tai 2, tässä tapauksessa. Aloittamalla estetystä solusta (r_3, c_4) , rivillä r_3 ei ole merkintää 4, joten tämä askel vaatii ympyrän rivillä r_3 olevan merkinnän 2 ympärille. Nyt ympyröity 2 on kohdassa (r_4, c_2) . Sen jälkeen on tarpeen etsiä merkintä 4 sarakkeesta c_2 . Koska sellaista ei ole, tämä sekvenssi päättyy. (Huomaa, että kaikki 2:t eivät tule ympyröidyiksi.)

$X \rightarrow B$ -sekvenssi alkaa välittömällä kytkennällä nime-
tyn elementin 224 läpi ja päättyy estetyn solun ollessa asetettu elementtiin B, tai 4, tässä tapauksessa. Aloitettaessa estetystä solusta (r_3, c_4) , c_4 :ssä ei ole merkintää 2, joten ympyrä menee c_4 :ssä olevan merkinnän 2 ympärille. Nyt ympyröity 4 on kohdassa (r_3, c_3) . Sen jälkeen tapahtuu merkinnän 2 etsiminen r_3 :sta. Koska sellaista ei ole, sekvenssin generointiprosessi on päättynyt. (Huomaa, että kaikki 4:t eivät tule ympyröidyiksi.)

Ylivuotoalgoritmi alkaa löytämällä $X \rightarrow A$ -sekvenssin ensimmäisen elementin. Jos $X \rightarrow A$ -sekvenssi ei ole päättynyt, löydetään $X \rightarrow B$ -sekvenssin ensimmäisen elementti. Sen jälkeen, jos $X \rightarrow B$ -sekvenssi ei ole päättynyt, löydetään seuraava $X \rightarrow A$ -sekvenssin elementti. Tämä prosessi jatkuu kunnes lyhyempi näistä kahdesta sekvenssistä päättyy. (Näin kuviossa 34 esitettyä $X \rightarrow B$ -sekvenssiä ei todellisuudessa generoida. Se on esitetty tässä esimerkkinä vaihtoehtoisesta polusta.)

Yhteenvetona todetaan, että esillä olevan keksinnön mukainen menetelmä ja järjestelmä määrittävät uudelleenjärjesteltävästi estetyille liikennematriisille uudelleenjärjestelysekvenssin, että minimimäärällä liikennematriis-

5 sin uudelleenjärjestelyjä esittämällä liikennematriisi neliömatriisina, jolla on ensimmäinen ulottuvuus joka edustaa liikennematriisin tuloportaita ja toinen ulottuvuus joka edustaa liikennematriisin lähtöportaita, ja jossa solut ensimmäisen ulottuvuuden ja toisen ulottuvuuden

10 leikkauskohdassa edustavat liikennematriisin väliportaan kytkimiä siten, että matriisi identifioi liikennematriisin estetyn väliportaan kytkimen ensimmäisen ulottuvuuden ja toisen ulottuvuuden avulla, ja jossa väliportaan kytkimien pari, jossa yksi väliportaan kytkimistä ei liity samaan

15 lähtöportaan kytkimeen kuin estetty väliportaan kytkin, ja parin toinen väliportaan kytkin ei liity samaan tuloportaan kytkimeen kuin estetty väliportaan kytkin. Rinnakkaisesti määritetään sitten uudelleenjärjestelyaskelten määrä, joka tarvitaan uudelleenjärjestelysekvenssiä varten

20 käyttämällä parin yhtä väliportaan kytkintä, ja liikennematriisin läpi tapahtuvaa uudelleenjärjestelysekvenssiä varten käyttämällä toista väliportaan kytkintä. Näiden askelmäärien tultua määritetyiksi rinnakkaisesti, menetelmä ja järjestelmä päättävät määrittämisoperaatiot määrittetyään ensin lyhyemmän uudelleenjärjestelysekvensseistä

25 käyttämällä mainittua kytkinparia.

Seurauksena yllä olevasta on, että vaikka keksintö on kuvattu yllä olevia toteutuksia tarkastelemalla, toteutuksen kuvausta ei ole tarkoitettu tulkittavaksi rajoittavassa mielessä. Alan ammattimiehelle ovat ilmeisiä kuvattun toteutuksen erilaiset muunnelmät, kuin myös sen vaihtoehtoiset toteutukset, yllä olevan kuvauksen lukemisen jälkeen. Sen vuoksi on ajateltu, että oheiset patenttivaatimukset kattavat sellaiset muunnelmät jotka kuuluvat keksinnön todelliseen vaikutuspiiriin.

35

```

/***** ***** */
/***** ***** */
/*
/*      Bowdonin välittömän kytkennän algoritmi (BICA)
/*
/***** ***** */
/***** ***** */

/*      Aloitusta varten siirry listauksen lopussa olevaan pääohjelmaan.      */

#include <stdio.h>
#include <math.h>
#include <malloc.h>
#include <sys/types.h>
#include <sys/timeb.h>
#include <sys/time.h>

/* Globaalit muuttujat välittömän kytkennän algoritmille.      */
int      in_line,
          out_line;
int      in_stage,
          in_num;
int      out_stage,
          out_num;
int      n,
          m;
int      command_count,
          command_end = 10;
int      rearrange_space,
          rearrange_time;

int      space_in_mat[16][16];
int      space_cnt_mat[17][16];
int      space_out_mat[16][16];
int      space_in_link[16][17];
int      space_out_link[16][17];
int      xtra_stage = 17;
int      traffic[2][28672];

/* Globaalit muuttujat uudelleenjärjestelyalgoritmille.      */
int      BTOA_MV[31][3],
          ATOB_MV[31][3];
int      flag_A = 0,
          flag_B = 0;

int      *sort_array ();
int      *chk_space_in_status ();
int      *chk_space_out_status ();
int      *chk_space_blocking ();
int      *shuffle ();
int      *implement_PM2 ();

char      name_in[13];
char      name_out[13];

static int      in_file[22];
FILE      *fp_in,
          *fp_out;
int      seed_array[10];
int      p_level;

```

```

/* Strukturi nykyisen verkon permutaation säilyttämiseksi ennen permutaatiota */
struct p_coordinate
{
    int      x;
    int      y;
    int      z;
};

struct p_mat_pos
{
    struct p_coordinate position[256];
};

struct p_mat_pos rearrange_3d;

/* Strukturi uudelleenjärjestelysekvenssin tuloksen säilyttämiseksi */
struct seq_len
{
    int input_sw;
    int old_mid_sw;
    int new_mid_sw;
    int output_sw;
};

struct rearrange_seq
{
    struct seq_len move[15];
};

struct rearrange_seq rearrange_moves;

/*****
/* Tämä on aliohjelma BICA:n siirtojen määrälle B-laskennasta
*/
*****/

/*****
/* Tämä funktio lukee ensin uudelleenjärjestelystruktuurin ja poimii kytkennät
/* keskimmaisessa kytkinparissa (A,B). Tämän funktion argumentit ovat aloitus-
/* kohta (RB, CB) ja pari (A, B). Funktio palauttaa siirtojen määrän B:stä
/* alkaen. Vertailun vuoksi todetaan, että Paull nimittää aloituskohtaa (R1, R2)
*****/

chk_Paull_mat_BTOA (RB, CB, B_least, A_least)
int      RB,
         CB;
int      A_least,
         B_least;
{
    int      i,
             j,
             m = 0,
             n = 0,
             y1 = 0,
             y2 = 0,
             y3 = 0,
             n1 = 0,

```

```

int      n2 = 0;
         x1 = 0;
         x2 = 0;
         x3 = 0;
         k1 = 0;
         m1 = 0;
         m2 = 0;
int      flag1 = 0;
         flag2 = 0;
         flag3 = 0;
         temp_flag = 0;
         k2 = 0;
int      r_temp = 0;
         c_temp = 0;
         F2 = 0;
int      chosen_msw_A[16][3];
         chosen_msw_B[16][3];

c_temp = CB;
m = 1;
n = 2;

zero (chosen_msw_A, 48);          /* Alusta kohdematriisit      */
zero (chosen_msw_B, 48);

for (i = 0; i < 256; i++)
{
    x1 = rearrange_3d.position[i].x; /* Lue uudelleenjärjestely-   */
    x2 = rearrange_3d.position[i].y; /* struktuuri                  */
    x3 = rearrange_3d.position[i].z;

    if (x3 == B_least)
    {
        chosen_msw_B[m1][0] = x1; /* Poimi kytkennät välikytki- */
        chosen_msw_B[m1][1] = x2; /* mestä B ja tallenna muuttu- */
        chosen_msw_B[m1][2] = x3; /* jaan chosen_msw_B.          */
        m1++;
    }

    if (x3 == A_least)
    {
        chosen_msw_A[m2][0] = x1; /* Poimi kytkennät välikytki- */
        chosen_msw_A[m2][1] = x2; /* mestä A ja tallenna muuttu- */
        chosen_msw_A[m2][2] = x3; /* jaan chosen_msw_A.          */
        m2++;
    }
}

while (temp_flag == 0)
{
    k1 = 0;
    for (i = 0; i < 16; i++)          /* Aloita (R1, C2):sta,      */
    {                                  /* tarkasta samassa sarakkeessa */
        x1 = chosen_msw_A[i][0];      /* olevat A:t.                */
        x2 = chosen_msw_A[i][1];
        x3 = chosen_msw_A[i][2];

        if (x2 == c_temp && x3 == A_least) /* B löytyi?                  */
        {
            BTOA_MV[m][0] = x1;      /* Kyllä, tallenna nykyinen   */

```

```

        BTOA_MV[m][0] = x2;          /* KytKentä: BTOA_MV:een.      */
        BTOA_MV[m][1] = x3;
        m++;
        r_temp = x1;
        break;
    }

    k1++;
}

if (k1 == 16)
{
    flag1 = 1;
    temp_flag = 1;
    F2 = m;
}

if (flag1 == 0)
{
    k2 = 0;
    for (i = 0; i < 16; i++)          /* Aloita (R1, C2):sta,      */
    {                                  /* tarkasta B:t samalta riviltä. */

        x1 = chosen_msw_B[i][0];
        x2 = chosen_msw_B[i][1];
        x3 = chosen_msw_B[i][2];

        if (x1 == r_temp && x3 == B_least) /* B löytyi?                */
        {
            BTOA_MV[m][0] = x1;          /* Kyllä, tallenna nykyi-    */
            BTOA_MV[m][1] = x2;          /* nen kytKentä              */
            BTOA_MV[m][2] = x3;          /* BTOA_MOV:iin.            */
            m++;
            c_temp = x2;
            break;
        }

        k2++;
    }

    if (k2 == 16)
    {
        flag2 = 1;
        temp_flag = 1;
        F2 = m;
    }
}

return (F2);                          /* Palauta siirtojen määrä B:stä. */
}

/*****
/*
/* Tämä on aliohjelma BICA:n siirtojen määrälle A:sta laskettuna.
/*
/*
*****/

/*****
/* Tämä funktio lukee ensin uudelleenjärjestelystruktuurin ja poimii kytKennät
/* välikytkinparissa (A, B). Argumentit tälle funktiolle ovat aloituspiste
*****/

```

```

/*      /RA, CA) ja pari (A, B). Funktio palauttaa siirtojen määrän A:sta.      */
/*      Vertailun vuoksi todetaan, että Paull kuvaa aloituskohdaksi (R2, C1).      */
/*      *****/
chk_Paull_mat_ATOB (RA, CA, B_least, A_least)
int      RA,
          CA;
int      B_least,
          A_least;
{
    int      i,
             j,
             m = 0,
             n = 0,
             y1 = 0,
             y2 = 0,
             y3 = 0,
             n1 = 0,
             n2 = 0;
    int      x1 = 0,
             x2 = 0,
             x3 = 0,
             k1 = 0,
             m1 = 0,
             m2 = 0;
    int      flag1 = 0,
             flag2 = 0,
             flag3 = 0,
             temp_flag = 0,
             k2 = 0;
    int      r_temp = 0,
             c_temp = 0,
             F1 = 0,
             F2 = 0;
    int      chosen_msw_A[16][3],
             chosen_msw_B[16][3];

    r_temp = RA;
    m = 1;
    n = 2;

    zero (chosen_msw_A, 48);          /* Alusta kohdematriisit.      */
    zero (chosen_msw_B, 48);

    for (i = 0; i < 256; i++)
    {
        x1 = rearrange_3d.position[i].x; /* Lue uudelleenjärjestely-      */
        x2 = rearrange_3d.position[i].y; /* struktuuri.                  */
        x3 = rearrange_3d.position[i].z;

        if (x3 == B_least)
        {
            chosen_msw_B[m1][0] = x1; /* Poimi kytkennät välikytki-      */
            chosen_msw_B[m1][1] = x2; /* mestä B ja tallenna            */
            chosen_msw_B[m1][2] = x3; /* chosen_msw_B.                  */
            m1++;
        }
    }
}

```

```

    if (x3 == A_least)
    {
        chosen_msw_A[m2][0] = x1;    /* Poimi kytkennät välikyt-    */
        chosen_msw_A[m2][1] = x2;    /* kimestä A ja tallenna    */
        chosen_msw_A[m2][2] = x3;    /* chosen_msw_A.            */
        m2++;
    }
}

while (temp_flag == 0)
{
    k1 = 0;
    for (i = 0; i < 16; i++)        /* Aloita (R2, C1):stä, tarkasta */
    {                                /* samalla rivillä olevat B:t.    */
        x1 = chosen_msw_B[i][0];
        x2 = chosen_msw_B[i][1];
        x3 = chosen_msw_B[i][2];

        if (x1 == r_temp && x3 == B_least)    /* A löytyi? */
        {
            ATOB_MV[m][0] = x1;    /* Kyllä, tallenna nykyinen    */
            ATOB_MV[m][1] = x2;    /* kytkentä ATOB_MV.            */
            ATOB_MV[m][2] = x3;
            m++;
            c_temp = x2;
            break;
        }
        k1++;
    }

    if (k1 == 16)
    {
        flag1 = 1;
        temp_flag = 1;
        Fl = m;
    }

    if (flag1 != 1)
    {
        k2 = 0;
        for (i = 0; i < 16; i++)        /* Aloita (R2, C1):stä, tarkasta */
        {                                /* samassa sarakkeessa olevat    */
                                           /* A:t.                            */
            x1 = chosen_msw_A[i][0];
            x2 = chosen_msw_A[i][1];
            x3 = chosen_msw_A[i][2];

            if (x2 == c_temp && x3 == A_least)    /* A löytyi? */
            {
                ATOB_MV[m][0] = x1;    /* kyllä, tallenna    */
                ATOB_MV[m][1] = x2;    /* nykyinen kytkentä    */
                ATOB_MV[m][2] = x3;    /* ATOB_MOV:iin.        */
                m++;
                r_temp = x1;
                break;
            }
            k2++;
        }

        if (k2 == 16)

```

```

    {
        flag2 = 1,
        temp_flag = 1;
        F1 = m;
    }
}
return (F1); /* Palauta siirtojen määrä A:sta. */
}

/*****
/*
/* Tämä on aliohjelma BICA:n uudelleenjärjestelyn tulostamista varten.
/*
/*
/*
*****/

/*****
/*
/* Tämä funktio on uudelleenjärjestelyproseduurin tulos. Argumentit tälle
/* funktiolle ovat vaadittu siirtojen määrä ja pari (A, B). Siirtosekvenssi
/* kirjoitetaan tulostusstruktuuriin rearrange_moves.
/*
*****/

/*****
/* Uudelleenjärjestelyprosessin tulos on seuraavaa muotoa oleva struktuuri:
*
*
* struct seq_len
* {
*     int input_sw;
*     int old_mid_sw;
*     int new_mid_sw;
*     int out_sw;
* }
*
*****/

int rearrangement_output (no_moves, AR, BR)
int no_moves;
int AR;
int BR;
{
    int i,
        j,
        k,
        m,
        n;
    int x1 = 0,
        x2 = 0,
        x3 = 0;
    int y1 = 0,
        y2 = 0,
        y3 = 0;

    for (j = 0; j < no_moves; j++)
    {
        if (flag_A == 1 && flag_B == 0) /* Onko sekvenssi A:sta pienempi? */
        {
            y1 = ATOB_MV[j][0]; /* Lue tulokytkin */
            y2 = ATOB_MV[j][1]; /* Lue lähtökytkin */

```

```

        y3 = ATOB_MV[j][2];          /* Lue välikytkin      */
    }

    if (flag_B == 1 && flag_A == 0) /* Onko sekvenssi B:stä pienempi? */
    {
        y1 = BTOA_MV[j][0];          /* Lue tulokytкин      */
        y2 = BTOA_MV[j][1];          /* Lue lähtökytkin     */
        y3 = BTOA_MV[j][2];          /* Lue välikytkin     */
    }

    rearrange_moves.move[j].input_sw = y1; /* Tallenna tulokytкин */
    rearrange_moves.move[j].old_mid_sw = y3; /* Tallenna vanha välikytkin */
    rearrange_moves.move[j].output_sw = y2; /* Tallenna lähtökytkin */

    /* Store New Middle switch */
    if (y3 == AR) /* Muuta A:t B:ksi. */
        rearrange_moves.move[j].new_mid_sw = BR;

    if (y3 == BR) /* Muuta B:t A:ksi. */
        rearrange_moves.move[j].new_mid_sw = AR;
}
return;

/*****
/*      Tämä on aliohjelma BICA:n nollafunktiota varten.
/*
/*
/*
/*****/

int      zero (A, n)
int      *A;
int      n;
{
    int      i;

    for (i = 0; i < n; i++)
        *(A + i) = 0;
    return;
}

/*****
/*      Tämä on aliohjelma BICA:n etsintäalgoritmin aloituskohdan asetusta
/*
/*      varten.
/*
/*****/

/*****
/*      Tämä funktio asettaa aloituskohdat etsintäalgoritmillemme A:sta ja B:stä ja
/*      suorittaa algoritmin. Siirrettävien kytkentöjen joukko tallennetaan joko
/*      ATOB_MV:een tai BTOA_MV:een jos vastaavasti jokin sekvenssi A:sta on lyhyempi
/*      tai sekvenssi B:stä on lyhyempi. Argumentteina tälle funktiolle ovat väli-
/*      kytkinpari (A, B), indeksi ja estetty matriisipositio (R1, C1). Funktio
/*      palauttaa lyhyimmän sekvenssin pituuden.
/*
/*****/

```

```

int      rearrange_3d (r1_B, c1_A, sw_index, R1, C1)
int      r1_B;
int      c1_A;
int      sw_index;
int      R1;
int      C1;
{
    int      i,
            j,
            k = 0,
            x1 = 0,
            x2 = 0,
            x3 = 0,
            m,
            n,
            temp_mov = 0,
            temp = 0;
    int      num_moves = 0,
            mr = 0,
            mc = 0,
            w_seq = 0;
    int      R[3],
            C[3],
            undef_flag = 0;
    int      num_moves_A = 0,
            num_moves_B = 0,
            F1 = 0;
    int      F2 = 0,
            R1_CONN[16][3],
            C1_CONN[16][3];

    zero (R, 15);
    zero (C, 15);
    zero (ATOB_MV, 93);
    zero (BTOA_MV, 93);
    zero (R1_CONN, 48);
    zero (C1_CONN, 48);

    m = 0;
    n = 0;

    R[0] = R1;
    C[0] = C1;

    BTOA_MV[0][0] = R[0];
    ATOB_MV[0][1] = C[0];

    for (i = 0; i < 256; i++) /* Poimi välikytkimet (R1, C1):stä. */
    {
        x1 = rearrange_3d.position[i].x; /* Lue struktuuri. */
        x2 = rearrange_3d.position[i].y;
        x3 = rearrange_3d.position[i].z;

        if (x1 == R1)
        {
            R1_CONN[m][0] = x1; /* Tallenna välikytkimet R1:een. */
            R1_CONN[m][1] = x2;
            R1_CONN[m][2] = x3;
            m++;
        }
    }
}

```

```

    if (x2 == C1)
    {
        C1_CONN[n][0] = x1;          /* Tallenna välikytkimet C1:een. */
        C1_CONN[n][1] = x2;
        C1_CONN[n][2] = x3;
        n++;
    }
}

for (i = 0; i < 15; i++)          /* Etsi matriisipositio R2, C2. */
{
    x1 = R1_CONN[i][2];          /* R1_CONN on kytkennät R1:ssä */
    x2 = C1_CONN[i][2];          /* C1_CONN on kytkennät C1:ssä */

    if (x1 == r1_B)
        C[1] = R1_CONN[i][1];
    if (x2 == c1_A)
        R[1] = C1_CONN[i][0];
}

BTOA_MV[0][1] = C[1];          /* Aseta aloituskohta haulle A:sta ja B:stä. */
ATOB_MV[0][0] = R[1];          /* A on (R2, C1); B on (R1, C2):sta. */

BTOA_MV[0][2] = r1_B;          /* 3. RTOA_MV:n ja ATOB_MV:n elementti on */
ATOB_MV[0][2] = c1_A;          /* vuorotteleva sekvenssi A:ta ja B:tä. */
                                /* A:t muutetaan B:ksi ja B:t muutetaan */
                                /* A:ksi. */
/* Aseta A:n aloituskohta. */
mr = R[1];
mc = C[0];

/* Etsintä algoritmi A:sta. */
num_moves_A = chk_Paull_mat_ATOB (mr, mc, r1_B, c1_A); /* Korvaa */
                                                        /* ylivuo- */
                                                        /* tamalla */

/* Aseta B:n aloituskohta */
mr = R[0];
mc = C[1];
/* Etsintä algoritmi B:stä. */

num_moves_B = chk_Paull_mat_BTOA (mr, mc, r1_B, c1_A); /* Korvaa */
                                                        /* ylivuo- */
                                                        /* tamalla */

F1 = num_moves_A;
F2 = num_moves_B;

if (F1 < F2)                    /* Onko num_moves_A < num_moves_B? */
{
    w_seq = 1;                  /* Jos on, aseta flag_A = 1, flag_B = 0. */
    num_moves = F1;
    flag_A = 1;
    flag_B = 0;
}

if (F2 < F1)                    /* Onko num_moves_B < number_moves_A? */
{
    w_seq = 2;                  /* Jos on, aseta flag_A = 0, flag_B = 1. */
    num_moves = F2;
    flag_A = 0;
    flag_B = 1;
}

```

```

    }
    if (F1 == F2)                /* Onko number_moves_A = number_moves_B? */
    {
        w_seq = 1;                /* Jos on, oletusarvoksi sekvenssi A:sta. */
        num_moves = F1;
        flag_A = 1;                /* ja aseta flag_A = 1, flag_B = 0. */
        flag_B = 0;
    }
    return (num_moves);          /* Palauta siirtojen pienin määrä. */
}

/*****
/* Tämä on aliohjelma BICA:n toteuta muutos algoritmia varten.
/*
/*
/*
*****/

/*****
/* Tämä ohjelma toteuttaa etsintäalgoritmin uudelleenjärjestelyyn tarvittavien
/* siirtojen sekvenssin löytämiseksi. Argumenttina tälle funktiolle ovat
/* välikytkinpari (msw_A, msw_B) ja indeksi.
/*
*****/

int                implement_change (msw_B, msw_A, index)
int                msw_B;
int                msw_A;
int                index;
{
    int            i,
                j,
                k;
    int            len_of_seq = 0;

    len_of_seq = rearrange_seq (msw_B, msw_A, index);
    rearrangement_output (len_of_seq, msw_A, msw_B);
    return (len_of_seq);
}

/*****
/* Tämä on aliohjelma BICA:n siirtojen määrän laskemiseksi.
/*
/*
/*
*****/

/*****
/* Tämä rutiini käyttää implement_change_algorithm -funktia löytääkseen
/* siirtojen määrän, joka tarvitaan siirtosekvenssille sw_A:sta ja siirtosek-
/* venssille sw_B:stä väliportaana kytkinparissa olemassa oleville kytkennöille
/* ja palauttaa siirtojen määrän joka tarvitaan pienemmässä näistä kahdesta
/* sekvenssistä.
*****/

int                calc_nom (sw_B, sw_A, index)
int                sw_B,
                sw_A,
                index;
{

```

```

    int          num_moves;
    num_moves = implement_range (sw_B, sw_A, index);
    return(num_moves);
}

/*****
/*
/*   Tämä on aliohjelma BICA:n matriisimallimuunnosta varten.
/*
/*
*****/

/*****
/*
/*   Esillä oleva Paullin menetelmän toteutus tarvitsee kytkentädatan
/*   seuraavassa struktuurissa.
/*
/*       struct p_coordinate
/*       {
/*           int input_sw;
/*           int output_sw;
/*           int cnt_sw;
/*       }
/*
/*       struct p_mat_pos
/*       {
/*           struct p_coordinate[256];
/*       }
/*
*****/

void          matrix_model_conversion ()
{
    int          i,
                j,
                x1,
                x2,
                x3,
                m1 = 0;

    for(i = 1; i <= 17; i++)
    {
        for(j = 1; j <= 17; j++)
        {
            x1 = i;
            x2 = j;
            x3 = space_cnt_mat[i][j];
            if(x3 != 0)
            {
                rearrange_3d.position[m1].x = x2;      /* Tulokytkin      */
                rearrange_3d.position[m1].y = x3;      /* Lähtökytkin    */
                rearrange_3d.position[m1].z = x1;      /* Välikytkin     */
                m1++;
            }
        }
    }
    return;
}

/*****
/*
/*   Tämä on aliohjelma BICA:n versiolle Paullin menetelmä II:sta
*****/

```



```

    }
}

m = 0;
n = 0;
i = 0;
while ((x1 = listofB[i]) != 0)          /* Luo kaikki listofA:n ja      */
{                                       /* listofB:n elementtien      */
    j = 0;                             /* yhdistelmät                */
    while ((x2 = listofA[j]) != 0)
    {
        cnt_stage_pairs[m][0] = x1;
        cnt_stage_pairs[m][1] = x2;
        m++;
        j++;
    }
    i++;
}

least_moves = xtra_stage;
for (k = 0; k < m; k++)                /* Kullekin välikytkimien      */
{                                       /* yhdistelmälle laske pienin  */
    B_temp = cnt_stage_pairs[k][0];    /* tarvittava siirtojen määrä. */
    A_temp = cnt_stage_pairs[k][1];

    if (B_temp != 0 && A_temp != 0)
    {
        num_moves = calc_nom (B_temp, A_temp, k);
    }

    if (num_moves < least_moves)
    {
        least_moves = num_moves;
        least_pair = m;
    }
}

*(selected_pair + 0) = cnt_stage_pairs[least_pair][1];
*(selected_pair + 1) = cnt_stage_pairs[least_pair][0];

return (selected_pair);
}

/*****
Seuraavat ovat korjattuja aliohjelmia ja pääohjelma BICA:aa varten.
*****/

/*****
Tämä on aliohjelma BICA:n törmäyksetöntä aikavudelleenjärjestelyä
varten.
*****/

int      hitless_time_rearrange ()

```

```

{
    /* Käytä Include-tiedostoa.
    return;
}

/*****
/*      Tämä on aliohjelma BICA:n ylivuoda aika-algoritmia varten.
/*
/*
/*
/*****/

int      flood_time ()
{
    /* Käytä Include-tiedostoa.
    return;
}

/*****
/*      Tämä on aliohjelma BICA:n kahden aikavälin valintaa varten.
/*
/*
/*
/*****/

int      select_time_slots ()
{
    /* Käytä Include-tiedostoa.
    return;
}

/*****
/*      Tämä on aliohjelma BICA:n törmäyksetöntä tilauudelleenjärjestelyä
/*      varten.
/*
/*
/*****/

int      hitless_space_rearrange ()
{
    /* Käytä Include-tiedostoa.
    return;
}

/*****
/*      Tämä on aliohjelma BICA:n ylivuoda tila-algoritmia varten.
/*
/*
/*
/*****/

int      flood_space ()
{
    /* Käytä Include-tiedostoa.
    return;
}

/*****
/*      Tämä on aliohjelma BICA:n pumpkaa yhteys -makroa varten.
/*
/*
/*
/*****/

/*****/

```

```

/* Tämä makro "summaa" loogisesti space_in_link/i/:n space_out_link/k):n */
/* kanssa numeroltaan alhaisimman väliportaan kytkimen "C" määrittämiseksi, */
/* jolla on sekä vapaa yhteys tuloportaaseen "i" ja vapaa yhteys lähtöpor- */
/* taaseen "k". Ensimmäinen nollasta poikkeava arvo loogisen "and"-operaation */
/* tuloksessa on "c" ja mainitut kaksi yhteyttä "c":hen nollataan arvoon "0" */
/* (merkitään varatuiksi) space_in_link/i/:ssä ja space_out_link/k/:ssa. Makro */
/* palauttaa arvon "c" */
/* **** */

```

```

int      pump_link ()
{
    int      c,
             j,
             x;

    for (j = 1; j <= xtra_stage; j++)
    {
        x = space_in_link[in_stage][j] && space_out_link[out_stage][j];

        if (x != 1)
        {
            c = j;
            break;
        }
    }

    clear_link(space_in_link, in_stage, c);
    clear_link(space_out_link, out_stage, c);

    if (c == xtra_stage)
    {
        rearrange_space = 1;
    }

    /* Palauta väliportas.      */
    return(c);
}

```

```

/* **** */
/* Tämä on aliohjelma BICA:n tyhjennä vapaan yhteyden bitti (arvoon 0)-toimin- */
/* netta varten. */
/* **** */

```

```

/* **** */
/* Tätä primitiiviä käytetään tyhjentämään (nollaamaan) io:n määrittelemän */
/* bittikartan j:n:n rivin k. bitti arvoon "0". Jos bittikartta sisältää n */
/* elementtiä ja arvo j > n, palautetaan virhelippu. Muussa tapauksessa k. */
/* elementti nollataan arvoon "0". */
/* **** */

```

```

int      clear_link(io, j, k)
int      io[16][17],
          j,
          k;
{
    int      error_flag;

    if(k <= xtra_stage)
    {

```

```

        io[j][k] = 0;
        error_flag = 0;
    }
    else
    {
        error_flag = 1;
    }
    return(error_flag);
}

/*****
/* Tämä on aliohjelma BICA:n aseta vapaan yhteyden bitti (arvoon 1) -toimin- */
/* netta varten. */
*****/

/*****
/* Tätä primitiiviä käytetään asettamaan io:n määrittelemän bittikartan */
/* j:n:n rivin k. bitti arvoon "1". Jos bittikartta sisältää n elementtiä */
/* ja arvo j > n, palautetaan virhelippu. Muussa tapauksessa k. elementti */
/* asetetaan arvoon "1". */
*****/

int      set_link(io, j, k)
int      io[16][17],
        j,
        k;
{
    int      error_flag;

    if(k <= xtra_stage)
    {
        io[j][k] = 1;
        error_flag = 0;
    }
    else
    {
        error_flag = 1;
    }
    return(error_flag);
}

/*****
/* Tämä on aliohjelma BICA:n valitse kaksi väliportaan kytkentä -toiminnetta */
/* varten. */
*****/

int      select_cnt_stage ()
{
    int *center_stage_pair;

    center_stage_pair = (int *)calloc(2, sizeof(int));

    /* valitse väliportaan kytkinpari. */

    center_stage_pair = implement_PM2 ();

    return(*center_stage_pair);
}

```

```

/*****
/*
/* Tämä on aliohjelma BICA:n aikauudelleenjärjestelyä varten.
/*
/*
*****/
Solmu D
/* Node D */
void time_rearrange ()
{
    int          XtoA = 0,
                XtoB = 0;
    char          last;

    /* Valitse kaksi aikaväliä uudelleenjärjestelyä */
    select_time_slots ();          varten.

    /* Etsi uudelleenjärjestettävät kytkennät. */
    flood_time ();

    /* Q6: Mikä polku vaatii pienimmän määrän uudelleenjärjestelyä? */
    if (last == 'A')
    {
        XtoA = 1;
    }
    else
    {
        if (last == 'B')
        {
            XtoB = 1;
        }
    }
    hitless_time_rearrange ();
    rearrange_time = 0;
    return;
}

/*****
/*
/* Tämä on aliohjelma BICA:n tilauudelleenjärjestelyä varten.
/*
/*
*****/
Solmu C */
void space_rearrange ()
{
    int          XtoA = 0,
                XtoB = 0;
    int          *cnt_stage_pair;
    char          last;

    /* Luovuta muisti siirrettävien kytkentöjen listan osoittimelle. */
    cnt_stage_pair = (int *)calloc(2, sizeof(int));

    /* Valitse kaksi väliportaana kytkintä uudelleenjärjestelyä varten. */
    *cnt_stage_pair = select_cnt_stage ();

    /* Etsi uudelleenjärjestettävät kytkennät. */
    flood_space ();

```

```

/* Q5: Mikä polku vaatii pienimmän määrän uudelleenjärjestelyjä? */
if (last == 'A')
{
    XtoA = 1;
}
else
{
    if (last == 'B')
    {
        XtoB = 1;
    }
}
hitless_space_rearrange ();
rearrange_space = 0;
return;
}

/*****
/* Tämä on aliohjelma BICA:n jäännösfunktiota varten.
/*
/*
/*****
/* Tämä funktio tuottaa jäännöksen sen jälkeen kun jakaja on vähennetty
/*
/* jakajasta niin monta (kokonaisluku) kertaa kuin mahdollista.
/*
/*****/

int      rem(dividend, divisor)
int      dividend,
         divisor;
{
    int      remainder;

    if (dividend <= 0)
    {
        remainder = 0;
    }
    else
        remainder = dividend;
    while (remainder >= divisor)
    {
        remainder = remainder - divisor;
    }
    return(remainder);
}

/*****
/* Tämä on aliohjelma BICA:n osamääräfunktiota varten.
/*
/*
/*****
/* Tämä funktio tuottaa osamäärän vähentämällä jakajan jaettavasta niin monta
/*
/* (kokonaisluku) kertaa kuin mahdollista.
/*
/*****/

int      quot(dividend, divisor)
int      dividend,

```

```

        divisor;

    {
        int            quotient,
                      remainder;

        quotient = 0;
        if (dividend <= 0)
        {
            remainder = 0;
        }
        else
            remainder = dividend;
        while (remainder >= divisor)
        {
            remainder = remainder - divisor;
            quotient++;
        }
        return(quotient);
    }

/*****
/* Tämä on aliohjelma BICA:n muunna I/O -linjan numerot -toiminnetta varten.    */
/*                                                                              */
/*                                                                              */
*****/

/*****
/* Tämä aliohjelma käyttää osamäärän ja jäännöksen funktioita I/O-linjan      */
/* numeroiden muuntamiseksi I/O-portaan kytkinnumeroiden pariin ja I/O-portaan */
/* kytkimen nastanumeron kytkemiseksi tai siitä purkamiseksi.                */
*****/

void            convert_line ()
{
    int            r = 16,
                  in_line_minus_1,
                  out_line_minus_1;

    in_line_minus_1 = in_line - 1;
    in_stage = quot(in_line_minus_1, r) + 1;
    in_num = rem(in_line_minus_1, r) + 1;

    out_line_minus_1 = out_line - 1;
    out_stage = quot(out_line_minus_1, r) + 1;
    out_num = rem(out_line_minus_1, r) + 1;

    return;
}

/*****
/* Tämä on aliohjelma BICA:n tulosta tilastot - n ja m -toiminnetta varten    */
/*                                                                              */
/*                                                                              */
*****/

int            print_stats ()
{
    /* Testitapaus print_stats:ia varten.
    printf ("BICA performed %d connect(s) and %d disconnect(s).\n", n, m);

```

```

    return;
}

/*****
/* Tämä on aliohjelma BICA:n purkua varten - tulosta lähtöön.
/*
/*
/*
*****/

/* Node B */
void disconnect ()
{
    int          cnt_stage,
                csw;          /* csw väliportaan kytkimelle */

/* Muunna IL-m -> OL-n purkupyynnö I (i, j) >< O(k, l) -muotoon.
convert_line ();
convert_line ();

/* Poista kytkennät pyynnön mukaan.
    csw = space_in_mat[in_stage][in_num];
    space_in_mat[in_stage][in_num] = 0;
    space_cnt_mat[csw][in_stage] = 0;
    space_out_mat[out_stage][out_num] = 0;

    * Päivitä taulukot.
    set_link(space_in_link, in_stage, csw);
    set_link(space_out_link, out_stage, csw);

/* Q7: Pakkaa kytkennät sen jälkeen kun A:n purku sallittu*/
    if (rearrange_space == 1)
        /* A3 =Kyllä-> Solmu C */
        {
            space_rearrange ();
        }
    /* A3 = EI.
    else
        /* Päätä tila-alue.
        {
            /* Q4: Aikauudelleenjärjestely tarvitaan*/
            if (rearrange_time == 1)
                /* A4 = KYLLÄ->Solmu D
                {
                    time_rearrange ();
                }
            /* A4 = EI.
            /* Päätä aika-alue.
        }
    return;
}

/*****
/* Tämä on aliohjelma BICA:n kytkentää varten - tulosta lähtöön.
/*
/*
/*
*****/

/*****
/* Tämä rutiini kytkee tulolinjan lähtölinjaan tilamatriisin läpi.
/*
/* Muuttujat määritellään seuraavasti:
*/

```

```

/*
/*  in_line:  tulo tilamatriisiin                               */
/*              0 <= in_line <= 255                             */
/* out_line:  lähtö tilamatriisista                             */
/*              0 <= out_line <= 255                             */
/*
/*  in_stage:  tilamatriisin tuloporras                         */
/*              0 <= in_stage <= 15                             */
/*   in_num:   tilamatriisin tuloportaan tulo                   */
/*              0 <= in_number <= 15                             */
/*
/* out_stage:  lähtöporras tilamatriisista                     */
/*              0 <= out_stage <= 15                             */
/*   out_num:  lähtö tilamatriisin lähtöportaasta              */
/*              0 <= out_number <= 15                             */
/*
/* cnt_stage:  tilamatriisin väliporras                         */
/*              0, 1 <= center_stage <= 17                     */
/* HUOMAUTUS:  center stage = 0, tarkoittaa, että mitään tilamatriisin
/*              väliporrasta ei ole varattu                     */
/*              center_stage = 1, ..., 16 tarkoittaa, että tilamatriisin
/*              vastaava väliporras on varattu.                 */
/*              center_stage = 17 on tilamatriisin ylimääräinen porras ja
/*              uudelleenjärjestely on käynnissä                */
/*
/******
/* Solmu A*/
void      connect ()
{
    int          csw;                      /* csw väliportaan kytkimelle. */
/* Muunna IL-m -> OL-n kytkentäpyyntö I (i, j) -> O (k, l) -muotoon. */
    convert_line ();
/* Aloita aika-alue. */
/* Valitse numeroltaan alhaisin kytkentään käytettävissä oleva aikaväli. */
/* Aloita tila-alue. */
/* Valitse numeroltaan alhaisin kytkentään käytettävissä oleva väliportaan kytkin. */
    csw = pump_link ();
/* Suorita kytkentä ja päivitä taulukot. */
    space_in_mat[in_stage][in_num] = csw;
    space_cnt_mat[csw][in_stage] = out_stage;
    space_out_mat[out_stage][out_num] = csw;
/* Q3: Tilauudelleenjärjestely vaaditaan? */
    if (rearrange_space == 1)
        /* A3 = KYLLÄ-> Solmu C */
        {
            space_rearrange ();
        }
/* A3 = EI. */
    else
        /* Päätä tila-alue. */
        {
            /* Q4: Aikauudelleenjärjestely vaaditaan? */

```

```

        if (rearrange_time == 1)
            /* A4 = KYLLÄ -> Solmu D */
            {
                time_rearrange ();
            }
        /* A4 = EI. */
        /* Päättää aika-arvo. */
    }
    return;
}

/*****
/* Tämä on aliohjelma BICA:n hae komento -toiminnetta varten. */
/* */
/* */
*****/

int get_command ()
{
    int command;

    /* Test case for get_command. */
    if (command_count < command_end)
    {
        /* command = 1 => connect. */
        command = 1;
        command_count++;
        in_line = command_count;
        out_line = command_count;
    }
    else if (command_count == command_end)
    {
        /* command = -1 => disconnect. */
        command = -1;
        command_count++;
    }
    else
    {
        /* command = 0 => end. */
        command = 0;
    }
    return(command);
}

/*****
/* Tämä on aliohjelma BICA:n tilauudelleenjärjestelyn alustusta varten. */
/* */
/* */
*****/

void initialize_rearrange ()
{
    int i;

    for (i = 0; i < 256; i++)
    {
        rearrange_3d.position[i].x = 0;
        rearrange_3d.position[i].y = 0;
        rearrange_3d.position[i].z = 0;
    }
}

```

```

    for (i = 0; i < 15; i++ ,
    {
        rearrange_moves.move[i].input_sw = 0;
        rearrange_moves.move[i].old_mid_sw = 0;
        rearrange_moves.move[i].new_mid_sw = 0;
        rearrange_moves.move[i].output_sw = 0;
    }
    return;
}

/*****
/* Tämä on aliohjelma BICA:n tilamatriisin alustusta varten.
/*
/*
/*
*****/

/*****
/* Tätä rutiinia käytetään alussa tyhjentämään (nollaamaan) tilamatriisit.
/*
*****/

void initialize_space_mat()
{
    int i, j;

    for(i = 0; i < xtra_stage; i++)
    {
        for(j = 0; j < xtra_stage; j++)
        {
            space_in_mat[i][j] = 0;
            space_cnt_mat[i][j] = 0;
            space_out_mat[i][j] = 0;
        }
        space_cnt_mat[xtra_stage][i] = 0;
    }
    return;
}

/*****
/* Tämä on aliohjelma BICA:n alusta vapaan yhteyden bittikartat -toiminnetta
/*
/* varten.
/*
*****/

/*****
/* Alusta yhteyden bittikartat-primitiivi (käytetään vain käynnistysoperaatioita
/* varten) alustaa sekä tilatulon vapaan yhteyden bittikartan että tilalähtö-
/* yhteyden bittikartan. Kaikki arvot vapaan yhteyden bittikartoissa alustetaan
/* arvoon "1", mikä osoittaa kaikkien yhteyksien olevan vapaita.
*****/

void init_links()
{
    int i, j;

    for(i = 0; i < xtra_stage; i++)
    {
        for(j = 0; j <= xtra_stage; j++)
        {
            space_in_link[i][j] = 1;

```

```

        space_out_link[i][j] = 1;
    }
}
return;
}

/*****
/*      Tämä on aliohjelma BICA:n alustusta varten.
/*
/*
/*
*****/

void      initialize ()
{
    /* Testidata komentoja varten */
    n = 0;
    m = 0;

    command_count = 0;
    rearrange_space = 0;
    rearrange_time = 0;

    /* Alusta kaikki matriisit. */
    init_links();
    initialize_space_mat ();
    initialize_rearrange ();

    return;
}

/*****
/*      Tämä on Bowdonsin välittömän kytkennän algoritmin pääohjelma.
/*
/*
/*
*****/

main ()
{
    int      command;

    /* Aloita ohjelmavuo tästä. */
    initialize ();
    do
    {
        command = get_command ();

        /* Q1: Porttipari kytkettävä? */
        /* Komento = 1 => kytkee */
        if (command == 1)
            /* A1 = KYLLÄ-> Solmu A */
            {
                connect ();
                n = n++;
            }
        /* A1 = EI. */
        else
            /* Q2: Porttipari purettava? */
            {
                /* Komento = -1 => pura. */

```

```

        if (command == 1)
            /* A2 = KYLLÄ-> Solmu B */
            {
                disconnect ();
                m = m++;
            }
        /* A2 = EI. */
        continue;
    }
}
/* komento = 0 => loppu */
while (command != 0);
{
    print_stats ();
}
}

```

Patenttivaatimukset:

1. Menetelmä, jolla määritetään minimimäärä uudelleenjärjestelyjä tuloportin kytkemiseksi lähtöporttiin
5 uudelleenjärjesteltävästi estetyn liikennematriisin läpi,
t u n n e t t u siitä, että se sisältää seuraavat vaiheet:

liikennematriisin esittämisen taulukolla, jolla on ensimmäinen ulottuvuus ja toinen ulottuvuus, mainitun ensimmäisen ulottuvuuden edustaessa liikennematriisin useita
10 tuloportaan kytkimiä, mainitun toisen ulottuvuuden edustaessa liikennematriisin useita lähtöportaan kytkimiä, mainitun taulukon sisältäessä useita soluja, kunkin mainitun solun edustaessa mainitun ensimmäisen ulottuvuuden
15 risteyskohtaa mainitun toisen ulottuvuuden kanssa ja edustaessa mahdollista väliportaan kytkimen kytkentää liikennematriisin läpi;

liikennematriisin estetyn väliportaan kytkimen identifioinnin mainitun ensimmäisen ulottuvuuden ja mainitun toisen ulottuvuuden avulla;

20 väliportaan kytkinparin, joka sisältää ensimmäisen väliportaan kytkimen ja toisen väliportaan kytkimen, identifioinnin siten, että mainittu ensimmäinen väliportaan kytkin ei liity samaan lähtöportaan kytkimeen kuin mainittu estetty väliportaan kytkin, eikä mainittu toinen väliportaan kytkin liity samaan tuloportaan kytkimeen kuin
25 mainittu estetty väliportaan kytkin;

mainitusta tulokytkimestä mainittuun lähtökytkimeen mainitun liikennematriisin läpi suoritettavaan uudelleenjärjestelysekvenssiin tarvittavien uudelleenjärjestelyaskelten määrän, mainittua ensimmäistä väliportaan kytkintä ensin käytettäessä, ja toiseen uudelleenjärjestelysekvenssiin mainitusta tulokytkimestä liikennematriisin läpi mainittuun lähtökytkimeen tarvittavien uudelleenjärjestelyas-

kelten määrän, mainittua toista väliportaan kytkintä ensin käytettäessä, määrittämisen rinnakkaisesti; ja

mainitun määrittämissäskelen päättämisen, kun on ensin määritetty lyhyempi mainituista ensimmäisestä ja toisesta
5 uudelleenjärjestelysekvenssistä.

2. Järjestelmä uudelleenjärjestelyjen minimimäärän määrittämiseksi kytkettäessä tuloportti lähtöporttiin uudelleenjärjesteltävästi estetyn liikennematriisin läpi,
t u n n e t t u siitä, että se sisältää:

10 taulukon liikennematriisin esittämiseksi, mainitun taulukon sisältäessä ensimmäisen ulottuvuuden ja toisen ulottuvuuden, mainitun ensimmäisen ulottuvuuden edustaessa liikennematriisin useita tuloportaan kytkimiä, mainitun toisen ulottuvuuden edustaessa liikennematriisin useita
15 lähtöportaan kytkimiä, mainitun taulukon sisältäessä useita soluja, kunkin solun edustaessa mainitun ensimmäisen ulottuvuuden risteyskohtaa mainitun toisen ulottuvuuden kanssa ja edustaessa mahdollista väliportaan kytkimen kytkentää liikennematriisin läpi;

20 keinon identifioida liikennematriisin estetty väliportaan kytkin mainitun ensimmäisen ulottuvuuden ja mainitun toisen ulottuvuuden avulla;

väliportaan kytkinparin, joka sisältää ensimmäisen väliportaan kytkimen ja toisen väliportaan kytkimen siten,
25 että mainittu ensimmäinen väliportaan kytkin ei liity samaan lähtöportaan kytkimeen kuin mainittu estetty väliportaan kytkin ja että mainittu toinen väliportaan kytkin ei liity samaan tuloportaan kytkimeen kuin mainittu estetty väliportaan kytkin;

30 määrittäyskeinon, jolla määritetään rinnakkaisesti uudelleenjärjestelyaskelten, jotka tarvitaan uudelleenjärjestelysekvenssiin liikennematriisin läpi mainitusta tulo-
kytkimestä mainittuun lähtökytkimeen mainittua ensimmäistä väliportaan kytkintä ensin käyttämällä ja toiseen uudelleenjärjestelysekvenssiin liikennematriisin läpi mainitus-
35

ta tulokytimestä mainittuun lähtökytkimeen mainittua toista väliportaan kytkintä ensin käyttämällä, määrä; ja päättämiskeinon, jolla päätetään mainitun määrityskeinon toiminta kun on ensin määritetty lyhyempi mainituista ensimmäisestä ja toisesta uudelleenjärjestelyse-

5 venssistä.

3. Menetelmä, jolla määritetään minimimäärä uudelleenjärjestelyjä tuloportin kytkemiseksi lähtöporttiin uudelleenjärjesteltävästi estetyn liikennematriisin läpi,

10 t u n n e t t u siitä, että se sisältää seuraavat askeleet:

liikennematriisin esittämisen neliömatriisilla, jossa on useita rivejä ja useita sarakkeita, mainittujen rivien edustaessa liikennematriisin useita tuloportaan

15 kytkimiä, mainittujen sarakkeiden edustaessa liikennematriisin useita lähtöportaan kytkimiä, mainitun neliömatriisin sisältäessä useita soluja, kunkin mainitun solun edustaessa mainittujen rivien risteyskohtaa mainittujen sarakkeiden kanssa ja edustaessa mahdollista väliportaan kytkimen kytkentää liikennematriisin läpi;

20

liikennematriisin estetyn väliportaan kytkimen identifioinnin mainittujen rivien ja mainittujen sarakkeiden avulla;

väliportaan kytkinten parin, joka sisältää ensimmäisen väliportaan kytkimen ja toisen väliportaan kytkimen, identifioinnin mainittujen rivien ja mainittujen sarakkeiden avulla siten, että mainittu väliportaan kytkin ei liity samaan lähtöportaan kytkimeen kuin mainittu estetty väliportaan kytkin ja että mainittu toinen väliportaan kytkin ei liity samaan tuloportaan kytkimeen kuin mainittu estetty väliportaan kytkin;

25

30

mainitusta tulokytimestä mainittuun lähtökytkimeen liikennematriisin läpi tapahtuvassa uudelleenjärjestelysekvenssissä tarvittavien uudelleenjärjestelyaskelten,

35 mainittua ensimmäistä väliportaan kytkintä ensin käyttä-

mällä, ja mainitusta tulokytimestä mainittuun lähtökytkimeen liikennematriisin läpi tapahtuvassa toisessa uudelleenjärjestelysekvenssissä tarvittavien uudelleenjärjestelyaskelten, mainittua toista väliportaan kytkintä ensin
 5 käyttämällä, määrän mainitun neliömatriisin läpi määrittämisen rinnakkaisesti useilla askeleilla; ja

mainitun määritysaskeleen päättämisen kun on ensin määritetty lyhyempi mainituista ensimmäisestä ja toisesta uudelleenjärjestelysekvenssistä sen mukaan kummassa on
 10 pienempi määrä askelia mainitun neliömatriisin läpi.

4. Järjestelmä, jolla määritetään minimimäärä uudelleenjärjestelyjä tuloportin kytkemiseksi lähtöporttiin uudelleenjärjesteltävästi estetyn liikennematriisin läpi,
 t u n n e t t u siitä, että se sisältää:

15 neliömatriisin liikennematriisin esittämiseksi, mainitun taulukon sisältäessä useita rivejä ja useita sarakkeita, mainitun rivin edustaessa liikennematriisin useita tuloportaan kytkimiä, mainittujen sarakkeiden edustaessa liikennematriisin useita lähtöportaan kytkimiä,
 20 mainitun taulukon sisältäessä useita soluja, kunkin mainitun solun edustaessa mainitun rivin risteyskohtaa mainittujen sarakkeiden kanssa ja edustaessa mahdollista väliportaan kytkimen kytkentää liikennematriisin läpi;

keinon identifioida estetty väliportaan kytkin muodostettaessa kytkentää liikennematriisin nimetyn väliportaan kytkimen läpi mainittujen rivien ja mainittujen sarakkeiden avulla;

parin väliportaan kytkimiä, joka identifioidaan mainituilla riveillä ja mainituilla sarakkeilla ja joka
 30 sisältää ensimmäisen väliportaan kytkimen ja toisen väliportaan kytkimen siten, että mainittu ensimmäinen väliportaan kytkin ei liity samaan lähtöportaan kytkimeen kuin mainittu estetty väliportaan kytkin ja että mainittu toinen väliportaan kytkin ei liity samaan tuloportaan kytkimeen kuin mainittu estetty väliportaan kytkin;
 35

määrityskeinon, jolla mainitusta tulokytimestä mainittuun lähtökytkimeen liikennematriisin läpi tapahtuvassa uudelleenjärjestelysekvenssissä tarvittavien uudelleenjärjestelyaskelten, mainittua ensimmäistä väliportaan

5 kytkintä ensin käyttämällä, ja mainitusta tulokytimestä mainittuun lähtökytkimeen liikennematriisin läpi tapahtuvassa toisessa uudelleenjärjestelysekvenssissä tarvittavien uudelleenjärjestelyaskelten, mainittua toista väliportaan kytkintä ensin käyttämällä, määrä mainitun neliömatriisin läpi määritetään rinnakkaisesti useilla aske-

10 leilla; ja

päätämiskeinon, joka päättää mainitun määrityslaitteen toiminnan kun on ensin määritetty lyhyempi mainituista ensimmäisestä ja toisesta uudelleenjärjestelysekvenssistä sen mukaan kummassa on pienempi määrä askelia mainitun neliömatriisin läpi.

15

5. Menetelmä, jolla määritetään minimimäärä uudelleenjärjestelyjä tuloportin kytkemiseksi lähtöporttiin uudelleenjärjesteltävästi estetyn liikennematriisin läpi,

20 t u n n e t t u siitä, että se sisältää seuraavat askeleet:

liikennematriisin esittämisen taulukolla, jossa on ensimmäinen ulottuvuus ja toinen ulottuvuus, mainitun ensimmäisen ulottuvuuden edustaessa liikennematriisin useita tuloportaan kytkimiä, mainitun toisen ulottuvuuden edustaessa liikennematriisin useita lähtöportaan kytkimiä,

25 mainitun taulukon sisältäessä useita soluja, kunkin mainitun solun edustaessa mainitun ensimmäisen ulottuvuuden risteyskohtaa mainitun ensimmäisen ulottuvuuden kanssa ja edustaessa mahdollista väliportaan kytkimen kytkentää liikennematriisin läpi;

30

liikennematriisin uudelleenjärjesteltävästi estetyn väliportaan kytkimen identifioinnin mainitun ensimmäisen ulottuvuuden ja mainitun toisen ulottuvuuden avulla;

useiden väliportaan kytkimien, jotka sisältävät ensimmäisen väliportaan kytkimen ja toisen väliportaan kytkimen siten, että mainittu ensimmäinen väliportaan kytkin ei liity samaan lähtöportaan kytkimeen kuin mainittu
 5 estetty väliportaan kytkin ja että mainittu toinen väliportaan kytkin ei liity samaan tuloportaan kytkimeen kuin mainittu estetty väliportaan kytkin, identifioinnin;

mainitusta tulokytkimestä mainittuun lähtökytkimeen liikennematriisin läpi tapahtuvaan uudelleenjärjestelysekvenssiin mainittua ensimmäistä väliportaan kytkintä ensin
 10 käyttämällä tarvittavien uudelleenjärjestelyaskelten määrän ja mainitusta tulokytkimestä mainittuun lähtökytkimeen liikennematriisin läpi tapahtuvaan toiseen uudelleenjärjestelysekvenssiin mainittua toista väliportaan kytkintä
 15 ensin käyttämällä tarvittavien uudelleenjärjestelyaskelten määrän määrittämisen rinnakkaisesti kullekin mainituista pareista; ja

mainitun määritysaskelen päättämisen kun on ensin määritetty lyhyempi mainitusta ensimmäisestä uudelleenjärjestelysekvenssistä ja mainitusta toisesta uudelleenjärjestelysekvenssistä.
 20

6. Järjestelmä, jolla määritetään minimimäärä uudelleenjärjestelyjä tuloportin kytkemiseksi lähtöporttiin uudelleenjärjesteltävästi estetyn liikennematriisin läpi,
 25 t u n n e t t u siitä, että se sisältää:

taulukon liikennematriisin esittämiseksi, mainitun taulukon sisältäessä ensimmäisen ulottuvuuden ja toisen ulottuvuuden, mainitun ensimmäisen ulottuvuuden edustaessa liikennematriisin tuloportaan useita kytkimiä, ja mainitun
 30 toisen ulottuvuuden edustaessa liikennematriisin lähtöportaan useita kytkimiä, mainitun taulukon sisältäessä useita soluja, kunkin mainitun solun edustaessa mainitun ensimmäisen ulottuvuuden risteyskohtaa mainitun toisen ulottuvuuden kanssa ja edustaessa mahdollista väliportaan kytkimen kytkentää liikennematriisin läpi;
 35

keinon identifioida uudelleenjärjesteltävästi es-
tetty väliportaan kytkin ilmaisemalla kytkentä liikenne-
matriisin nimetyn väliportaan kytkimen läpi mainitun en-
simmäisen ulottuvuuden ja mainitun toisen ulottuvuuden
5 avulla;

useita väliportaan kytkimien pareja, jotka sisäl-
tävät ensimmäisen väliportaan kytkimen ja toisen välipor-
taan kytkimen siten, että mainittu ensimmäinen väliportaan
kytkin ei liity samaan lähtöportaan kytkimeen kuin mainit-
10 tu estetty väliportaan kytkin ja että mainittu toinen vä-
liportaan kytkin ei liity samaan tuloportaan kytkimeen
kuin mainittu estetty väliportaan kytkin;

määrityskeinon, jolla mainitusta tulokytimestä
mainittuun lähtökytkimeen liikennematriisin läpi tapahtu-
15 vaan uudelleenjärjestelysekvenssiin mainittua ensimmäistä
väliportaan kytkintä ensin käyttämällä tarvittavien uudel-
leenjärjestelyaskelten määrä ja mainitusta tulokytimestä
mainittuun lähtökytkimeen liikennematriisin läpi tapahtu-
vaan toiseen uudelleenjärjestelysekvenssiin mainittua
20 toista väliportaan kytkintä ensin käyttämällä tarvittavien
uudelleenjärjestelyaskelten määrä määritetään rinnakkai-
sesti kullekin mainituista pareista; ja

päättämiskeinon, jolla mainitun määrityskeinon toi-
minta päätetään kun on ensin määritetty lyhyempi maini-
25 tuista ensimmäisestä ja toisesta uudelleenjärjestelysek-
venssistä.

FIG.1

128 AIKAVÄLIÄ TULOJA KOHDEN
3/10-HYLLESTÄ MATRIISIIN

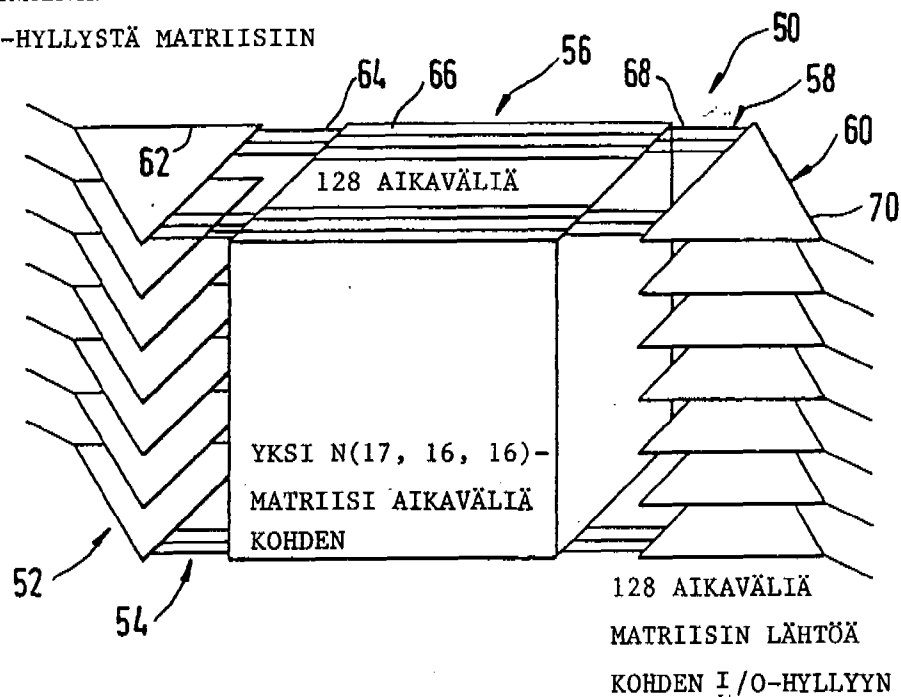
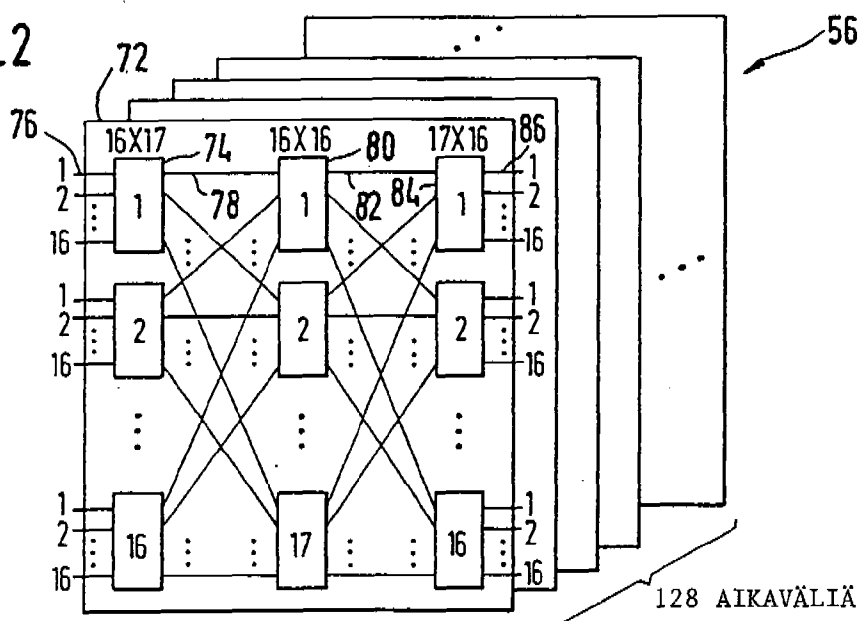


FIG.2



LOOGISESTI YKSI N(17, 16, 16)-MATRIISI AIKAVÄLIÄ KOHDEN

FIG. 3

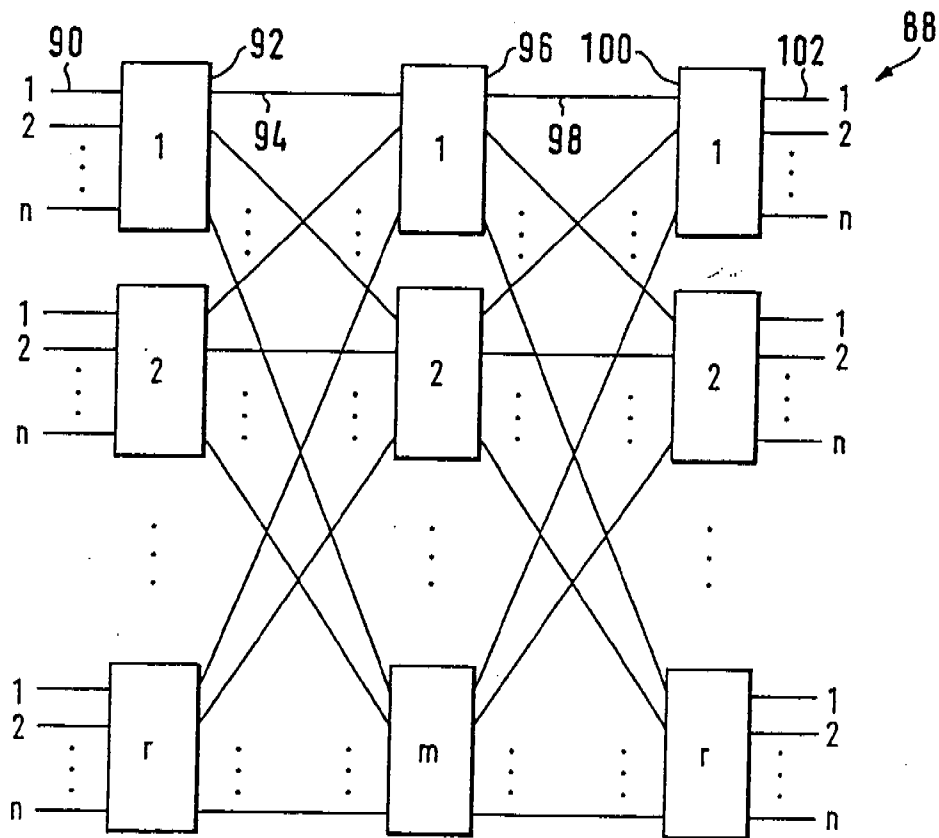


FIG. 4

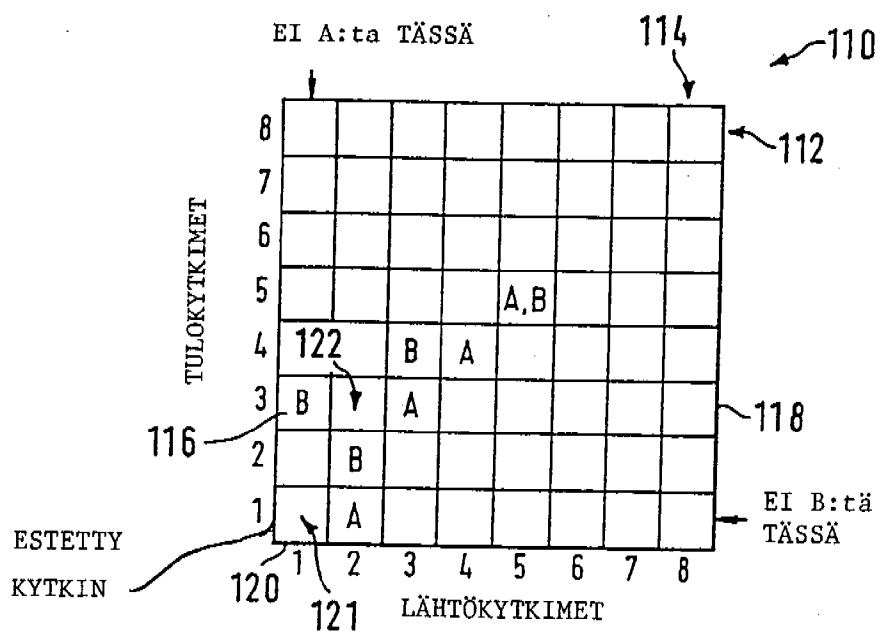


FIG. 5

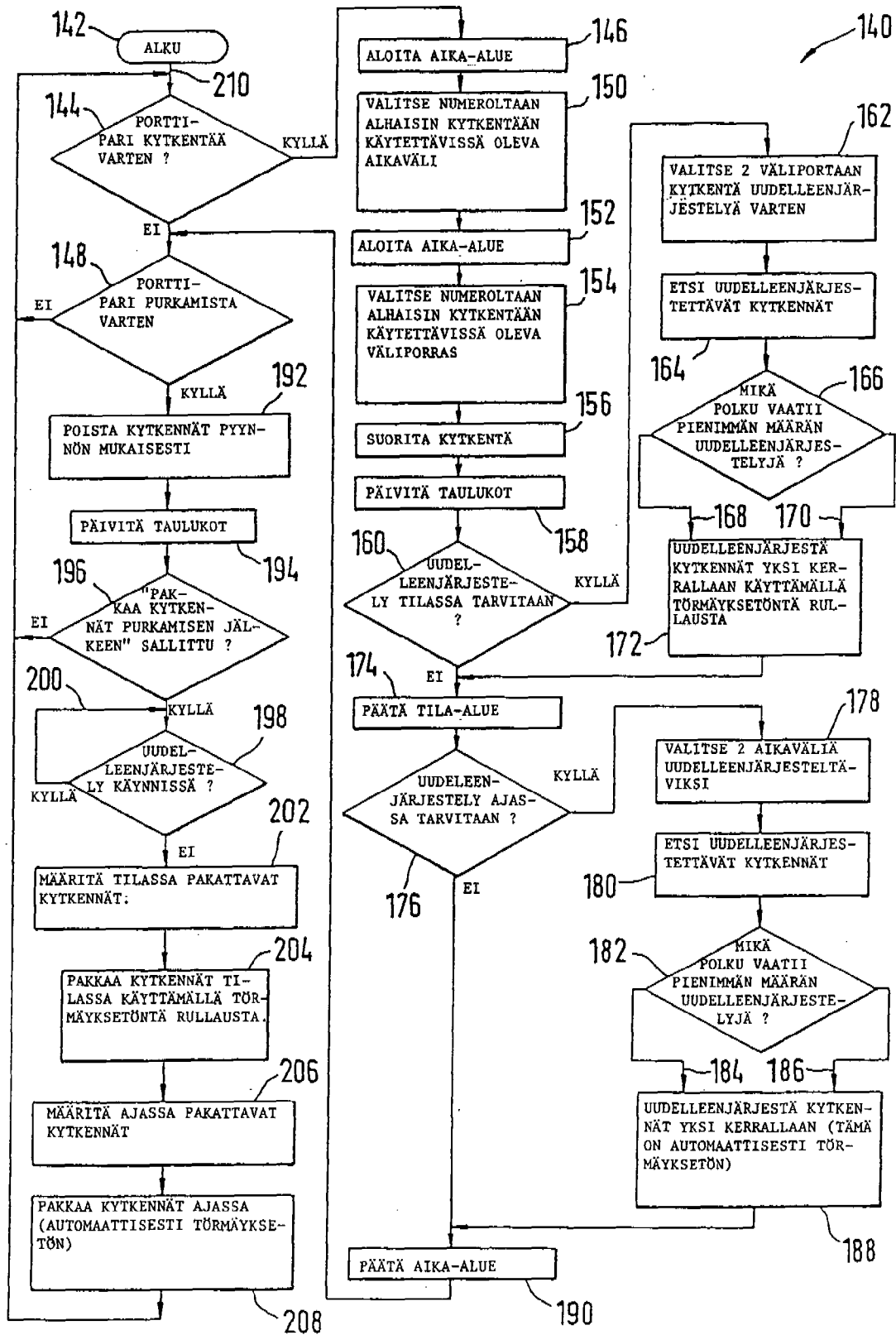


FIG. 6

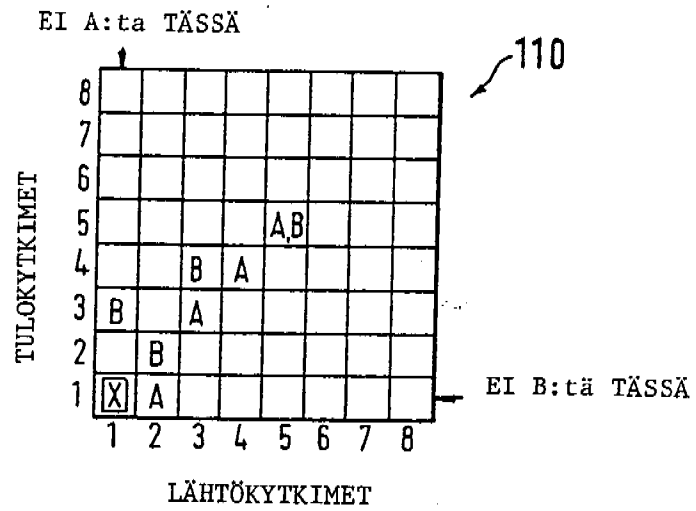


FIG. 7

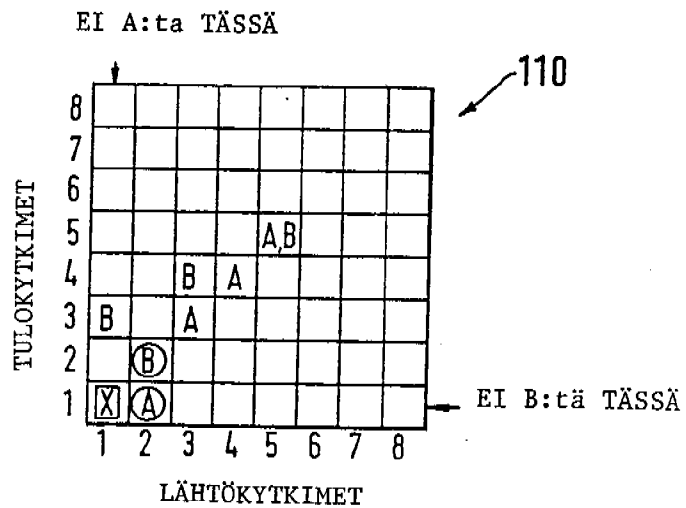


FIG. 8

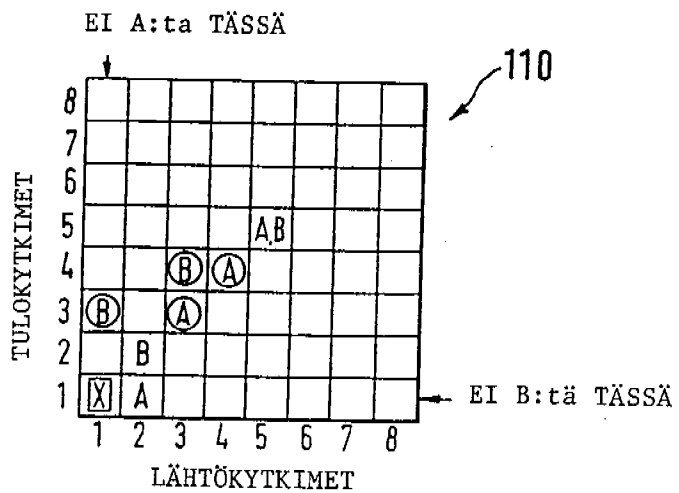


FIG. 9

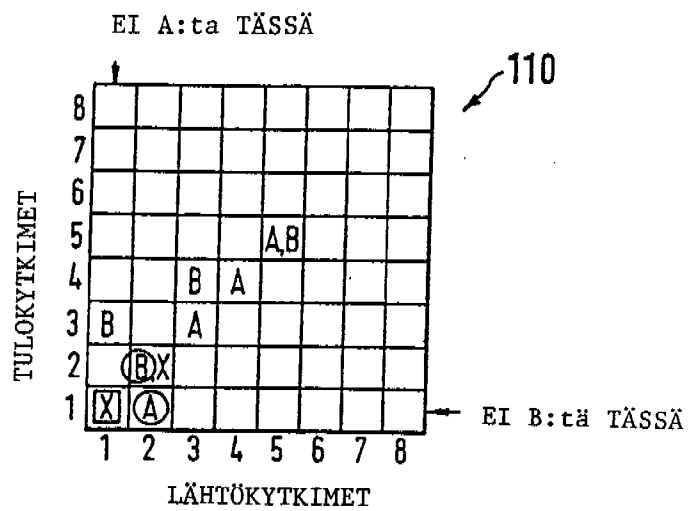


FIG. 10

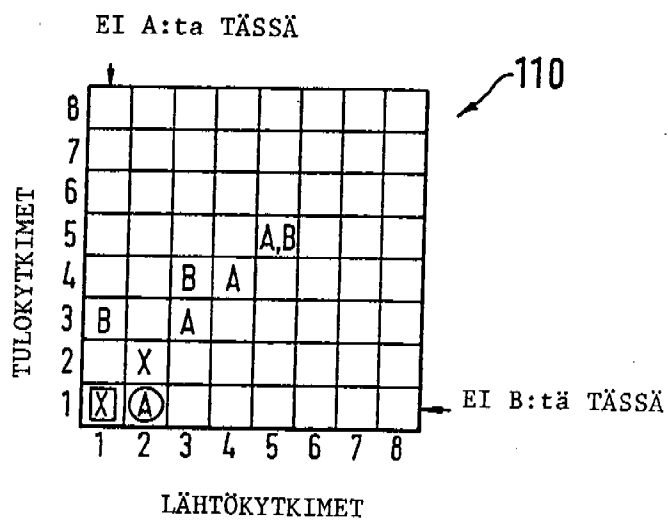


FIG. 11

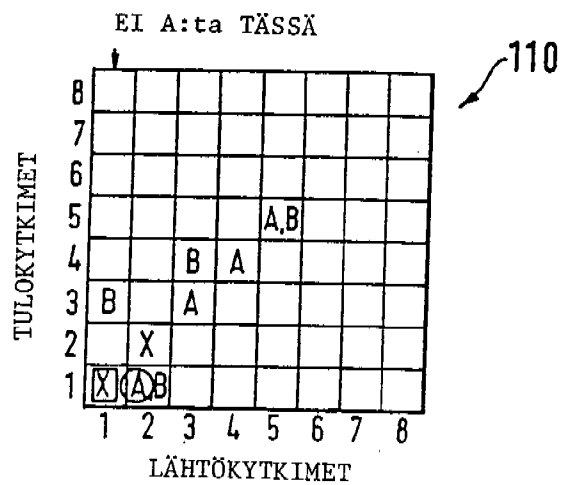


FIG. 12

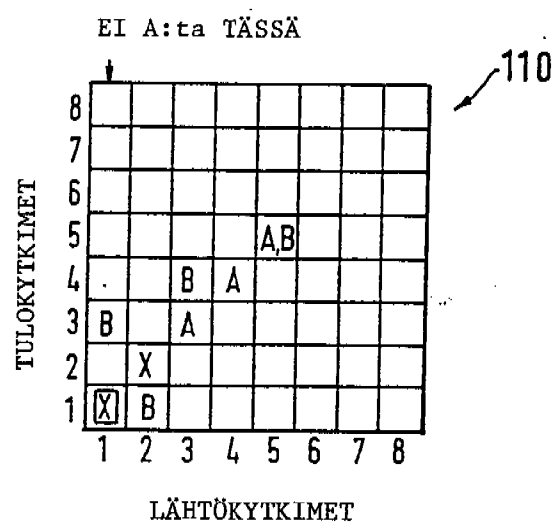


FIG. 13

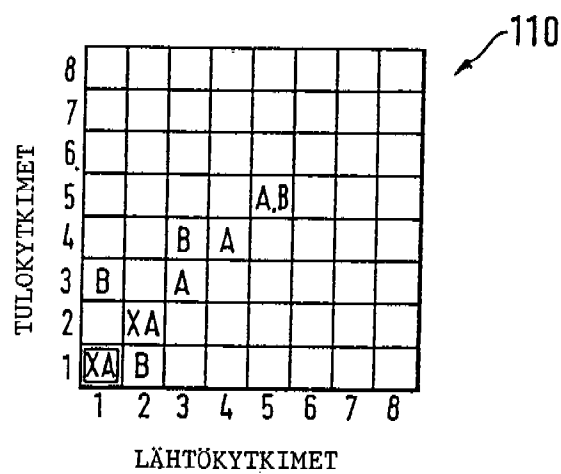


FIG. 14

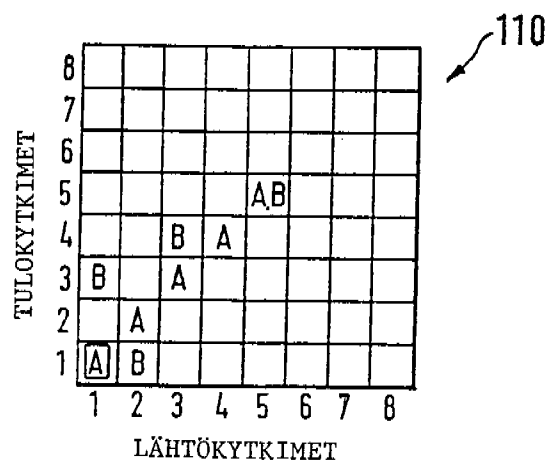


FIG. 15

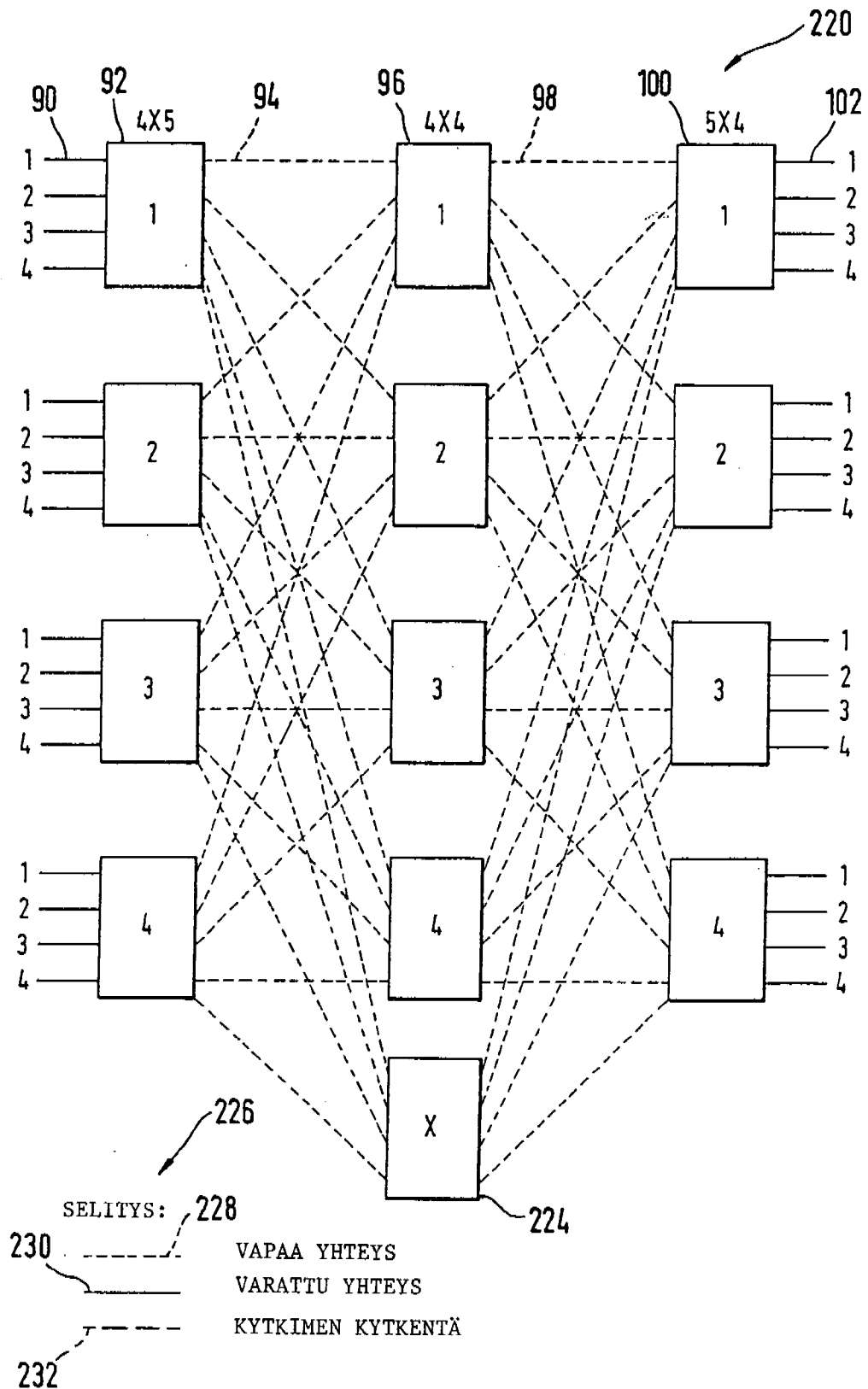


FIG. 16

222

	1	2	3	4
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	(0)

238

	1	2	3	4	X
1	1	1	1	1	1
2	1	1	1	1	1
3	1	1	1	1	1
4	1	1	1	1	(1)

234

	1	2	3	4
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0
X	0	0	0	(0)

236

	1	2	3	4
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	(0)	0	0	0

240

	1	2	3	4	X
1	1	1	1	1	1
2	1	1	1	1	1
3	1	1	1	1	1
4	1	(1)	1	1	1

FIG. 17

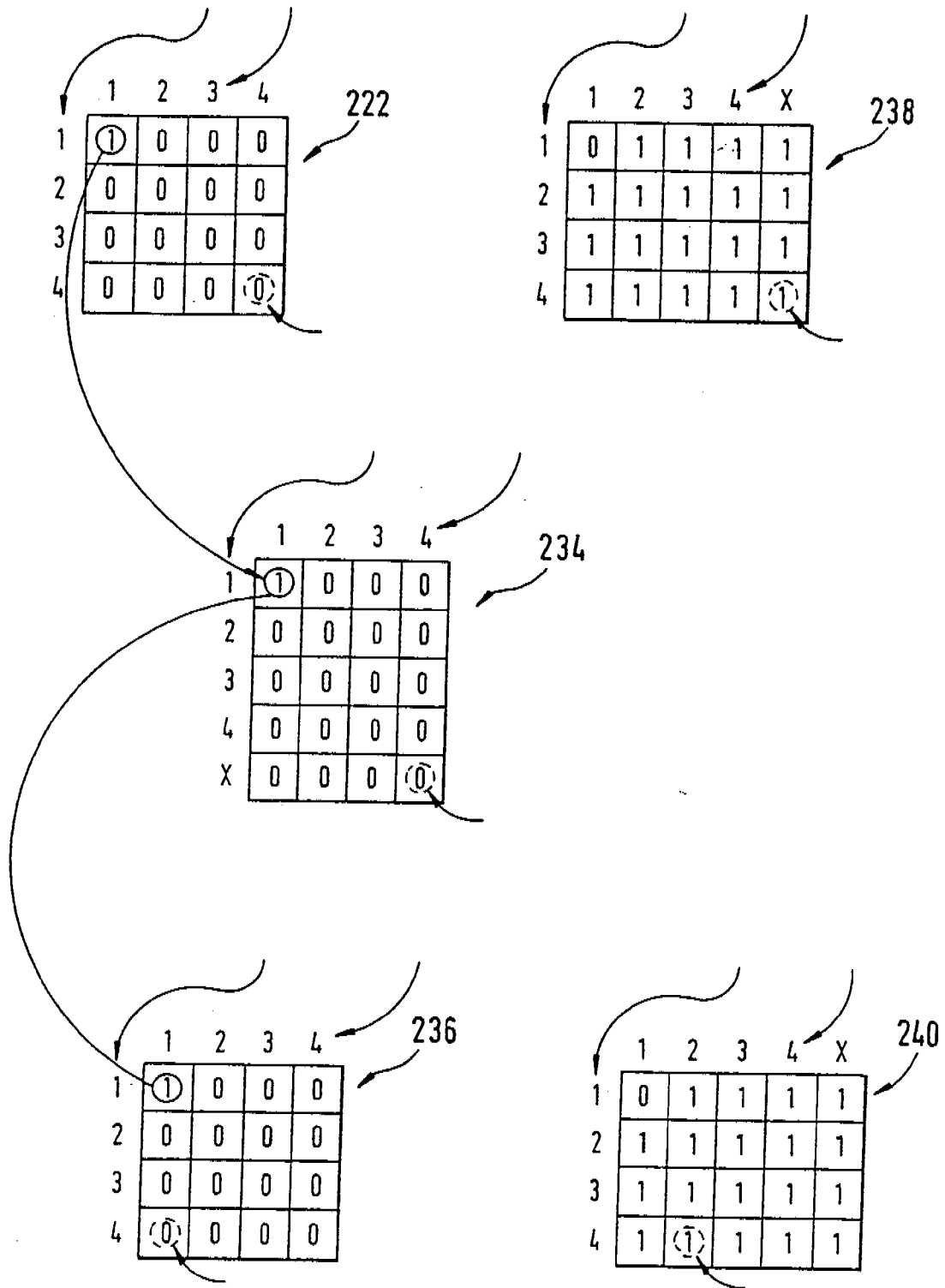


FIG. 18

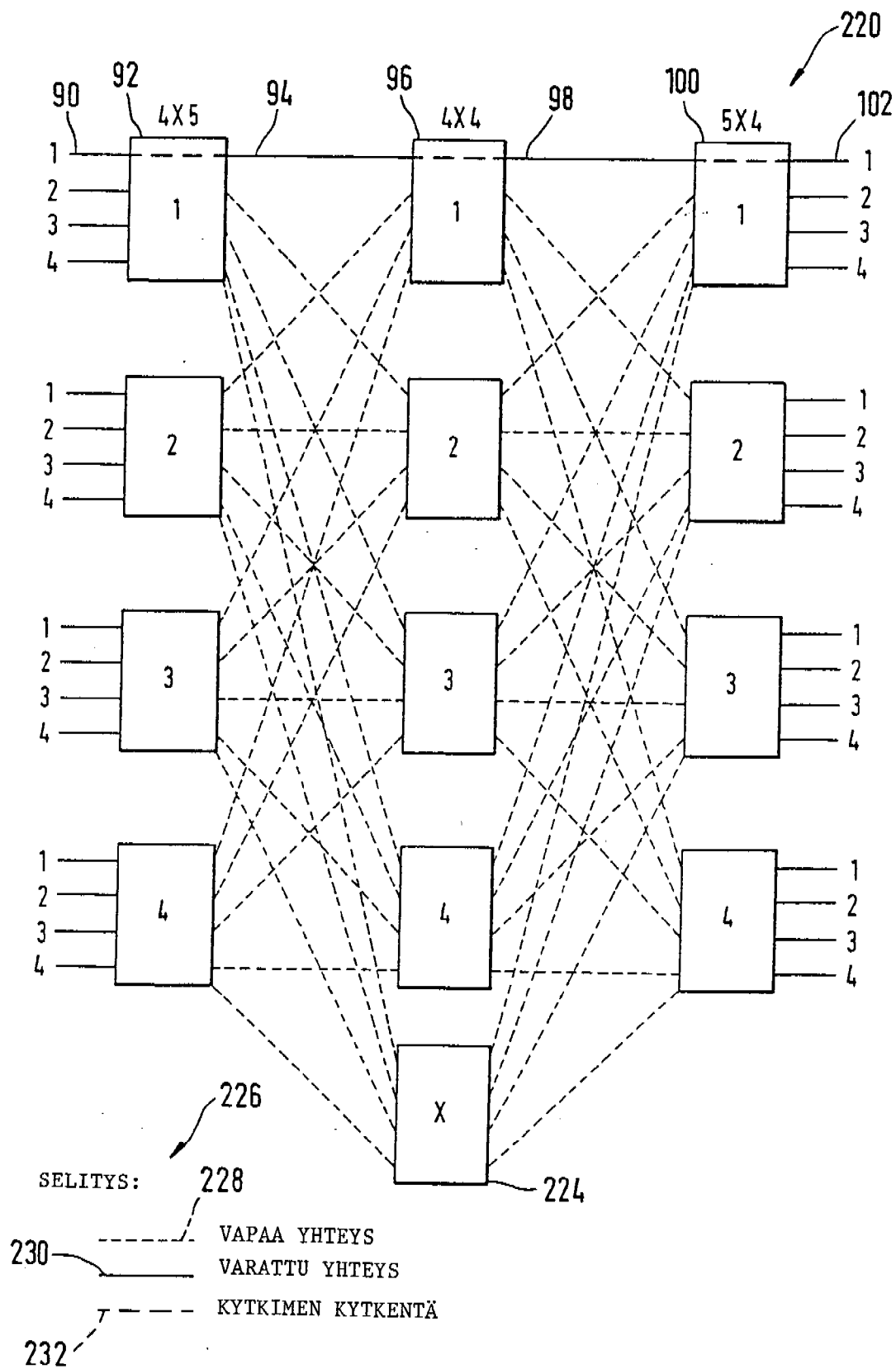


FIG. 19

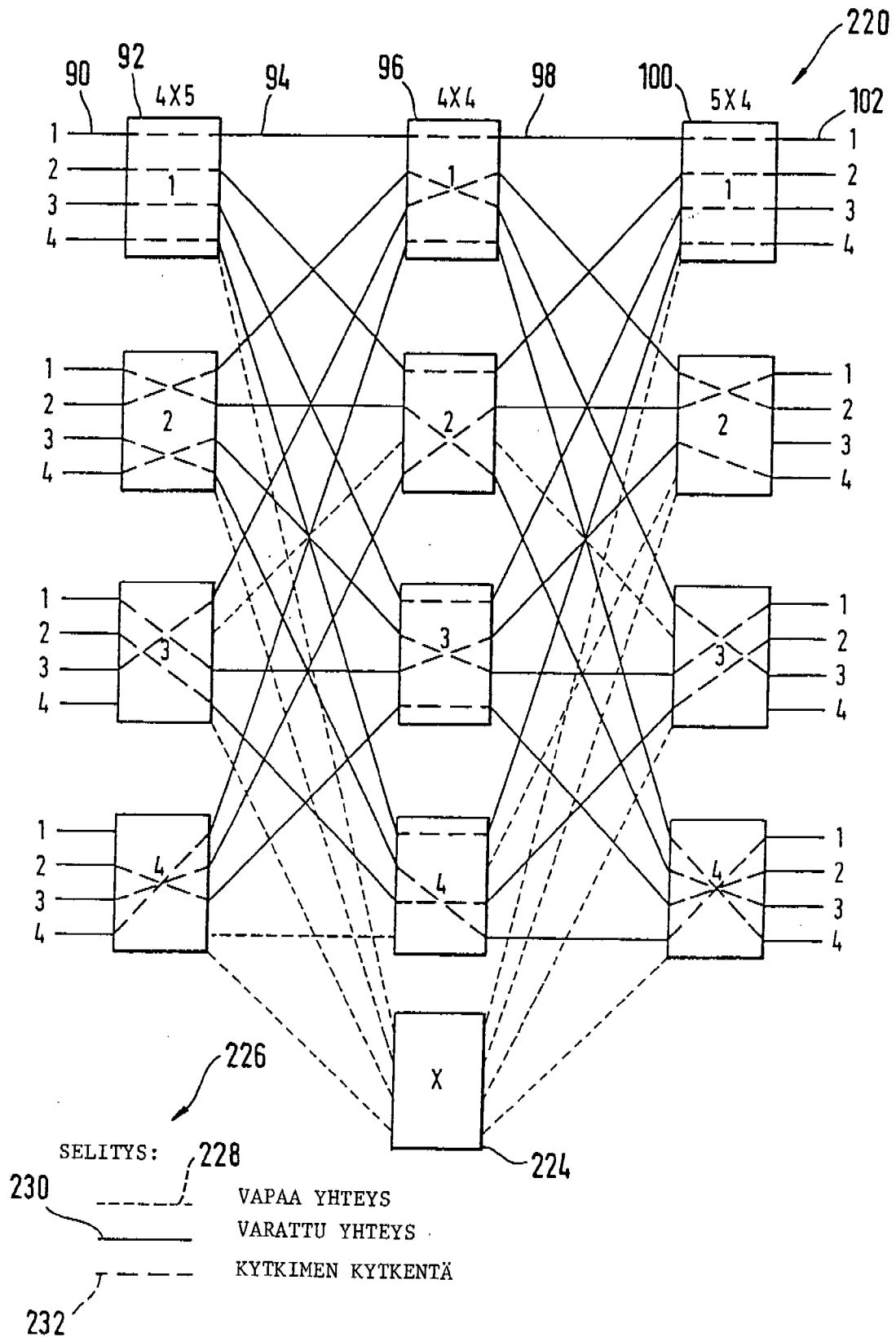


FIG. 20

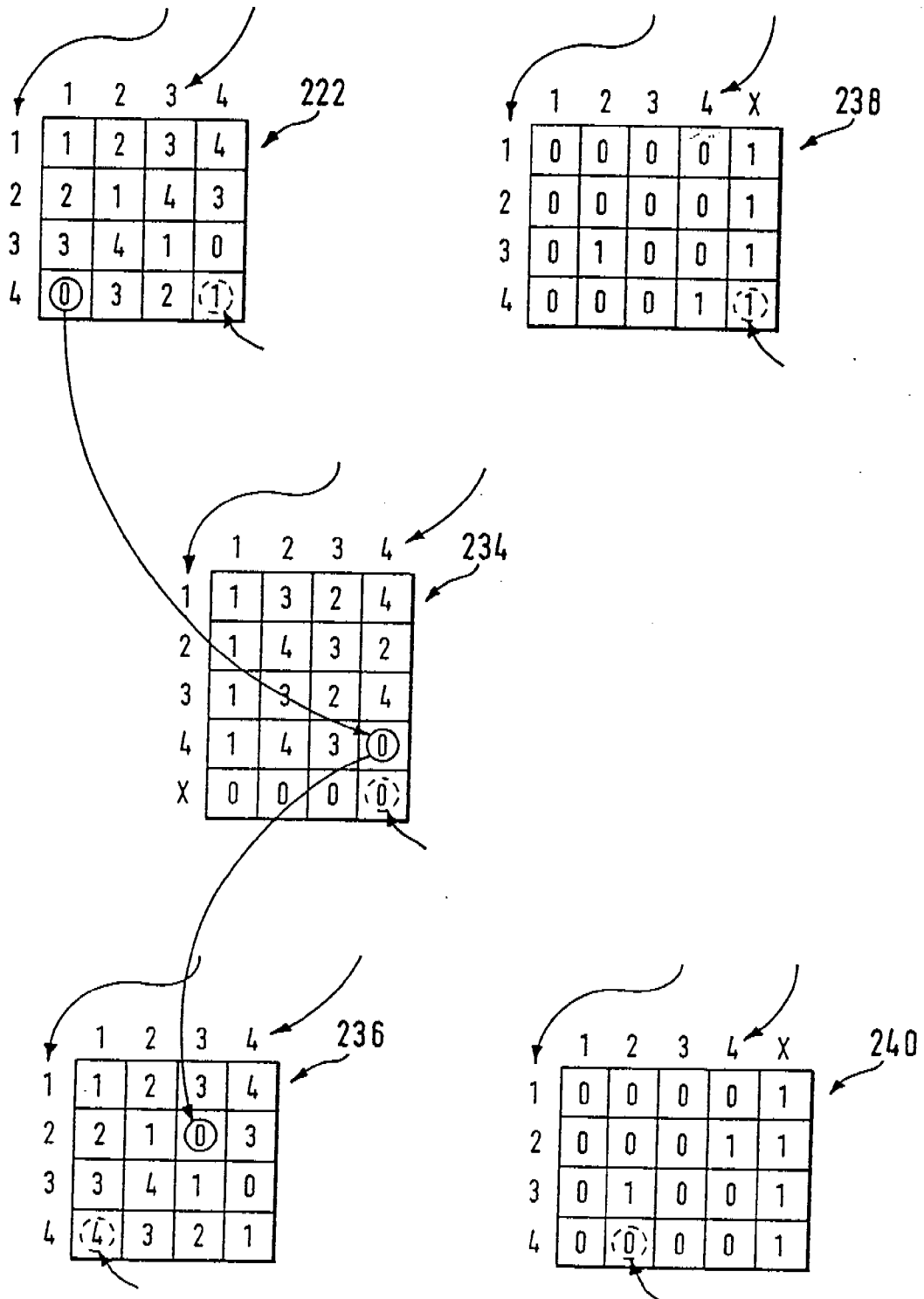


FIG. 21

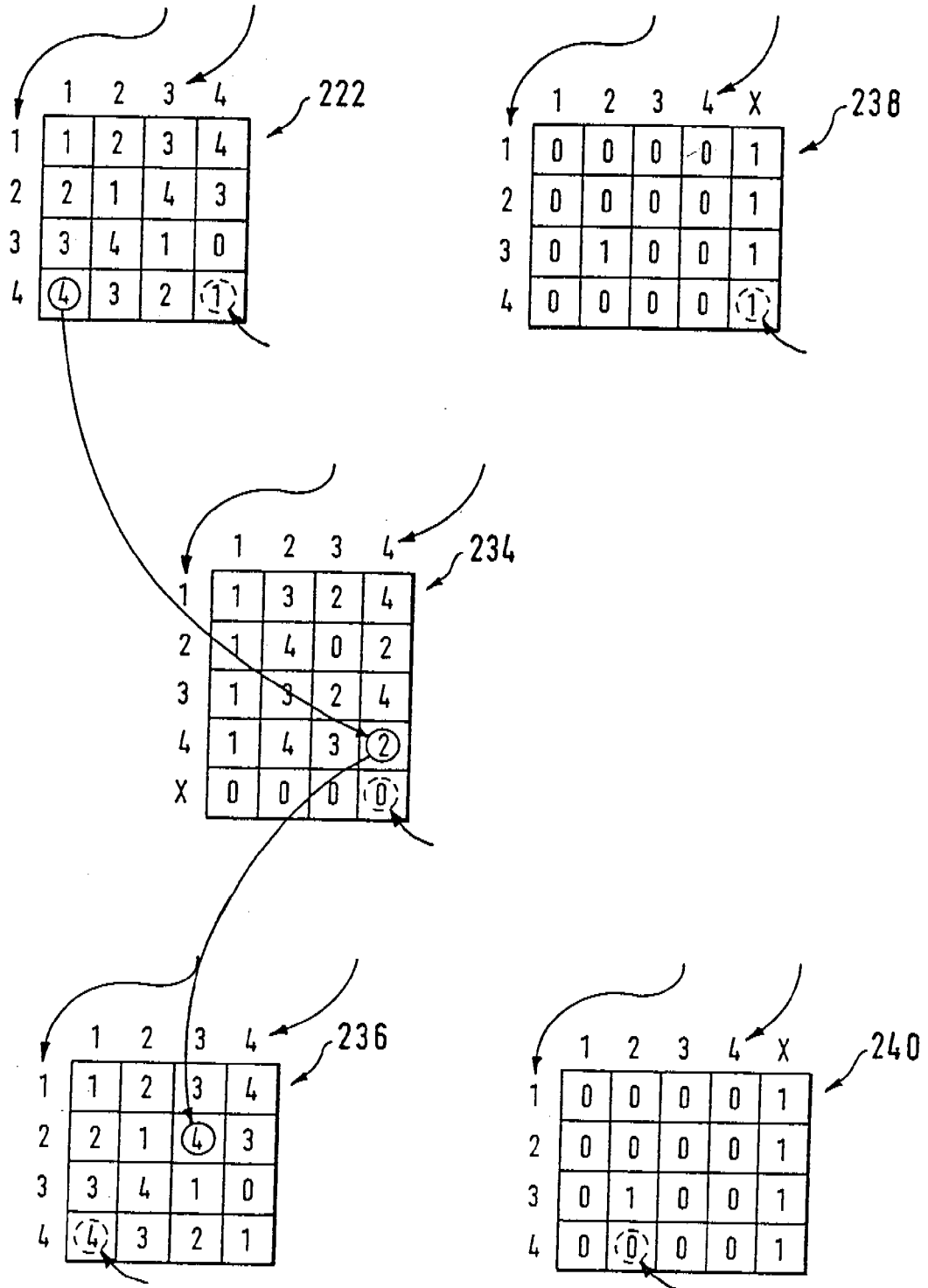


FIG. 22

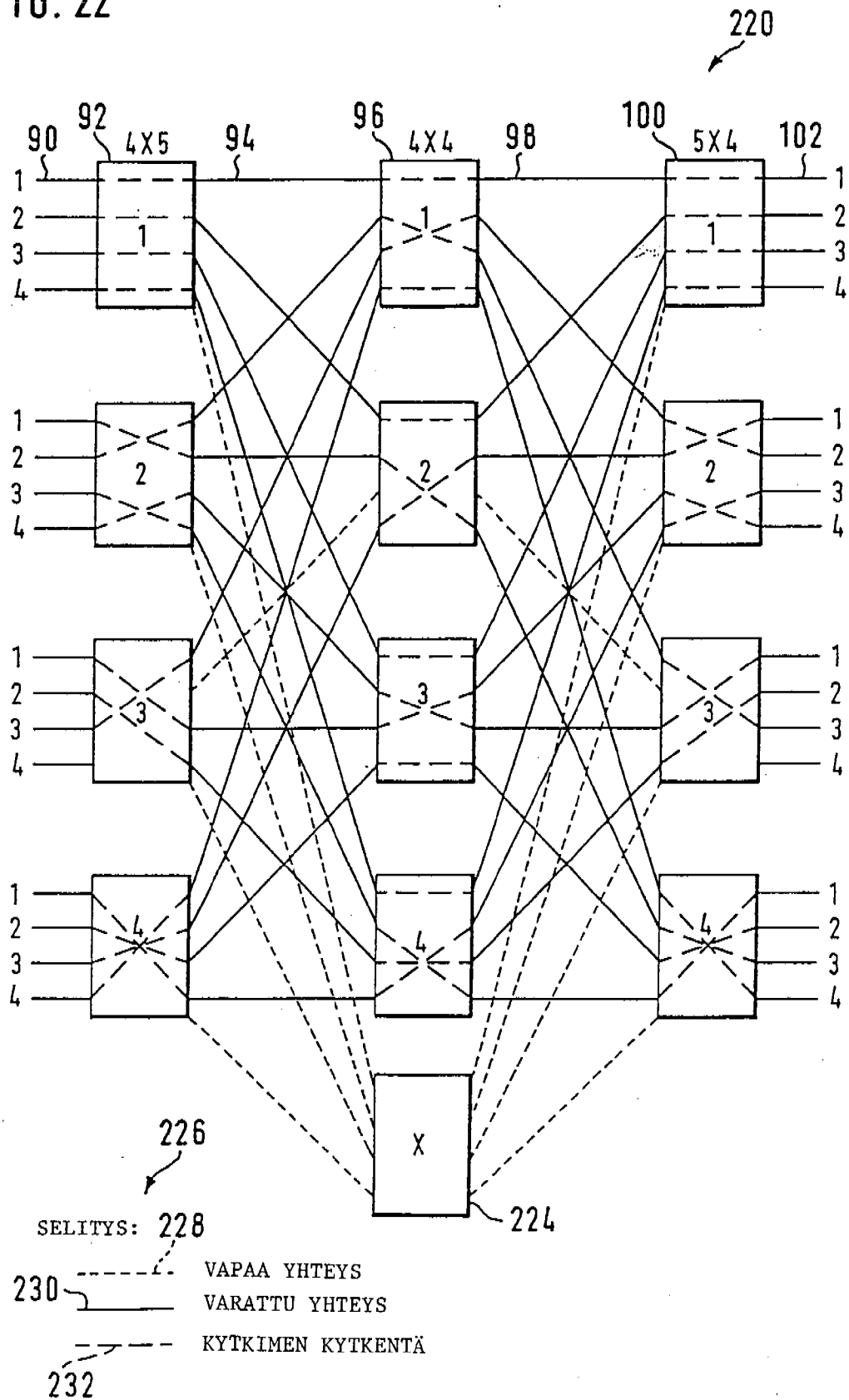


FIG. 23

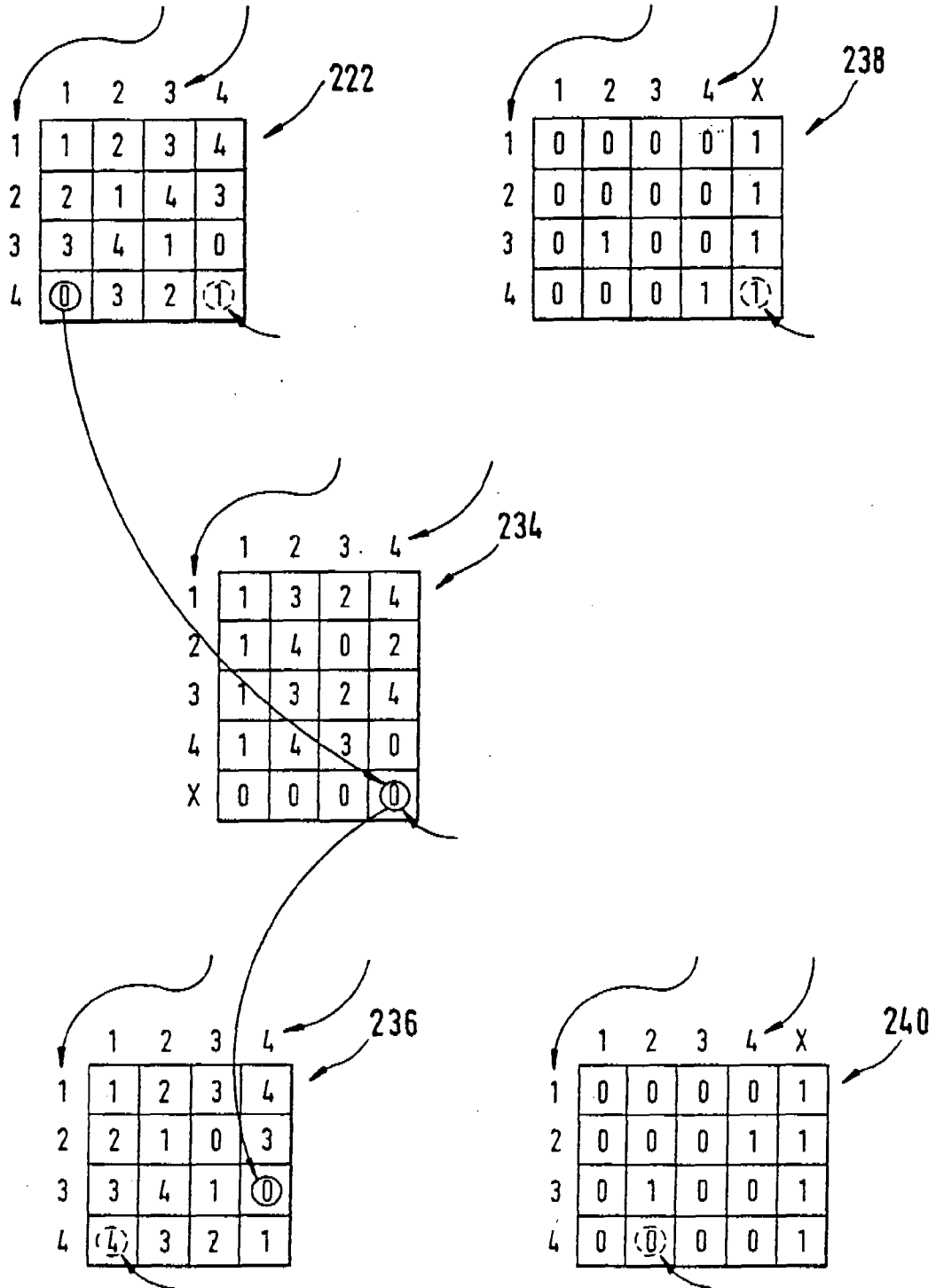


FIG. 24

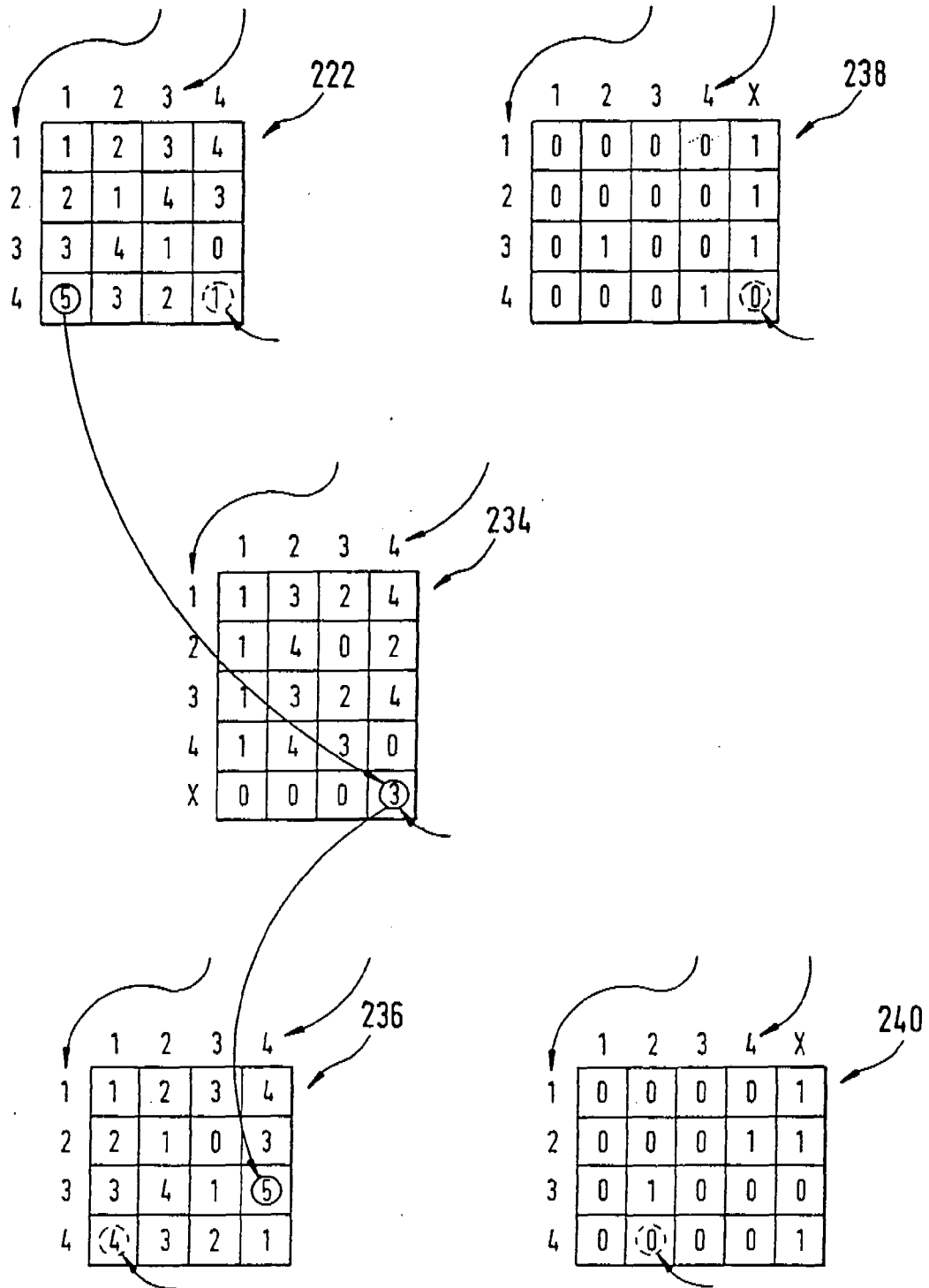


FIG. 25

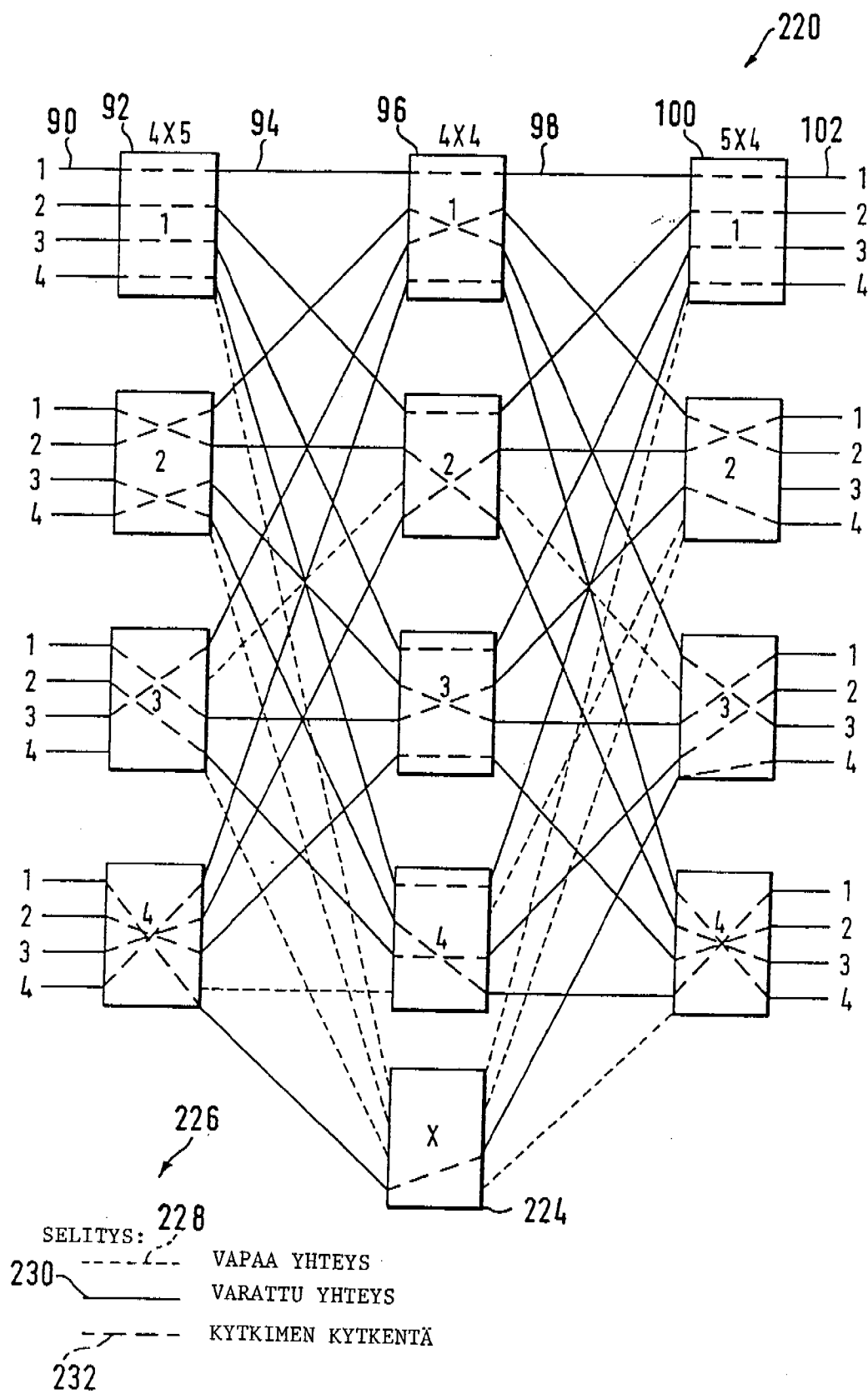


FIG. 26

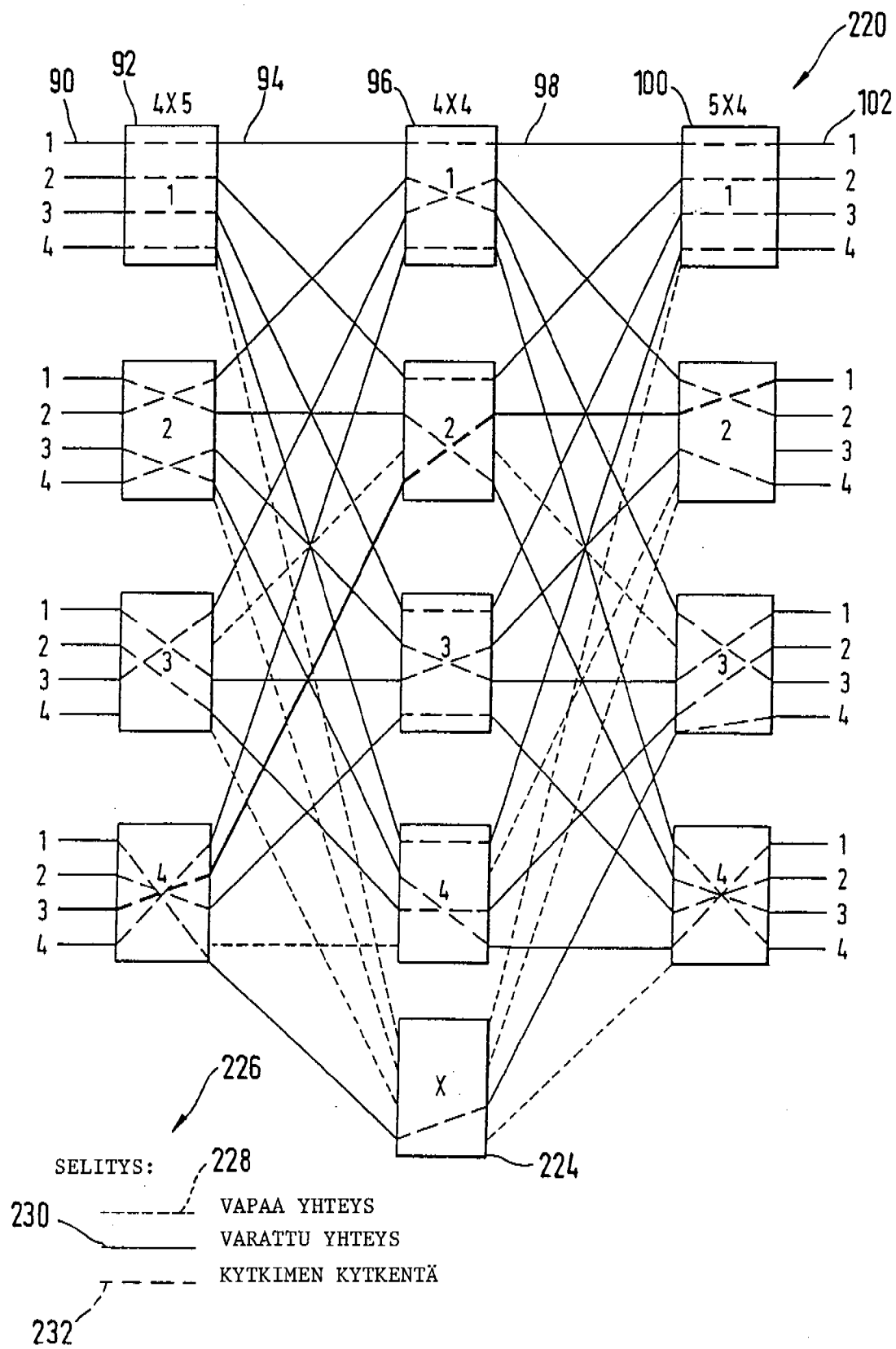


FIG. 28

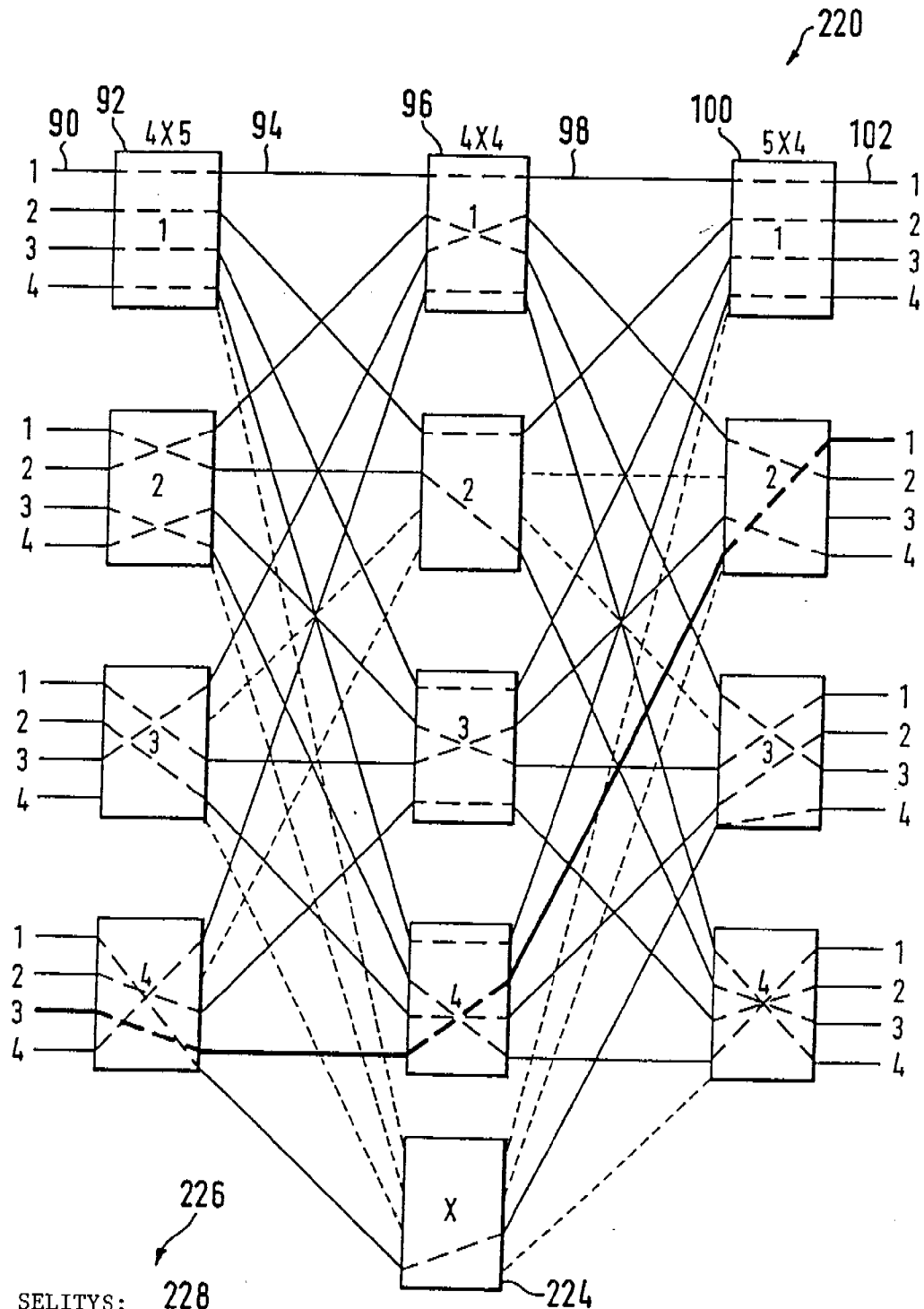


FIG. 29

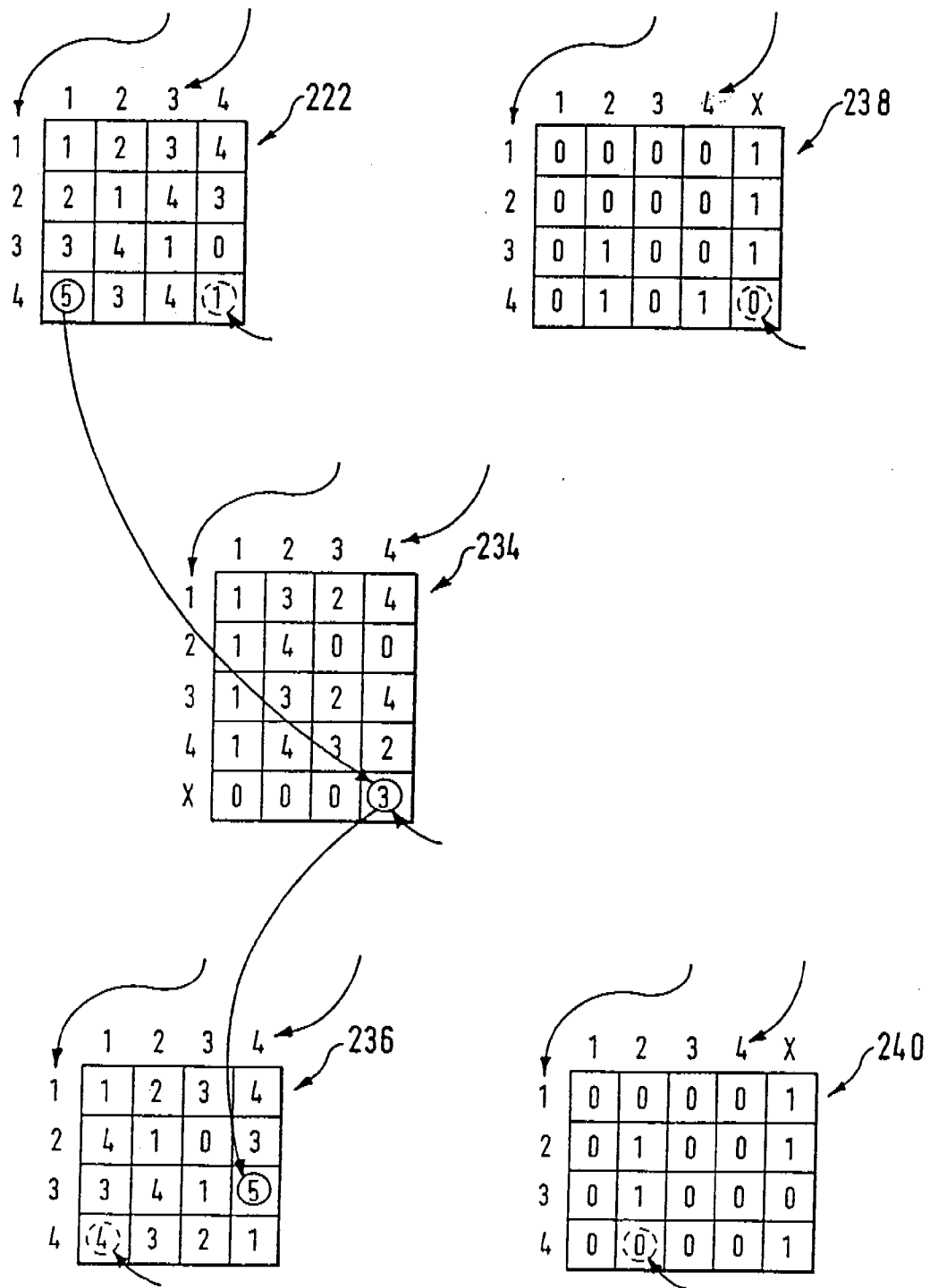


FIG.30

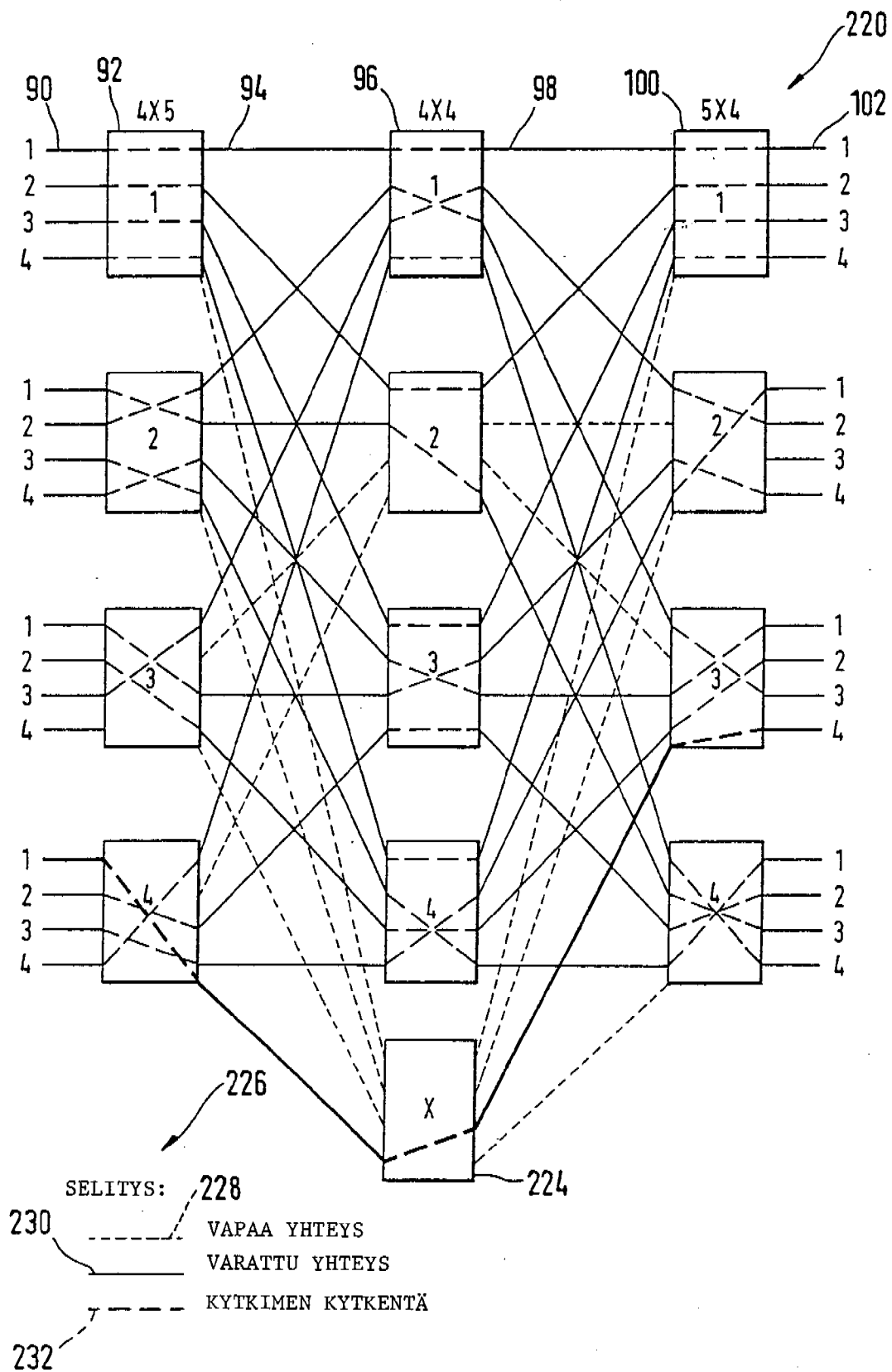
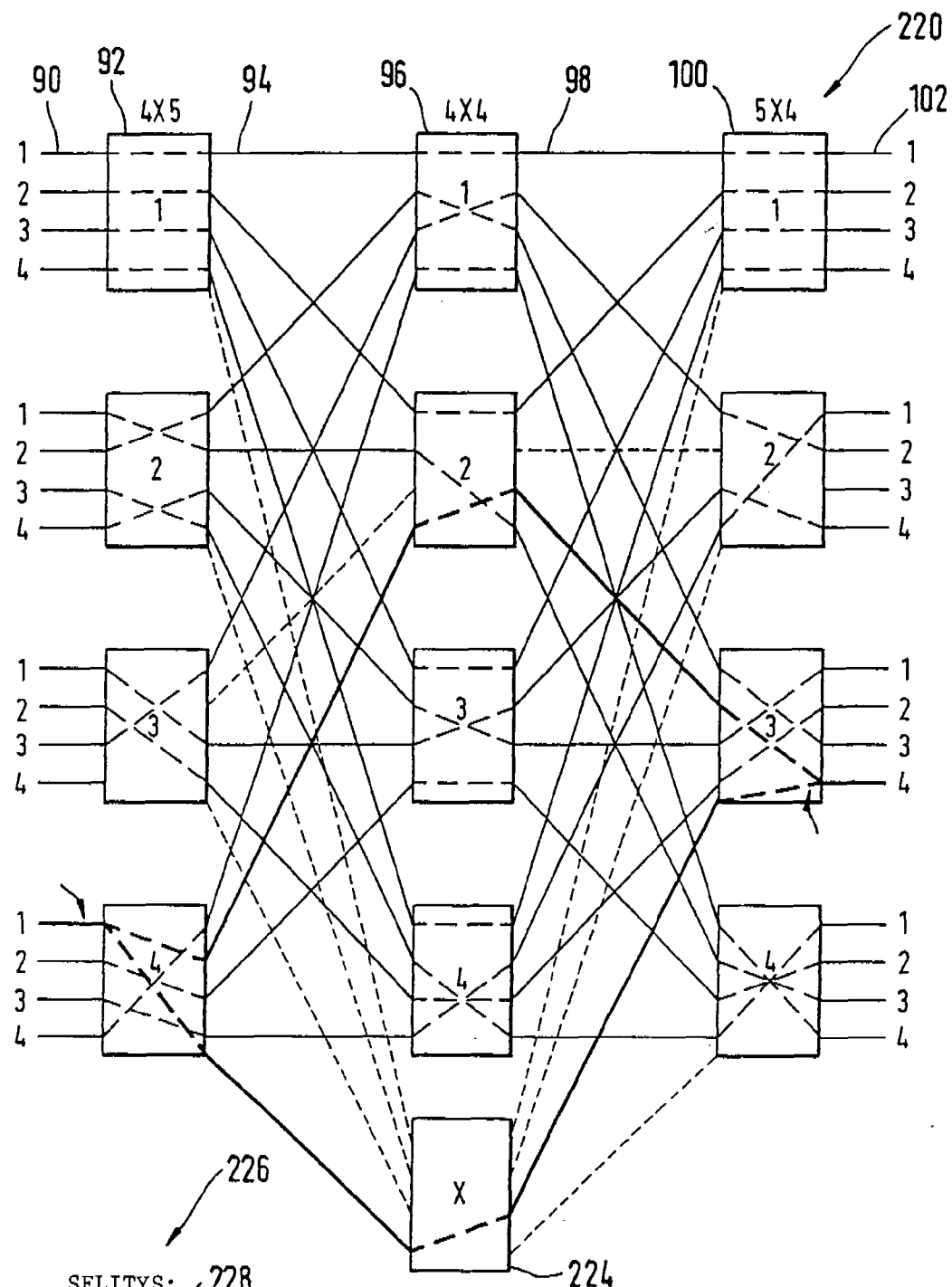
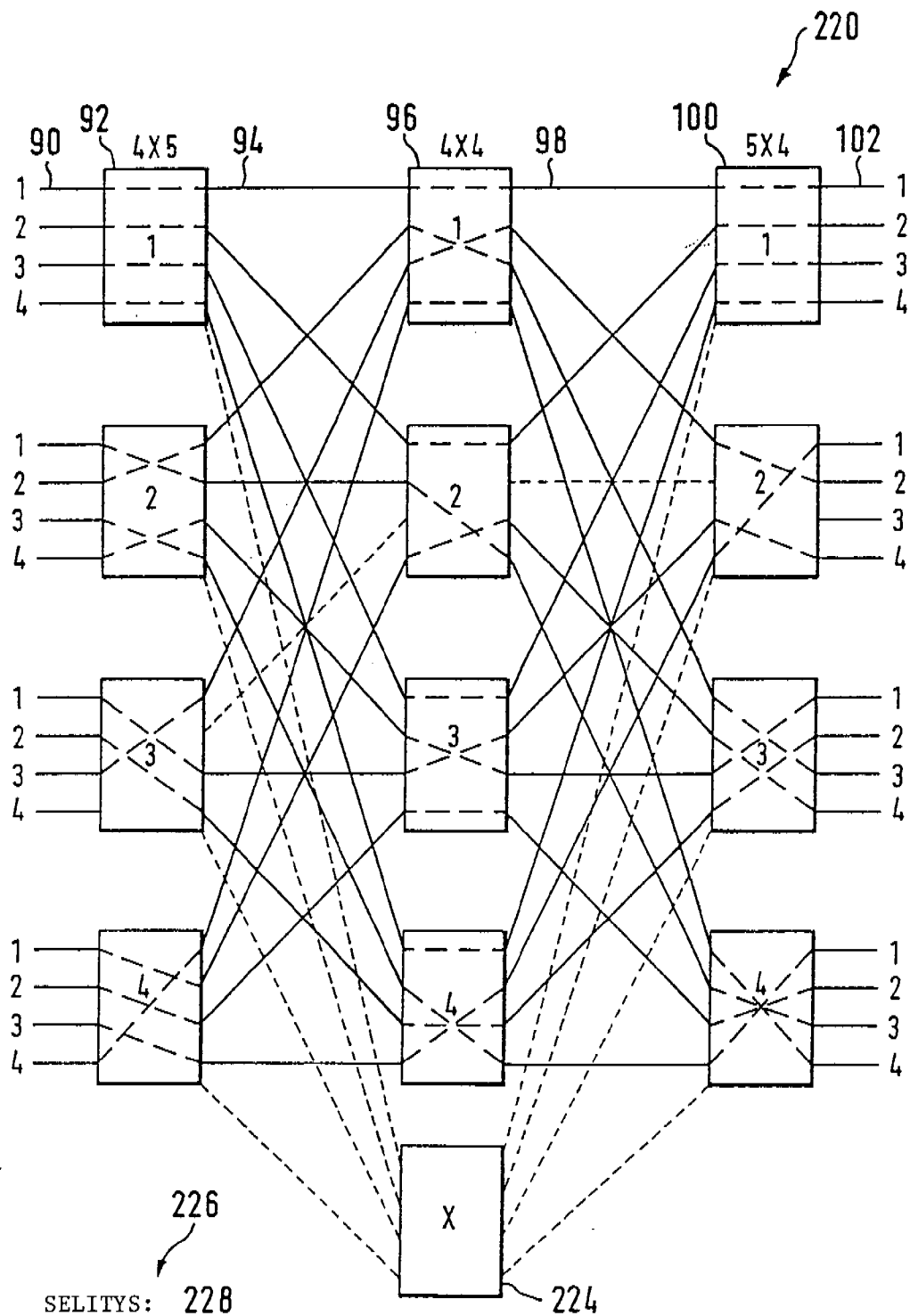


FIG.31



SELITYS: 228
 230 ——— VAPAA YHTEYS
 ——— VARATTU YHTEYS
 -.-.- KYTKIMEN KYTKENTÄ
 232

FIG. 32



SELITYS: 228

230 VAPAA YHTEYS
VARATTU YHTEYS
KYTKIMEN KYTKENTÄ

232

FIG. 33

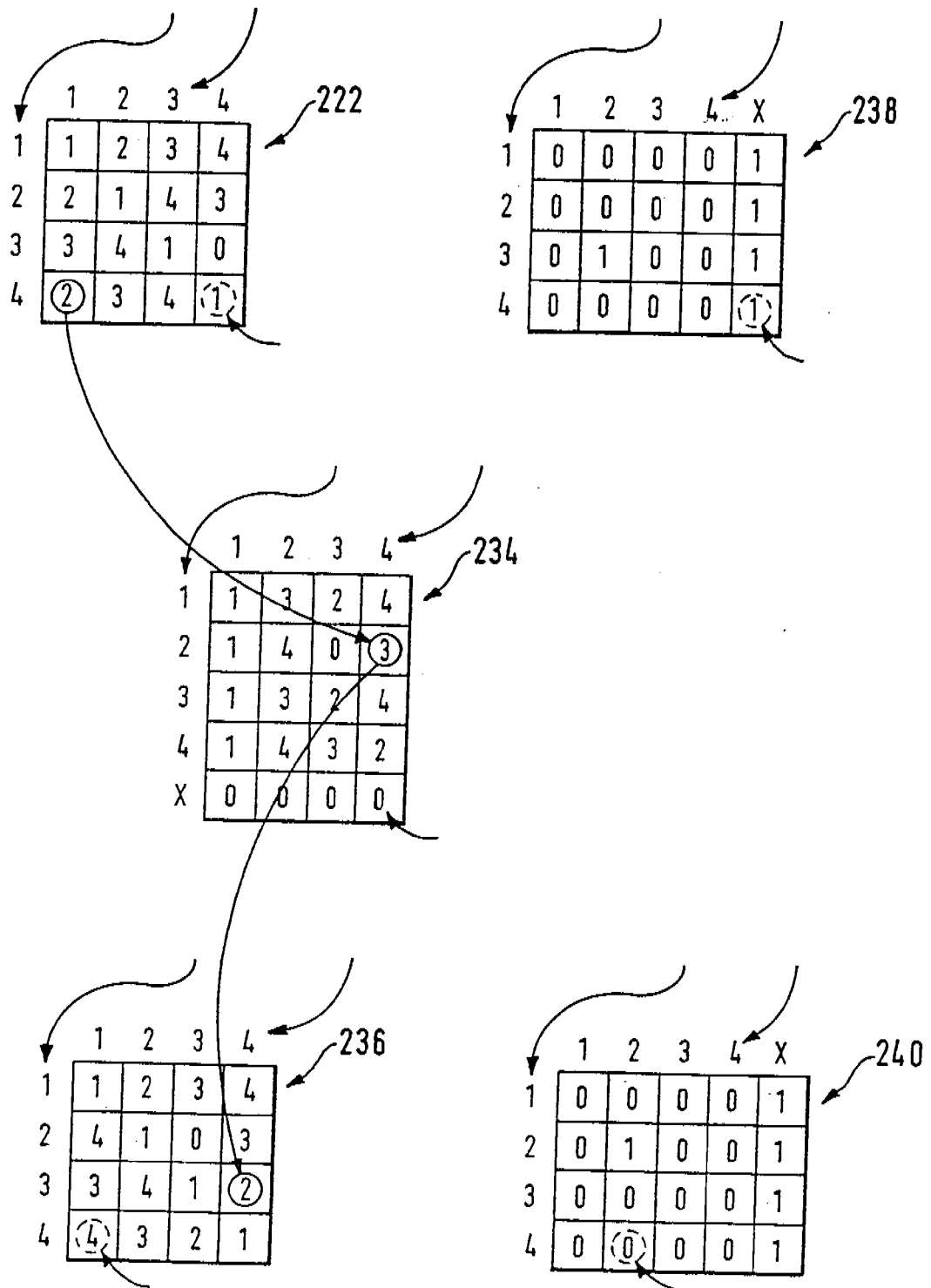
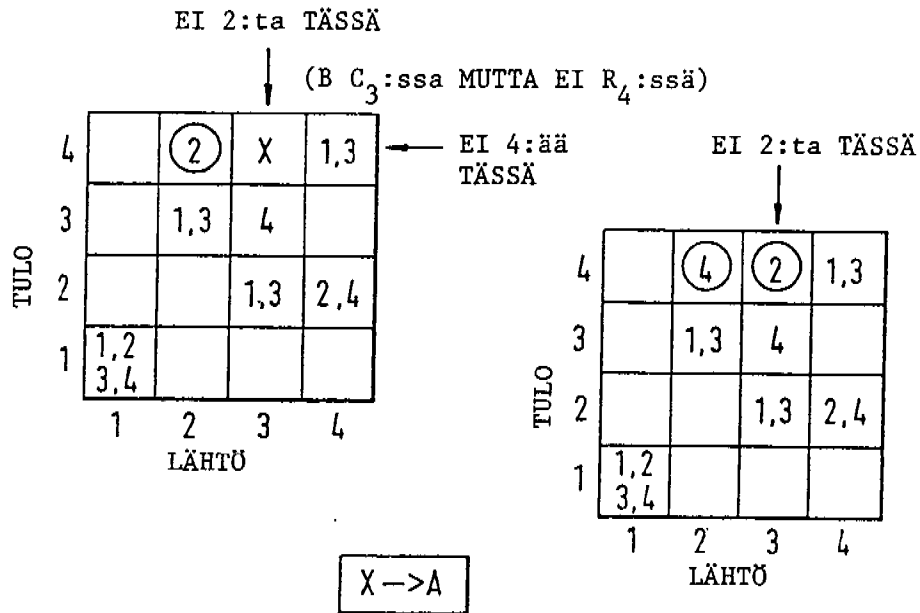


FIG.34

(A R_4 :ssä MUTTA EI C_3 :ssa)



(A R_4 :ssä MUTTA EI C_3 :ssa)

