Universitetet i OSLO

Masterprogrammet for elektronikk og datateknologi

Matrisemultiplikasjon i FPGA

Masteroppgave

Geir Haug Andersen

August 2005

SAMMENDRAG

Denne rapporten omhandler matrisemultiplikasjon i FPGA. Det er laget systemer for multiplikasjon av NxN matriser med størrelse 4x4 og 12x12. Systemene er skrevet i Händel C i dataverktøyet DK Design Suite 3.1 fra Celoxica. Sentralt i systemene er sum av produkter operasjoner. Operasjonene utføres av uavhengige prosesseringsenheter som er realisert på to måter. Den første typen enheter kan brukes uavhengig av matrisestørrelse og bruker lite ressurser, men bruker mange steg på å beregne elementene i produktmatrisen. Den andre typen må tilpasses matrisestørrelsen den skal brukes på og bruker mer ressurser enn den første typen, men beregner elementene produktmatrisen med færre steg.

Systemene er testet for funksjon og om de får plass i FPGAen(Xilinx Virtex-II XC2V6000), videre testing av I/O, optimalisering av ytelse, etc. har måttet utgå pga tidsnød mot slutten av arbeidet med systemene.

Systemene bruker lite av ressursene i FPGAen, i størrelsesorden < 1 % med unntak av multiplikasjonsblokkene som utnyttes opptil 100 %. Systemene er sammenliknet med tidligere designede systemer for matrisemultiplikasjon og bruker mindre av ressursene i FPGAen (for design mot samme FPGA) og oppnår estimert bedre ytelse enn tidligere design for samme matrisestørrelse.

Tilslutt i rapporten skisseres mulige utvidelser og forbedringer av systemene.

Forord

Denne rapporten er resultatet av min masteroppgave ved Masterprogrammet for Elektronikk og Datateknologi, studieretning mikroelektronikk. Arbeidet er gjennomført ved Institutt for Informatikk ved Universitetet i Oslo.

Jeg vil i første rekke takke min hovedveileder Jim Tørresen for all hjelp og støtte. Mine medstudenter for godt arbeidsmiljø og innholdsrike pauser, ingen nevnt ingen glemt. Tidligere overingeniør, nå Scientific Programmer Håvard Kolle Riis for utvist tålmodighet og hjelp med programvare. Brukerne av Händel C forumet på www.celoxica.com fortjener takk for hjelp med store og små problemer rundt bruken av Händel C og DK 3.1

Min kjære Torill skal også ha takk for kjærlig omsorg og støtte i den tiden jeg ikke var opptatt av annet enn matriser og FPGA.

Tilslutt vil jeg takke fru Murphy for hennes lov, som har vært til trøst når ting ikke har gått etter planen.

Geir Haug Andersen, august 2005

INNHOLDSFORTEGNELSE

1 INNLEDNING

1.1 Oversikt over tema i oppgaven

Matrisemultiplikasjon er kjernen i algoritmer innen signal-, bildebehandling og datagrafikk. Å gjøre matrisemultiplikasjon mer effektiv vil kunne bidra til å gjøre disse algoritmene mer effektive. Grunnoperasjonen i matrisemultiplikasjon er sum av produkter(SAP). SAP består av multiplikasjoner og akkumulasjon av produktene. Antall multiplikasjoner som trengs for å beregne et element i produktmatrisen er proporsjonalt med matrisestørrelsen. For store matrisestørrelser vil antall multiplikasjoner bli så stort at bruk av parallellprosessering er aktuelt. FPGA kretser er et alternativ for å utføre matrisemultiplikasjon med parallellprosessering. I denne oppgaven er det utviklet systemer for parallell multiplikasjon av matriser med universelle og tilpassede enheter som utfører SAP operasjonen. Systemene er laget for 4x4 og 12x12 matriser. Matrisemultiplikasjon, FPGA arkitektur og tidligere arbeider innen FPGA og matrisemultiplikasjon presenteres kort før systemene for multiplikasjon av 4x4 og 12x12 matriser beskrives. Deretter blir resultatene for systemene presentert før forslag til videre arbeid og konklusjon.

1.2 Kapitteloversikt

Kapittel 1 Innledning: presenterer temaet i rapporten

Kapittel 2:Bakgrunn for rapporten, matriser, matrisemultiplikasjon. Beskrivelse av FPGA som systemene i kapittel 3 er utviklet for. Presentasjon av programmeringsspråket Händel C som systemene er skrevet i. Tidligere arbeid innen matrisemultiplikasjon på FPGA.

Kapittel 3: Systemene som er laget for rapporten med beskrivelse av felles løsninger.

Kapittel 4: Resultater og diskusjon av resultatene for systemene i kapittel 3

Kapittel 5: Videre arbeid med systemene i kapittel 3 og forslag til utvidelser og forbedringer

Kapittel 6: Konklusjon

2 BAKGRUNN

2.1 Matriser

Definisjon

En matrise er et rektangulært skjema eller tabell som inneholder tall eller symboler. Eksempel: M i Likning 2-1 er en matrise. M har to rader og tre kolonner og kalles en 2x3 matrise. 2x3 kalles også matrisens størrelse[1]. I denne rapporten behandles kun NxN matriser

$$\begin{bmatrix} a\ b\ c \\ d\ e\ f \end{bmatrix}$$

Likning 2-1Eksempel på matrise

Matrisenotasjon

Elementene i matriser identifiseres ut fra posisjonen i matrisen. Eksempel: om elementet i andre rad og tredje kolonne i 3x3 matrisen L i Likning 2-2 brukes notasjonen l_{23}

$$L = \begin{bmatrix} l_{11} & l_{12} & l_{13} \\ l_{21} & l_{22} & l_{23} \\ l_{31} & l_{32} & l_{33} \end{bmatrix}$$

Likning 2-2: 3x3 matrisen L

I NxN matriser er antall elementer i rad/kolonne lik N.

2.2 Matrisemultiplikasjon

For at to matriser skal kunne multipliseres må produktmatrisen være definert. Antall kolonner i første matrise må være likt antall rader i andre matrise. Antall rader i første matrise og antall kolonner i andre matrise gir dimensjonene for produktmatrisen [1].

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}, B = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}, R = A \times B = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

Likning 2-3: Matrisemultiplikasjon

I denne rapporten behandles bare NxN matriser så R er alltid definert da antall rader/kolonner er likt i begge matrisene, se Likning 2-3.

Rad-kolonne regelen for matrise multiplikasjon

Hvis R er definert er elementet i i'te rad og j'te kolonne i R (r_{ij}) summen av produktene av de korresponderende elementene fra rad *i* i matrise A og kolonne *j* i matrise B [1], se Likning 2-4.

$$r_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj}$$

Likning 2-4: matriseelement som sum av produkter

Så produktmatrisen kan representeres som i Figur 2-4

$$R = \begin{bmatrix} \sum_{k=1}^{n} a_{ik} \cdot b_{kj} & \sum_{k=1}^{n} a_{ik} \cdot b_{kj} & \sum_{k=1}^{n} a_{ik} \cdot b_{kj} \\ \sum_{k=1}^{n} a_{ik} \cdot b_{kj} & \sum_{k=1}^{n} a_{ik} \cdot b_{kj} & \sum_{k=1}^{n} a_{ik} \cdot b_{kj} \\ \sum_{k=1}^{n} a_{ik} \cdot b_{kj} & \sum_{k=1}^{n} a_{ik} \cdot b_{kj} & \sum_{k=1}^{n} a_{ik} \cdot b_{kj} \end{bmatrix}$$

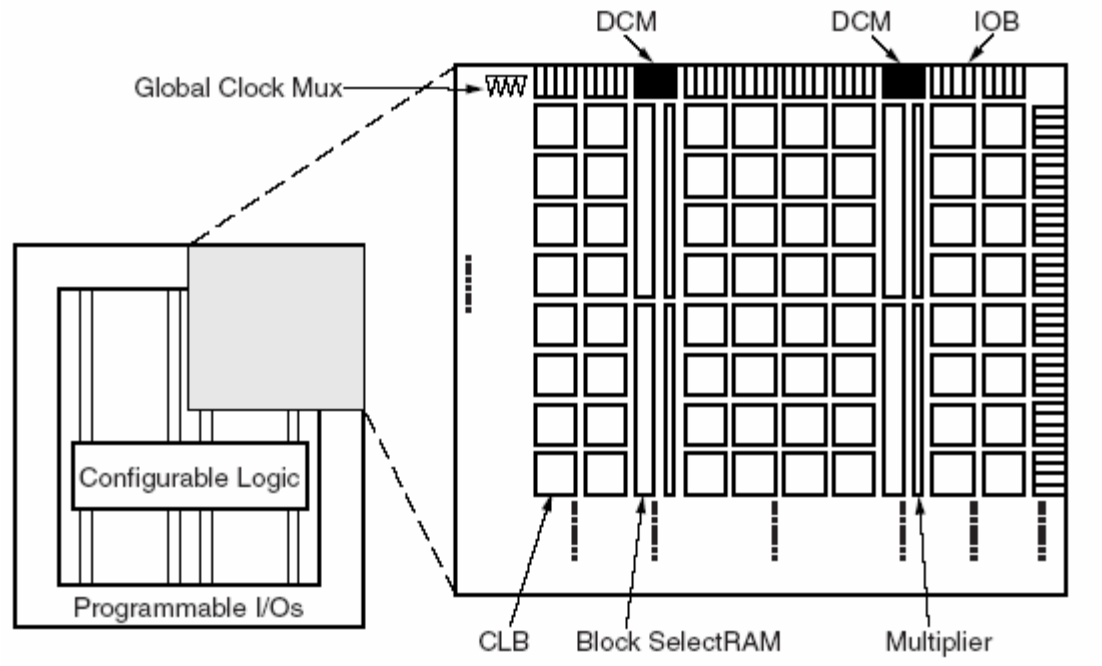
Likning 2-5: Elementene i produktmatrisen R som summer av produkter

Anvendelser av matrisemultiplikasjon

Matriser og lineær algebra anvendes i matematikken for løsning av lineære likninger og systemer av lineære likninger. Matrisemultiplikasjon anvendes innen datagrafikk for modellering av 3D objekter og deres bevegelser, innen digital signalbehandling til signal transformasjon og filtrering (FIR/IIR) og innen bildebehandling til kompresjon (jpeg/mpg) og filtrering.

2.3 Field Programmable Gate Array

Field Programmable Gate Array (FPGA) er rekonfigurerbare digitale kretser. FPGA kretser inneholder en kjerne av logiske celler og kommunikasjonskanaler i et rutenett[2]. Kommunikasjonskanalene knytter sammen de logiske cellene og forbinder dem med IO blokkene som ligger langs kanten av kretsen, se Figur 2-1. Innholdet i FPGAen (funksjonen) lagres i SRAM og kan endres et uendelig antall ganger etter at kretsen er produsert ved å endre innholdet i SRAMen. Funksjonen er begrenset til kombinatorisk og digital logikk.



Figur 2-1: Innholdet i en Xilinx Virtex-II FPGA[3]

Innholdet i en FPGA, eksempel, se Figur 2-1 kan deles inn i følgende kategorier

1 I/O blokker (Programmable I/O)

2 Blokk RAMer (Block Select RAM)

3 Logiske celler (CLB)

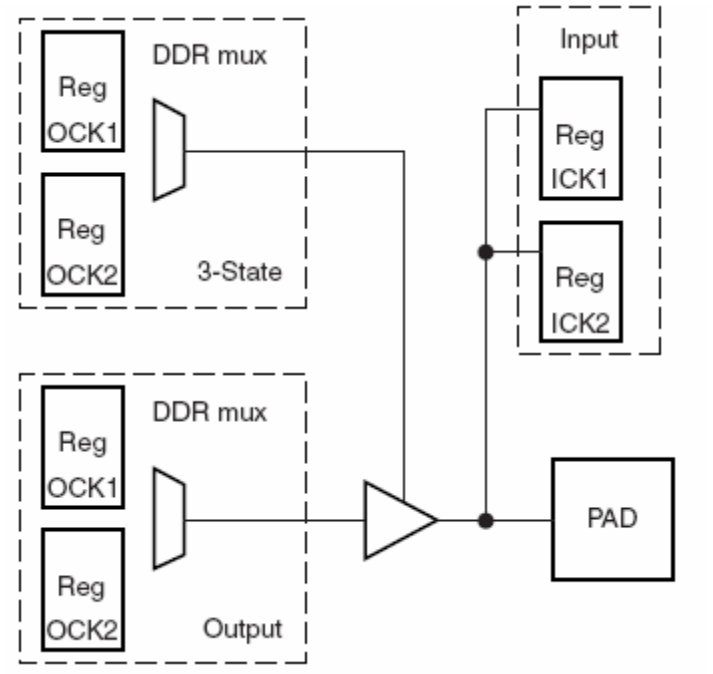
Xilinx Virtex-II FPGAer inneholder i tillegg interne multiplikasjonskretser (Multiplier) og digitale klokke multipleksere (DCM).

I fortsettelsen vil de logiske ressursene nevnt over beskrives med innholdet i Xilinx Virtex-2 XC2V6000 som eksempel.

Logiske celler	8 448
BRAM (Kbit)	2 592
18bit Multiplikasjonsblokker	144
Distribuert RAM (Kbit)	1 056
IO pinner (tilgjengelig for bruker)	824

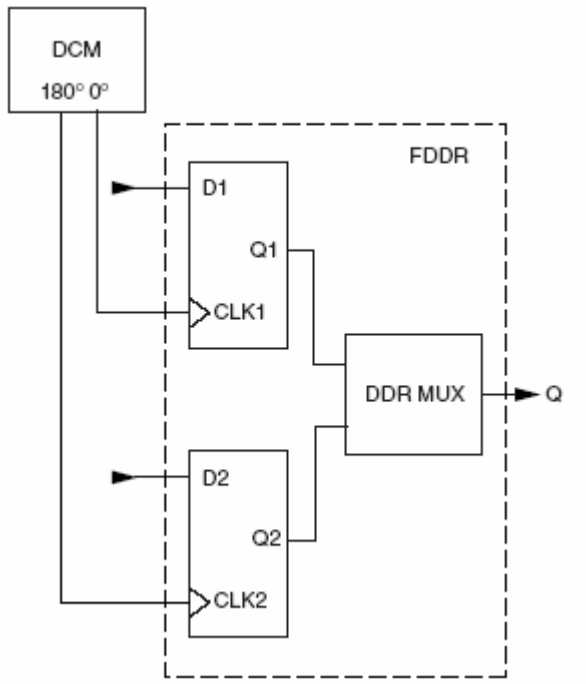
2.3.1 I/O blokker

En Virtex-II XC2V6000 FPGA i FF1152 pakke har 824 I/O pinner tilgjengelig for bruker I/O. Hver I/O pinne er koblet til en I/O blokk, se Figur 2-2



Figur 2-2: Virtex-II IO blokk[3]

I/O blokkene i FPGAen har støtte for 19 I/O standarder, både ”single ended” som PCI og PCI-X og differensielle som LVDS og BLVDS.

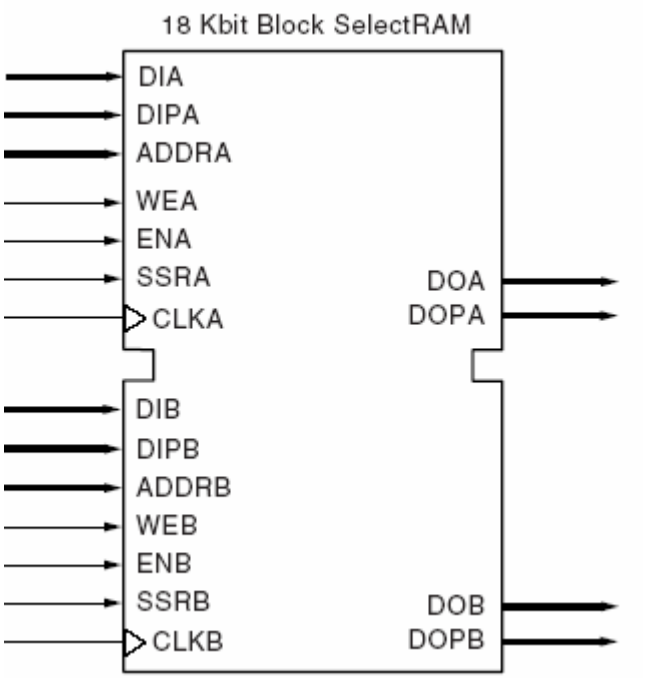


Figur 2-3: Dobbel datarate registre i Virtex-II I/O blokk[3]

Dobbel datarate oppnås ved at registrene på hver transmisjonslinje klokkes med klokker fra to forskjellige klokkenett. De to klokkesignalene genereres av en DCM, se Figur 2-7 og må være 180˚ faseforskjøvet i forhold til hverandre som vist i Figur 2-3.

2.3.2 Blokk RAM

XC2V6000 inneholder 144 18Kbit Block Select RAMer på til sammen 2654208 bit. Hver RAM har to funksjonelt identiske, uavhengig kontrollert og klokkede synkrone porter, se Figur 2-4. Portene har tilgang til et felles 18Kbit lagringsområde[3].



Figur 2-4: Xilinx Virtex-II 18 Kbit Blokk RAM i toports modus[3]

Hvis begge portene er konfigurert likt er hele RAMen tilgjengelig fra begge portene. Er portene konfigurert ulikt er hele 18Kbit området tilgjengelig fra den ene porten, mens den andre porten har tilgang til et underområde av RAMen tilsvarende 16Kbit, se Tabell 2.1

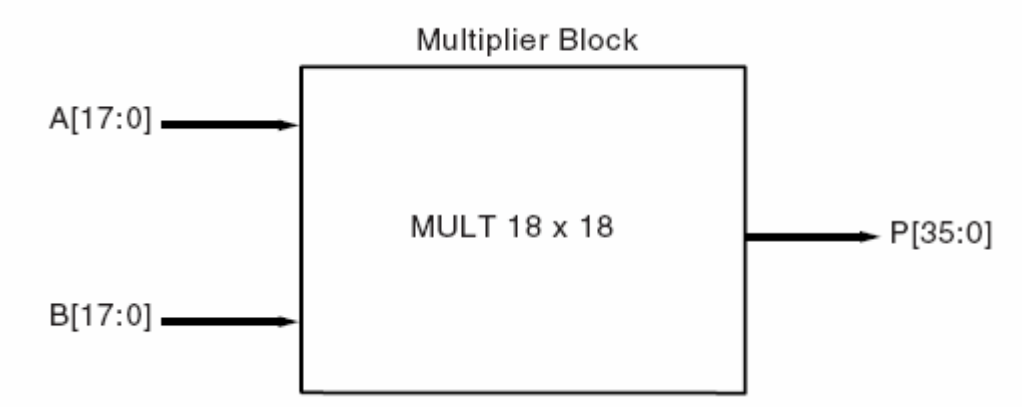
port A	16K x 1	16K x 1	16K x 1	16K x 1	16K x 1	16K x 1
port B	16K x 1	8K x 2	4K x 4	2K x 9	1K x 18	512 x 36
port A	8K x 2	8K x 2	8K x 2	8K x 2	8K x 2	
port B	8K x 2	4K x 4	2K x 9	1K x 18	512 x 36	
port A	4K x 4	4K x 4	4K x 4	4K x 4		
port B	4K x 4	2K x 9	1K x 18	512 x 36		
port A	2K x 9	2K x 9	2K x 9			
port B	2K x 9	1K x 18	512 x 36			
port A	1K x 18	1K x 18				
port B	1K x 18	512 x 36				
port A	512 x 36					
port B	512 x 36					

Tabell 2-1:toports konfigurasjoner for 18Kbit blokk RAM [3]

.

2.3.3 18 bit Multiplikasjonsblokk

FPGAen inneholder 144 18bit multiplikasjonsblokker. Multiplikasjonsblokkene er 18bit 2's komplement multiplikasjonskretser. Hver blokk har to 18bits innganger og en 36bit utgang, se Figur 2-5.



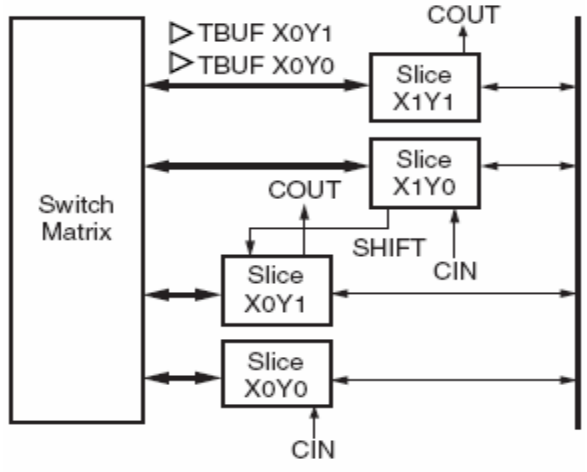
Figur 2-5:Virtex-II 18bit multiplikasjonsblokk[3]

Multiplikasjonsblokkene og blokk RAMene i kretsen deler forbindelse (interconnect) mot resten av kretsen. Denne forbindelsen er optimalisert med tanke på at RAMene skal forsyne multiplikasjonsblokkene, men RAM og multiplikasjonsblokker kan brukes uavhengig av hverandre og samtidig.

2.3.4 Konfigurerbar Logisk Blokk (CLB)

CLBene brukes til å realisere kombinatorisk og digital logikk i FPGAen. Hver CLB er knyttet til det interne rutenettet i FPGAen. En CLB består av fire slices som er organisert i to kolonner, se Figur 2-6. En CLB inneholder eller kan implementere følgende logiske ressurser/funksjoner.

Slice	4
LUTer	8
vipper	8
mentekjede	2
Sum av produkter (SOP) kjeder	2
Distribuert Select RAM (bit)	128
Skiftregister (bit)	128
MULT_AND	8



Figur 2-6: Virtex-II CLB [3]

SLICE

En slice inneholder, se Figur 2-11.

2 Funksjonsgeneratorer (F og G) (*)

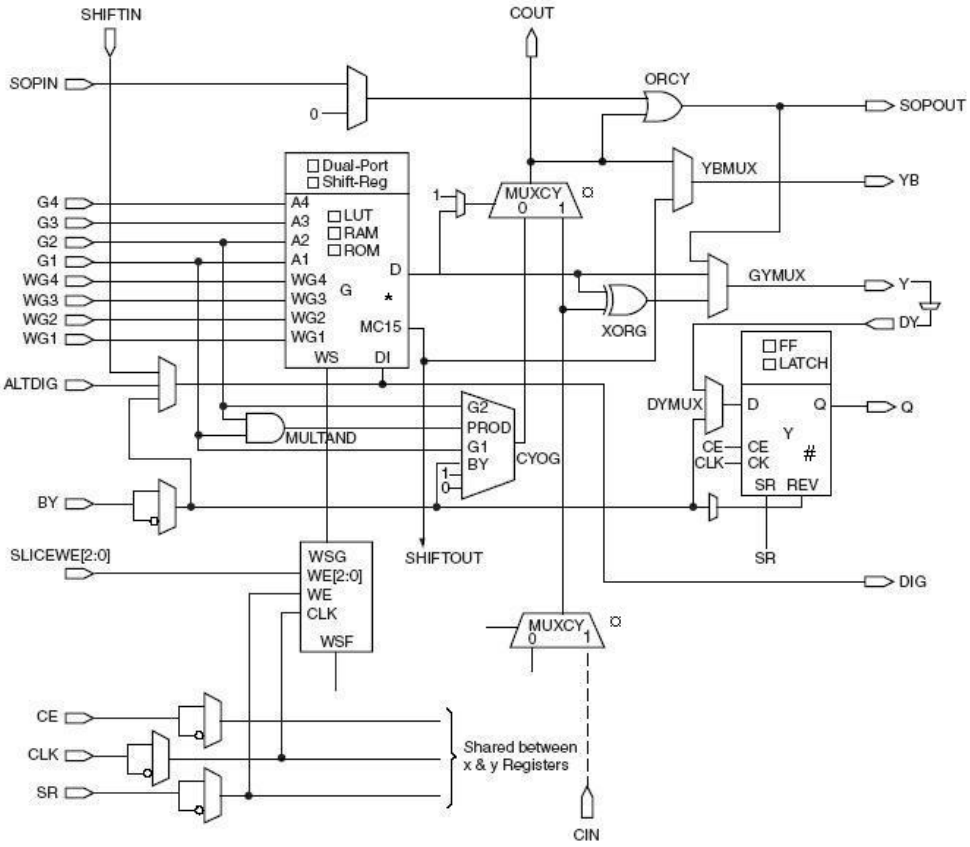
2 Lagringselementer (#)

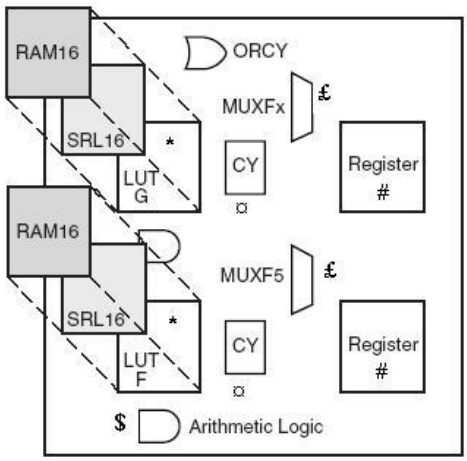
Aritmetisk logiske porter ($)

Multipleksere (£)

Carry logikk (¤)

Lagringselementene kan brukes som flankestyrte D-vipper eller nivåsensitive latcher. Funksjonsgeneratorene er fire inngangs Look Up Tables (LUT) og kan brukes som 16bit RAM eller skifteregister i tillegg til LUT.





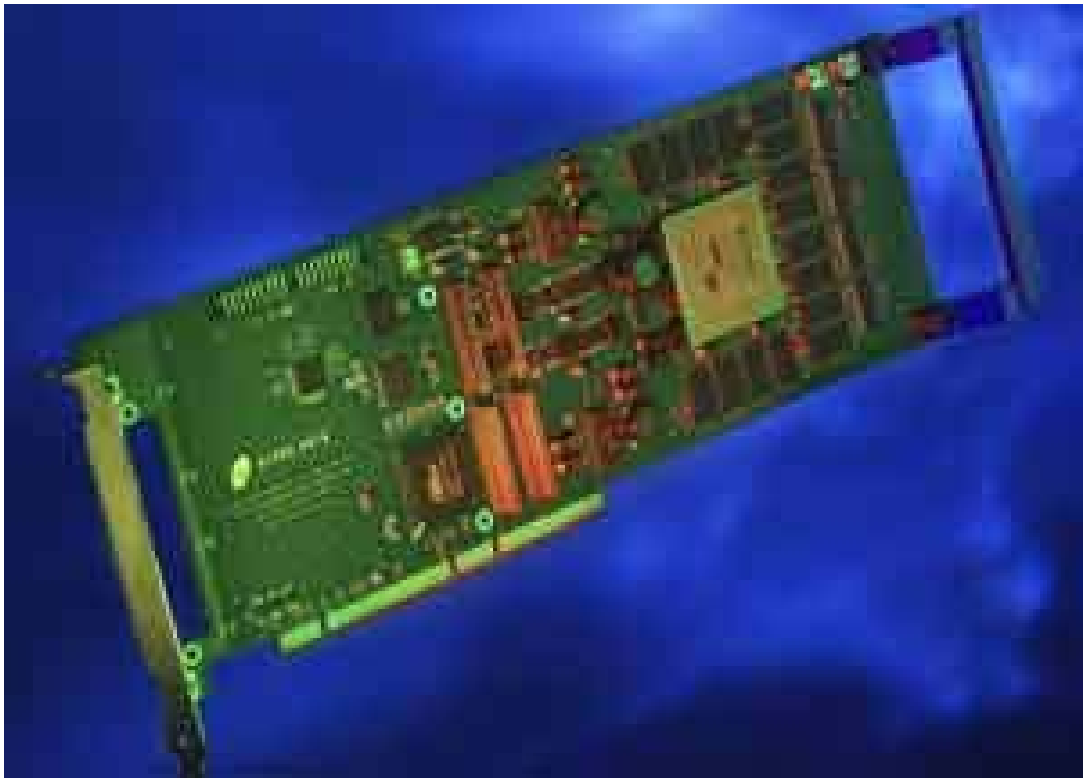
Figur 2-7: Virtex-II slice [3], øvre del av figuren viser øvre halvdel av slicen i nedre del

LOOK UP TABLE (LUT)

Funksjonsgeneratorene i slicene er fire inngangs Look Up Tables (LUT). Disse kan benyttes som 16bit RAM eller 16bit skiftregister i tillegg til LUT, men bare som en av de tre av gangen. En LUT kan realisere boolske funksjoner med fire innganger. Ved å bruke en av multipekserne i slicen, se Figur 2-7, kan funksjoner med opptil åtte innganger realiseres. Ved å kombinere to eller flere slicer kan en CLB realisere funksjoner med 32 innganger. CLBer kan kombineres for å lage funksjoner med mer enn 32 innganger.

2.3.5 ADM-XRC-II

Xilinx Virtex-II XC2V6000 FPGAen sitter på et RC2000 FPGAkort fra Celoxica, kortet er produsert av Alphadata (kalles ADM-XRC-II av Alphadata), se Figur 2-8. Kortet er et PMC (PCI Mezzanine Card) kort som foruten FPGAen har 6 minnebanker for ZBT (”Zero Bus Turnaround”) RAM og opptil 146 I/O pinner for bruker I/O med en maksimal overføringsrate på 40 Gb/S [4]. Kortet kan utvides med inntil 512 MB DDR(double data rate) RAM.



Figur 2-8: RC2000 PMC FPGAkort

2.4 Händel C

Systemene i denne rapporten er laget med dataverktøyet DK 3.1 fra Celoxica. DK 3.1 er basert på programmeringsspråket Händel C.

Händel C er basert på C og bruker C syntaks[5], men er utvidet med instruksjoner for parallell kjøring av programgrener og kommunikasjon mellom parallelle programgrener. Händel C inneholder altså instruksjoner for å spesifisere maskinvare, så prosessororienterte instruksjoner som pekere og flyttallsaritmetikk er utelatt.

Parallell kjøring spesifiseres ved å bruke instruksjonen *par*. For å kjøre to programgrener parallelt brukes *par* som i eksempelet under.

Eksempel: parallelle programgrener

```
Par
{
Programgren1;
Programgren2;
}
```

Om Händel C sammenliknes med VHDL som er et svært utbredt språk for å konstruere hardmaskinvare fremkommer det at de to språkene har forskjellig fokus. Händel C skiller seg fra tradisjonelle språk for maskinvare spesifikasjon som VHDL og verilog ved at det er laget for å realisere algoritmer i maskinvare uten å ha behov for detaljert kunnskap om maskinvaren, mens VHDL fokuserer på å spesifisere digital elektronikk på portnivå. VHDL er språket for hardware ingeniøren som arbeider med spesifikasjon av enkelt porter og forsinkelsen gjennom dem. Händel C er laget for programmerere uten detaljkunnskaper om hardware[6].

2.5 Tidligere arbeid

Matriseoperasjoner og særlig matrisemultiplikasjon har en rekke anvendelser i algoritmer innen datagrafikk, signal-, og bildebehandling. Eksempler er Diskret Cosinus Transform (DCT) som brukes i bildeformatet jpeg og videoformatet mpg[7] og Diskret Fourier Transform (DFT) som brukes i signalanalyse og datakompresjon[8]. For å oppnå bedre ytelse for systemer med matrisemultiplikasjon som basis er det arbeidet med å lage effektive løsninger for matrisemultiplikasjon. FPGA kretser kan benyttes for å lage slike løsninger på grunn av: store logikk ressurser som kan anvendes i parallell, stor lagringskapasitet.

For å oppnå økt ytelse for matrisemultiplikasjonssystemer kan flere metoder benyttes.

- Lage mer effektive multiplikasjonsenheter.
- Å organisere matrisemultiplikasjonen mer effektivt.

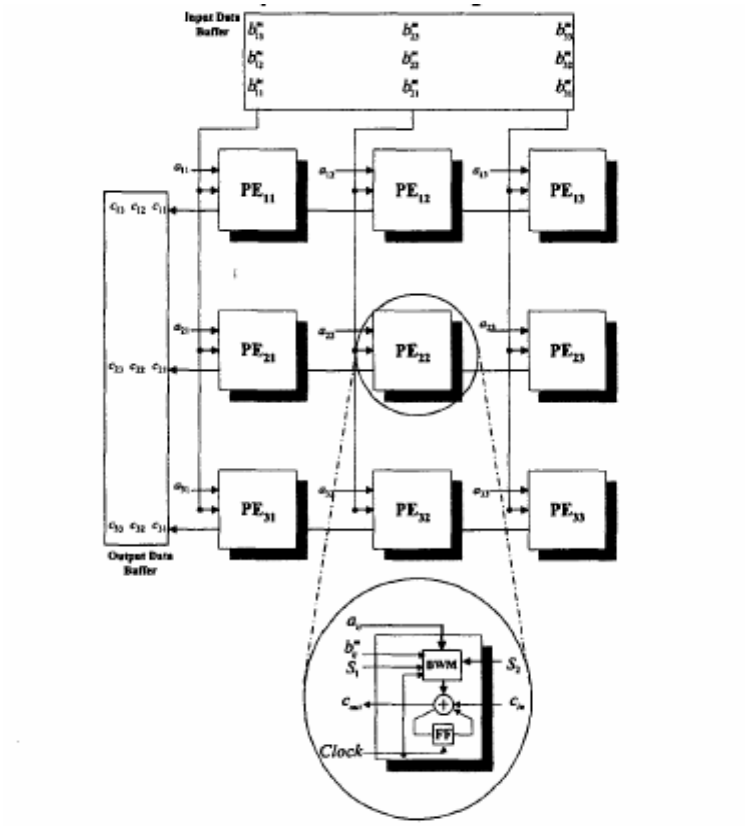
Arbeidet er ofte rettet mot en anvendelse av matrisemultiplikasjon og kombinerer begge metodene ved både å lage effektive multiplikasjonsenheter og å organisere bruken av dem på en effektiv måte.

For å lage effektive multiplikasjonsenheter kan enhetene lages fra bunnen av (i LUT) eller å utnytte interne multiplikasjonsblokker i FPGA til å lage multiplikasjonsenheter.

Organisering av multiplikasjonen er gjort ved å lage systoliske arkitekturer [9]. Systoliske arkitekturer er arkitekturer hvor hver del av arkitekturen gjør en del av beregningsoppgaven og sender resultatet videre i arkitekturen hvor det brukes av neste del i arkitekturen. Arkitekturene organiseres etter algoritmer, eksempel på en slik algoritme er Baugh-Wooley's algoritme [10].

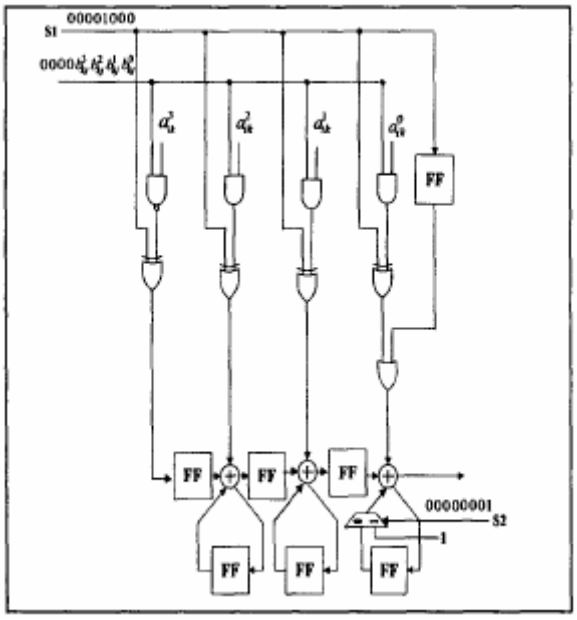
Multiplikasjonen kan også organiseres ved å lage systemer som består av enheter som arbeider parallelt, men i motsetning til enhetene i en systolisk arkitektur utføres hele multiplikasjonen i en enhet uavhengig av de andre enhetene, et eksempel er distribuert aritmetikk i [11].

I [12] presenterer Amira et. al en systolisk arkitektur[9] for multiplikasjon av NxN matriser, se Figur 2-9. Det oppgis ikke i artikkelen om arkitekturen er skrevet i VHDL eller laget med annet verktøy. Arkitekturen består av multiplikasjonsenheter laget fra bunnen av og brukes i en systolisk arkitektur.



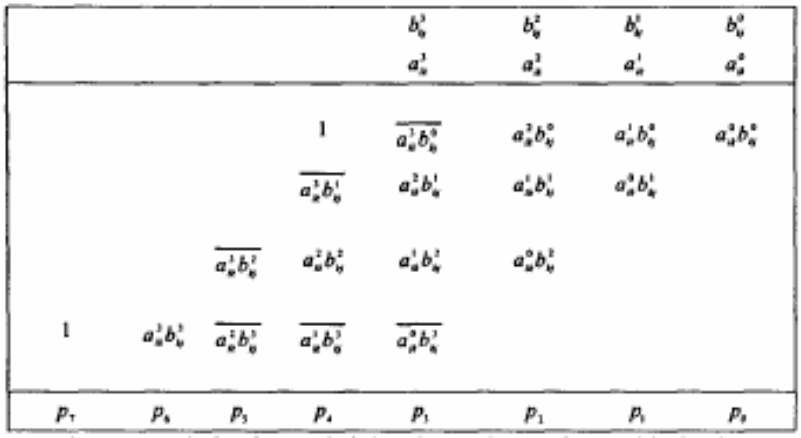
Figur 2-9: systolisk arkitektur for multiplikasjon av 3x3 matriser[12]

Arkitekturen er testet for N =2,3,4,5 og 8, alle med åtte bit matriseelementer. Arkitekturen består av N^2 prosesseringselementer (PE) og buffere for inn-/utlesing av data. PEene består av en multiplikasjonsenhet laget for Baugh-Wooley's algoritme, se Figur 2-10, en vippe for lagring av mente og en addisjonsenhet (fulladder) for addisjon av delprodukter, se Figur 2-9.



Figur 2-10: Baugh-Wooley multiplikasjonsenhet[12]

Multiplikasjonene er organisert etter Baugh-Wooleys algoritme og systemet inneholder PEer laget for bruk med Baugh-Wooley's algoritme. PEene inneholder en Baugh-Wooley multiplikasjonsenhet som multipliserer matriseelementene bitvis, se Figur 2-11.



Figur 2-11: Bit nivå multiplikasjon etter Baugh-Wooley's algoritme på tabellform[12]

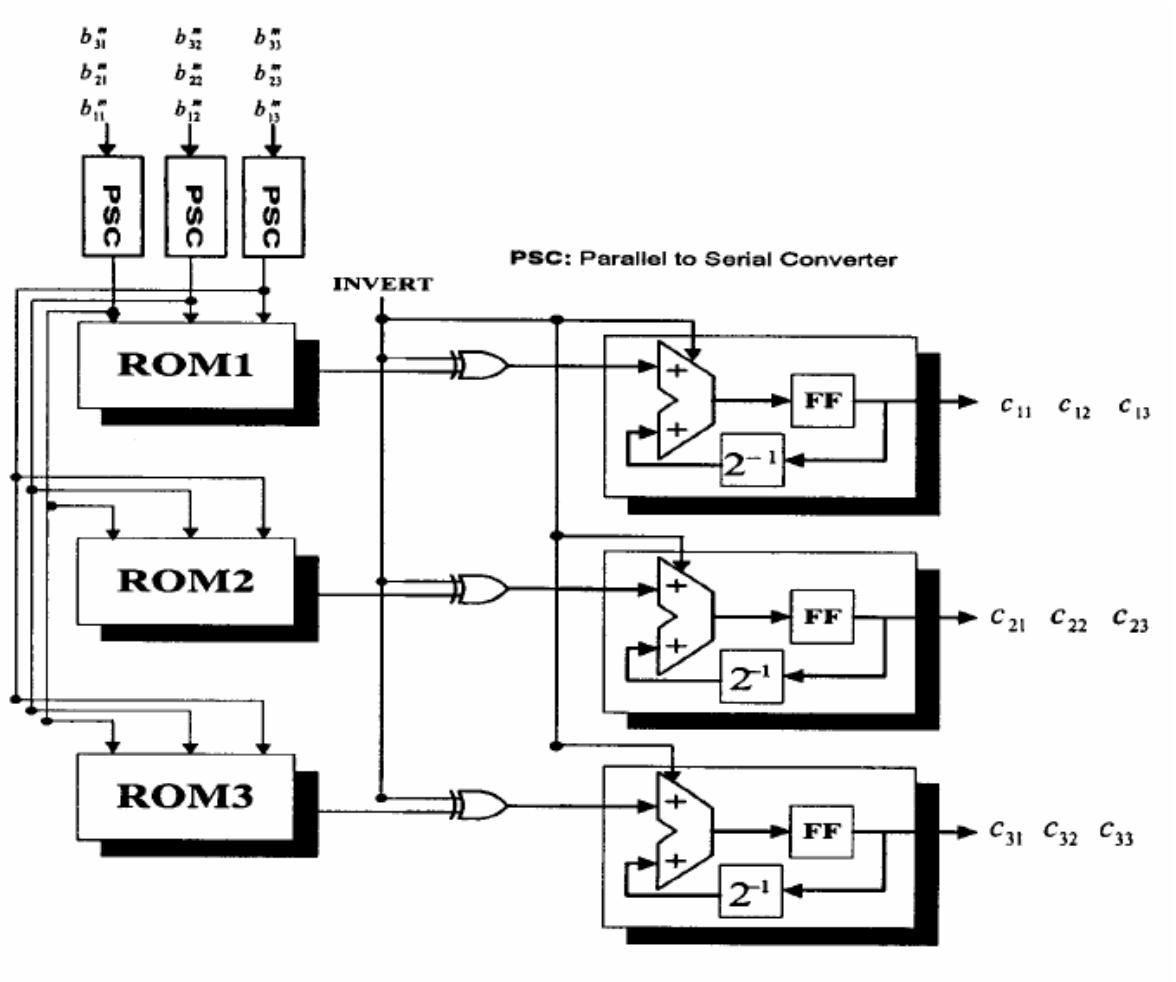
For å illustrere hvordan arkitekturen i Figur 2-9 fungerer skal vi se på hvordan produktelementet c11 beregnes. c11 beregnes av PEene PE11, PE12 og PE13. Disse har elementene a11, a12 og a13 fra matrise A liggende fast. Elementene b11, b21 og b31 mates inn i PEene fra inngangsbufferet. PE13 multipliserer a13 og b31, PE12 multipliserer a12 og b21 og PE11 multipliserer a11 og b11. Produktet i PE13 sendes til PE12 og legges sammen med produktet der før summen av de to produktene sendes til PE11 og legges sammen med produktet der. Denne summen er element c11. De andre elementene i produktmatrisen beregnes på tilsvarende måte ved at elementene som ligger fast i PEene multipliseres med elementer fra inngangsbufferet og produktene legges sammen. Arkitekturene for andre verdier på N fungerer tilsvarende, men med N^2 PEer. Arkitekturen for N=4 oppnår 50MHz og bruker 80 slicer i en Xilinx XCV1000E FPGA.

I [11] presenterer Amira og Bensaali et system for matrisemultiplikasjon hvor matrisestørrelse, antall bit i elementene og arkitektur kan defineres av bruker. Systemet er laget i VHDL. Arkitekturene som kan velges av brukeren er systolisk arkitektur som i [12] og arkitektur basert på distribuert aritmetikk.

De systoliske arkitekturene i systemet fungerer på samme måte som arkitekturene i [12] og systemet generer VHDL kode for nye arkitekturer basert på brukerens behov, matrisestørrelse og antall bit pr element. En systolisk arkitektur for multiplikasjon av 6x6 matriser med åtte bit pr element oppnår 25MHz og bruker 300 CLB.

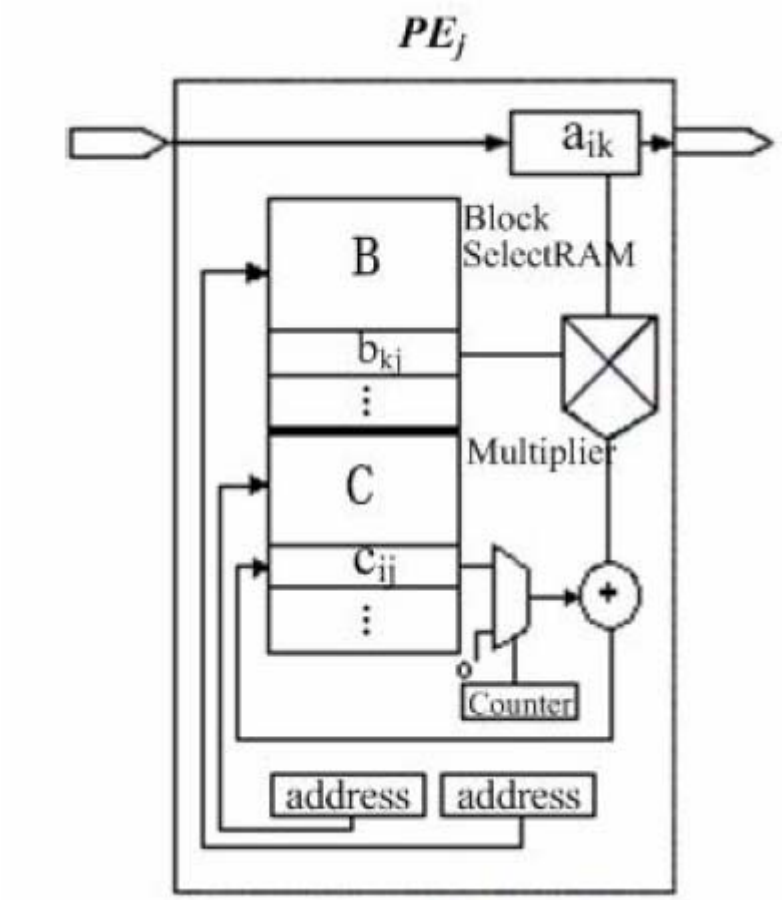
Arkitekturene basert på distribuert aritmetikk bruker ROMer for å implementere konstant koeffisient multiplikasjonsenheter[13] og akkumulatorer for å implementere en sum av produkter operasjon. Innholdet i ROMen avhenger av elementene i matrise A. Hver

multiplikasjonsenhet multipliserer to matriseelementer (et fra hver matrise) og akkumulerer produktet. Når alle elementene er multiplisert og produktene akkumulert er elementet i produktmatrisen beregnet. Et eksempel på en arkitektur basert på distribuert aritmetikk for multiplikasjon av 3x3 matriser er vist i Figur 2-12. Artikkelen oppgir ikke resultater for denne arkitekturen, men distribuert aritmetikk for multiplikasjon av 4x4 matriser med åtte bit elementer oppnår 166.47 MHz og bruker 57 slicer. Systemet er laget for implementasjon av systemene på en Xilinx XCV2000E FPGA.



Figur 2-12: arkitektur basert på distribuert aritmetikk for multiplikasjon av 3x3 matriser[11]

Jianwen og Chuen presenterer i [14] prosesseringselementer (PE) som benytter Xilinx Mac Core, se Figur 2-13. PEene er laget i VHDL. Mac Core bruker de interne multiplikasjonsenhetene i en Xilinx Virtex-II FPGA for å utføre multiplikasjon. PEene brukes uavhengig av hverandre. I artikkelen formuleres også tre setninger om krav til ressurser til matrisemultiplikasjon, de to første presenteres her mens 3. setning som omhandler bruk av færre enn N multiplikasjonsenheter ikke er aktuell for denne rapporten.



Figur 2-13:Prosesseringselement som benytter Xilinx Mac Core[14]

PEene som presenteres inneholder en MaC core, et register, en blokk RAM (på minst 2N lokasjoner à størrelse på matriseelementene) og en I/O port. PEene er testet for multiplikasjon av NxN matriser med N= 3,6,9,12,15,24 og 48 og bruker fra 110 til 1798 slices. Oppnådd operasjonsfrekvens er 74 MHz. PEene er testet ut på en Xilinx Virtex-II FPGA.

1. setning: minimum antall multiplikasjoner som trengs for å multiplisere NxN matriser med N multiplikasjonsenheter er N^2, med en multiplikasjonsenhet trengs N^3.

Bevis: En NxN matrise inneholder N^2 elementer. For å beregne et element i produkt matrisen trengs N multiplikasjoner (av elementene i rad/kolonne á N elementer). For å beregne N^2 elementer med en multiplikasjonsenhet trengs da N^2 x N = N^3 multiplikasjoner. Med N multiplikasjonsenheter trengs N^3 /N = N^2 multiplikasjoner. Dette er minimum antall multiplikasjoner som trengs.

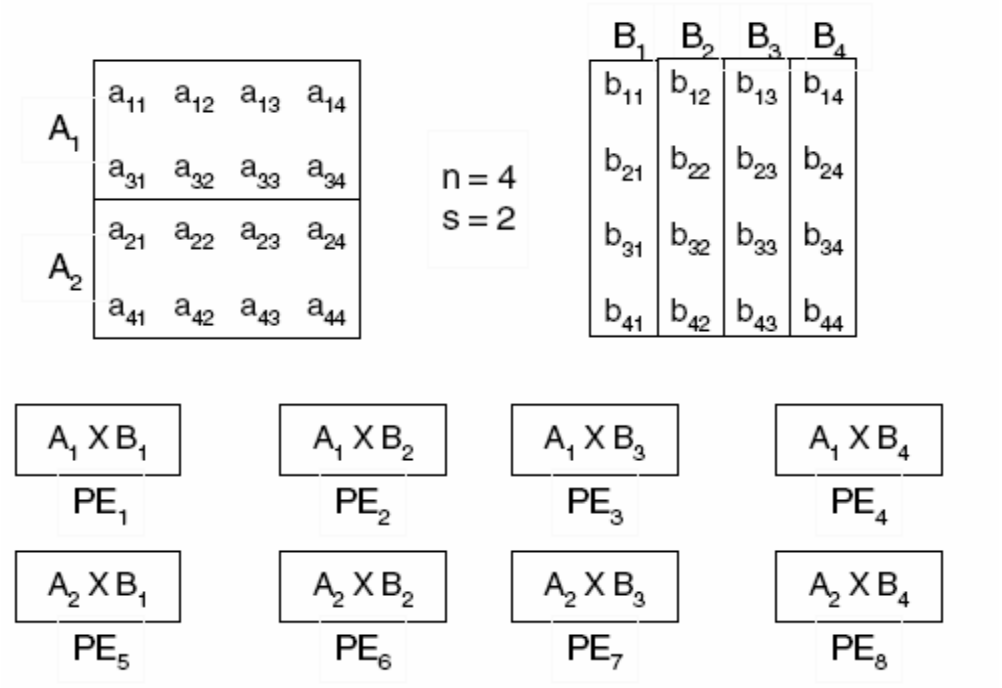
Kommentar: Denne setningen omhandler multiplikasjon av NxN matriser generelt og er trolig kjent fra før, selv om jeg ikke har kunnet finne referanser til den.

2. setning: For multiplikasjon av NxN matriser med PEer bestående av en Mac, et register, en blokk RAM (innebygd i FPGAen) på 2N lokasjoner og en I/O port, se Figur 2-13, trengs N^2 +N sykler med N PEer. *Kommentar:* PEene inneholder en kolonne fra matrise B hver og elementene i radene i matrise A lastes inn utenfra.

Bevis: Her forutsettes at PEene er organisert slik at elementene i matrise A mates inn i PEene en etter en slik at første element i A når siste PE etter N sykler slik at denne begynner å multiplisere etter N sykler, derfor tillegg på N sykler.

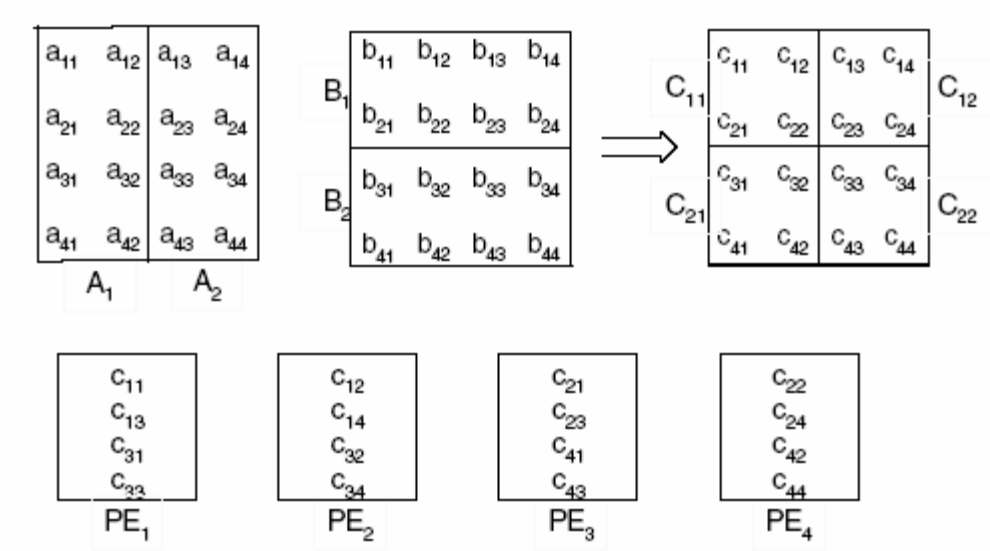
I [15] presenterer Zhuo og Prasanna to algoritmer for matrisemultiplikasjon av NxN matriser med flytallselementer. Algoritmene er laget i VHDL. De to algoritmene bruker PEer (multiplikasjonsenheter) som benytter interne multiplikasjonsblokker i FPGA i en arkitektur med uavhengige enheter.

Første algoritme baserer seg på å dele den ene matrisen i to undermatriser med størrelse N/2xN og den andre matrisen i fire kolonner, se Figur 2-14. Elementene i produktmatrisen beregnes av 2N prosesseringselementer (PE). Hver PE multipliserer elementene i en av undermatrisene med elementene i en kolonne fra den andre matrisen. De fire første PEene multipliserer elementene i den første undermatrisen med elementene kolonnene i den andre matrisen og de fire siste PEene multipliserer elementene i den andre undermatrisen med elementene i kolonnene den andre matrisen. Hver PE beregner N/2 elementer i produktmatrisen.



Figur 2-14:Illustrasjon av algoritme 1[15]

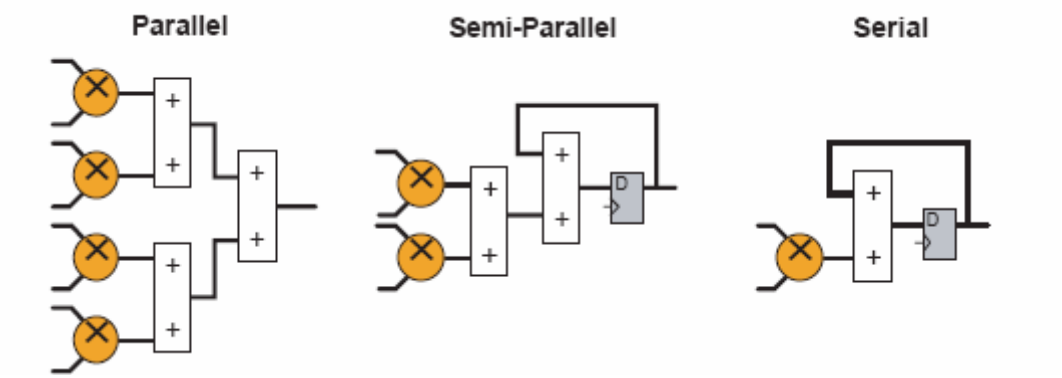
Den andre algoritmen deler hver matrise i to, den første i to undermatriser med størrelse NxN/2 og den andre matrisen i to undermatriser med størrelse N/2/N, se Figur 2-15. Elementene i produktmatrisen beregnes av N PEer, hver PE beregner N elementer i produktmatrisen i form av en N/2xN/2 undermatrise av produktmatrisen. Elementene beregnes ved at kolonnene i undermatrisen av første matrise og radene i undermatrisen av andre matrise som er tilknyttet hver PE multipliseres og produktene legges sammen.



Figur 2-15: Illustrasjon av algoritme 2[15]

Første algoritme oppnår 182 MHz og bruker 19045 slices. Andre algoritme oppnår 135 MHz og bruker 33589 slices. Det er også laget et Händel-C program for flyttalls matrisemultiplikasjon, dette programmet oppnår 12.6 MHz og bruker 22059 slices. Alle tre systemene er implementert i en Xilinx XC2VP125 FPGA

I [16] presenteres implementasjon av Multiply Accumulate (Sum av produkter (SAP)) i Xilinx FPGA. SAP kan implementeres på flere måter, parallell som bruker en multiplikasjonsenhet pr rad/kolonne element og addisjonsblokker som legger sammen delproduktene, semi-parallell, som bruker to multiplikasjonsblokker, en addisjonsblokk og en akkumulator, se Figur 2-16. Til slutt kan SAP implementeres serielt med en multiplikasjonsblokk og en akkumulator.



Figur 2-16: forskjellige muligheter for implementasjon av sum av produkter[16]

Arbeidet i artiklene som er beskrevet i dette kapittelet er utgangspunkt for mine egne forsøk innen bruk av FPGA til matrisemultiplikasjon. Selv om artiklene omhandler bruk av andre FPGA kretser og det en av artiklene omhandler flyttall har konseptene som er presentert i dem og den inspirasjonen de har gitt vært avgjørende i mitt eget arbeid.

3 PARALLELL MATRISEMULTIPLIKASJON I FPGA

I dette kapittelet beskrives systemene for multiplikasjon av 4x4 og 12x12 matriser.

3.1 Felles for systemene

Tanken bak systemene i dette kapittelet er at radene og kolonnene i matrisene som skal multipliseres skal lagres bare en gang i FPGAen for å spare plass og unngå problemer med flere utgaver av rad/kolonne.

Systemene er basert på metodene brukt i og inspirert av arbeidene omtalt i kapittel 2.5. Sentralt i systemene er sum av Produkter (SAP) enheter. SAP enhetene er laget på to måter universell som er basert på ”serial multiply-accumulate” i [16] og tilpasset som er basert på ”parallel multiply-accumulate” i [16]. SAP enhetene er som PEene i [14] bygget opp rundt interne multiplikasjonsblokker i FPGAen.

Organiseringen av SAP enhetene er basert på Amira og Bensaali’s arkitektur med distribuert aritmetikk i [11]. Hver SAP enhet i systemene skal være uavhengig av de andre SAP enhetene i systemet.

Organiseringen av matrisene bygger på Zhuo og Prasanna’s oppdeling av matrisene i [15], men er i motsetning til deres oppdeling finere og deler matrisene i rader/kolonner. Oppdelingen samsvarer med rad-kolonne regelen i 2.2.

Systemene skiller seg fra systemet i [15] ved at matrisene deles inn i enkelt rader/kolonner, mens de i [15] deles inn i undermatriser og kolonner. Systemene skiller seg og fra systemene i [14] ved at matrisene behandles i rader/kolonner og ikke i elementer. Systemene bruker en organisering tilsvarende den som brukes i [12], men skiller seg fra systemet i [12] ved at matrisemultiplikasjonene utføres ved hjelp av interne multiplikasjonsblokker i FPGAen.

3.1.1 Organisering av matriser

Systemene beregner produktmatrisen R ved å multiplisere to matriser A og B. I systemene for multiplikasjon av 4x4 er A, B og R 4x4 matriser, se Figur 3-1. Matrisene A og R deles i fire rader hver og lagres i RAMene A0-A3 (matrise A) og R0-R3 (matrise R), matrise B deles i fire kolonner og lagres i RAM B0-B3 (matrise B), se Tabell 3-1, Tabell 3-2 og Tabell 3-3.

I systemene for multiplikasjon av 12x12 matriser er A 12x12 matriser. Matrise A og R deles i 12 rader hver og lagres i RAMene A0-A11 (matrise A) og R0-R11 (matrise R), matrise B deles i 12 kolonner som lagres i RAMene B0-B11, se vedlegg A.

Produktmatrisen R beregnes i rader, hver rad lagres i en RAM, se Tabell 3-3. Det beregnes et element pr rad av gangen.

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \quad B = \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{bmatrix} \quad R = AB = \begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ r_{21} & r_{22} & r_{23} & r_{24} \\ r_{31} & r_{32} & r_{33} & r_{34} \\ r_{41} & r_{42} & r_{43} & r_{44} \end{bmatrix}$$

Figur 3-1: 4x4 matrisene A, B og R.

For å forenkle programmeringsarbeidet er indeksene vist i Figur 3-2 benyttet for å identifisere elementene i matrisene.

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} = \begin{bmatrix} a0 & a1 & a2 & a3 \\ a4 & a5 & a6 & a7 \\ a8 & a9 & a10 & a11 \\ a12 & a13 & a13 & a15 \end{bmatrix}$$

$$B = \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{bmatrix} = \begin{bmatrix} b0 & b1 & b2 & b3 \\ b4 & b5 & b6 & b7 \\ b8 & b9 & b10 & b11 \\ b12 & b13 & b14 & b15 \end{bmatrix}$$

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ r_{21} & r_{22} & r_{23} & r_{24} \\ r_{31} & r_{32} & r_{33} & r_{34} \\ r_{41} & r_{42} & r_{43} & r_{44} \end{bmatrix} = \begin{bmatrix} r0 & r1 & r2 & r3 \\ r4 & r5 & r6 & r7 \\ r8 & r9 & r10 & r11 \\ r12 & r13 & r14 & r15 \end{bmatrix}$$

Figur 3-2: Indekser benyttet i systemene

	RAM			
Adresse	**A0**	**A1**	**A2**	**A3**
0	a0	a4	a8	a12
1	a1	a5	a9	a13
2	a2	a6	a10	a14
3	a3	a7	a11	a15

Tabell 3-1: Elementene i matrise A i RAMene A0-A3.

	RAM			
Adresse	**B0**	**B1**	**B2**	**B3**
0	b0	b1	b2	b3
1	b4	b5	b6	b7
2	b8	b9	b10	b11
3	b12	b13	b14	b15

Tabell 3-2:Elementene i matrise B i RAMene B0-B3

	RAM			
Adresse	**R0**	**R1**	**R2**	**R3**
0	ro	r4	r8	r12
1	r1	r5	r9	r13
2	r2	r6	r10	r14
3	r3	r7	r11	r15

Tabell 3-3: Elementene i produktmatrisen R i RAMene R0-R3

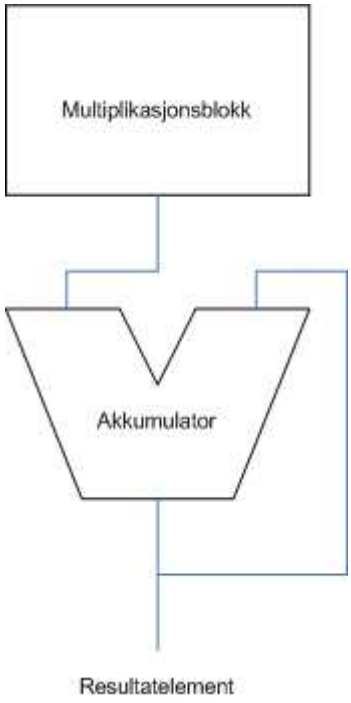
3.1.2 Sum av produkter

Elementene i produktmatrisen er en sum av produkter av radene og kolonnene i matrisene som multipliseres, se avsnitt 2.2. Sum av produkter (SAP) (engelsk: multiply accumulate) er en kjent teknikk fra digital signalbehandling [16]. SAP består av å multiplisere elementer til delprodukter og legge sammen delproduktene til endelige produkter. SAP er realisert på to måter; universell og tilpasset.

3.1.3 Universell SAP

Universell SAP er basert på ” serial multiply-accumulate” i [16] og realiseres med en multiplikasjonsblokk og en akkumulator, se Figur 3-3. Multiplikasjonsblokka multipliserer et element fra raden med et fra kolonnen som skal multipliseres om gangen og resultatene akkumuleres i akkumulatoren. Etter siste akkumulasjon er et element i produktmatrisen

beregnet. Elementet ligger i samme rad/kolonne som det er produktet av. For å beregne et element i produktmatrisen trengs N multiplikasjoner og N akkumulasjon da antall elementer i rad/kolonne er N. Et element i en 4x4 matrise beregnes med fire multiplikasjoner og fire akkumulasjoner, mens et element i en 12x12 matrise krever 12 multiplikasjoner og 12 akkumulasjoner.

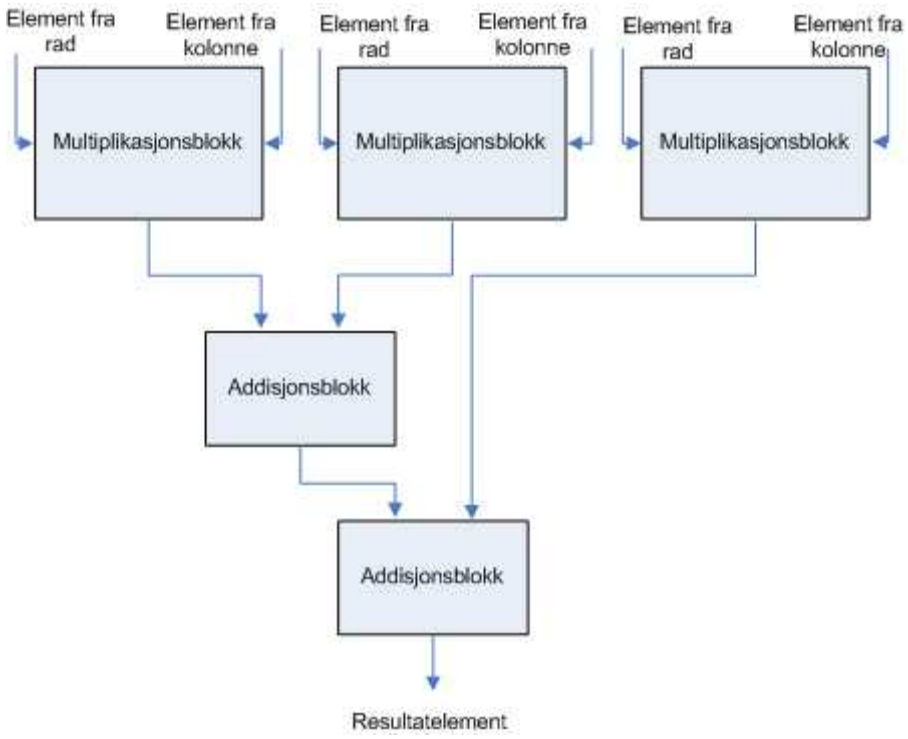


Figur 3-3: Universell SAP

Denne måten å realisere SAP operasjonen kan brukes for alle NxN matriser uten tilpassning.

3.1.4 Tilpasset SAP

Tilpasset SAP er basert på og ”parallel multiply-accumulate” i [16] og skiller seg fra universell SAP ved at alle elementene multipliseres samtidig i parallell istedenfor to og to serielt. Dette krever N multiplikasjonsblokker (en pr element i rad/kolonne) og N-1 addisjonsblokker i en trestruktur for å addere produktene se Figur 3-4. Trestruktur brukes for å unngå at addisjonen implementeres som stor og treg logikk [5]. Elementer fra samme adresse i A og B RAMene multipliseres i samme multiplikasjonsblokk.



Figur 3-4: Tilpasset SAP for multiplikasjon av 3x3 matriser.

Tilpasset SAP må tilpasses størrelsen på matrisene som skal multipliseres.

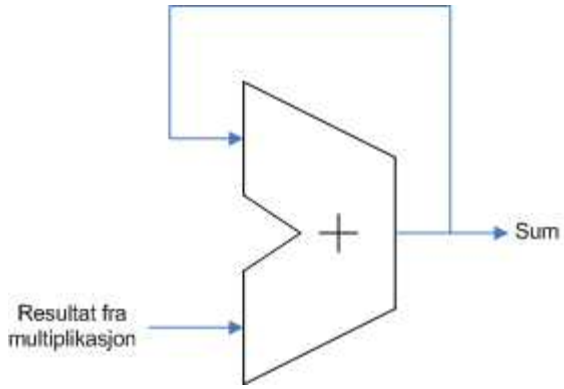
For NxN matriser trengs N multiplikasjonsblokker og N-1 addisjonsblokker. Antall addisjonsblokker avgjøres av antall multiplikasjonsblokker. Addisjonsblokkene organiseres i en trestruktur hvor antall addisjonsblokker i hvert nivå er halvparten av antallet i forrige nivå, antallet i første nivå er halvparten av antallet multiplikasjonsblokker.

3.1.5 Multiplikasjonsblokk

Multiplikasjonsblokkene i både universell og tilpasset SAP er interne 18 bits multiplikasjonsblokker i FPGAen. Multiplikasjonsblokkene er nærmere beskrevet i 2.3.3.

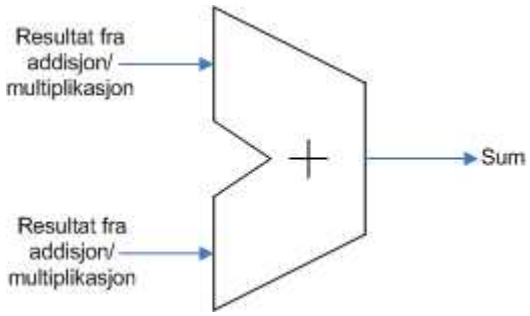
3.1.6 Addisjonsblokker

I universell SAP brukes en akkumulator som den i Figur 3-5. En akkumulator er en adder med utgangen koblet tilbake til den ene inngangen slik at verdiene som mates inn på den frie inngangen legges sammen med verdien fra forrige akkumulasjon.



Figur 3-5: 36 bits akkumulator

I tilpasset SAP brukes addere, lik den i Figur 3-6 i en trestruktur som vist i Figur 3-4.



Figur 3-6: 36 bits adder

Både akkumulatorer og addere er implementert med LUTer og vipper i FPGAens CLBer.

3.1.7 Valgte matrisestørrelser

Det er laget løsninger for 4x4 og 12x12 matriser. For å komme i gang begynte jeg å arbeide med 4x4 matriser pga liten størrelse som medfører færre SAP operasjoner og mindre kompleksitet. Multiplikasjon av 4x4 matriser brukes til å transformere nord-øst koordinater til x-y koordinater[17] og i 3D-programmering for å kontrollere et objekts bevegelser om aksene[18]. 12x12 matriser ble valgt med utgangspunkt i hvilke ressurser som er tilgjengelig i FPGAen løsningene er utviklet mot. Med 12x12 matriser utnyttes ressursene i denne maksimalt. FPGAen innholder 144 multiplikasjonsblokker[3] og ved å benytte tilpasset SAP vil 12x12 matriser være maksimal matrise størrelse som kan benyttes. 12x12 matrise dvs. 12 matriseelementer pr rad/kolonne og 12 rader/kolonner pr matrise og med *en* multiplikasjonsblokk pr element i rad/kolonne kreves da 144 multiplikasjonsblokker for å realisere systemet. Å finne en anvendelse for multiplikasjon av 12x12 matriser har vært vanskelig, men 12x12 matriser anvendes i HDTV for koding/dekoding av signal [19].

3.1.8 I/O

For å kunne beregne produktmatrisen R av matrisene A og B må A og B overføres til RAMer inne i FPGAen. For å unngå at denne overføringen blir en flaskehals i systemet er det nødvendig å finne gode løsninger for overførsel av elementene i matrisene A, B og R.

To alternativer for å løse I/O problemet

Virtex-II FPGAen sitter på et ”pci mezzanine card” (pmc)[4]. Ved å bruke PCI standarden kan data overføres til FPGAen med overføringsrater på opptil 528 Mb/s[20]. Å benytte denne I/O løsningen vil kreve 724 LUTer og 89 I/O blokker [20]. XC2V6000 FPGAen inneholder 67584 LUT [3]. Det største av systemene (12x12 multiplikasjon med tilpasset SAP) krever 170 LUTer, se kapittel 4.2, så det er plass til å implementere løsningen i FPGAen.

Om bruk av PCI I/O ikke gir høy nok overføringsrate kan det lages et kretskort til FPGAen fra bunnen og benytte Low Voltage Differential Signalling (LVDS) I/O[21]. Ved å benytte LVDS kan overføringsrater på 10Gb/s for 16bit overførsel eller 2.5 Gb/s for 4bit[22]. Denne løsningen vil gi svært høy overføringsrate, men den ulempen at det må lages et kretskort fra bunnen for å bruke LVDS. Kretskortet må møte LVDS standardens krav til forsinkelse, lengden på transmisjonslinjene og plasseringen av disse på kretskortet må avveies nøye [22].

3.1.9 Begrensinger for systemene

Begrepet *omgang* benyttes for å beskrive beregningen av et element i produktmatrisen. En omgang er det antall operasjoner (multiplikasjoner og addisjoner/akkumulasjoner) som trengs for å beregne et element.

Generelt for matrisemultiplikasjon med SAP enheter (universelle og tilpassede) gjelder følgende.

Multiplikasjon av to NxN matriser kan ikke utføres i løpet av mindre enn N omganger med N SAP enheter.

Kommentar: En NxN matrise inneholder N^2 elementer. Med N SAP enheter som hver beregner et element i løpet av en omgang beregnes produktmatrisen i løpet av N omganger.

For de universelle SAP enhetene består en *omgang* av N serielle multiplikasjoner og N akkumulasjoner. Totalt 2N steg.

For de tilpassede SAP enhetene er lengden på omgangen avhengig av matrisestørrelsen den enkelte SAP enheten er tilpasset.

I denne rapporten er det brukt tilpassede SAP enheter for 4x4 og 12x12 matriser.

For SAP enhet tilpasset 4x4 matriser består en *omgang* av fire parallelle multiplikasjoner etterfulgt av tre addisjoner i to steg. Først fire delprodukter til to delresultater og tilslutt to delresultater til ett produktelement. En *omgang* = et steg multiplikasjoner + to steg addisjoner = tre steg.

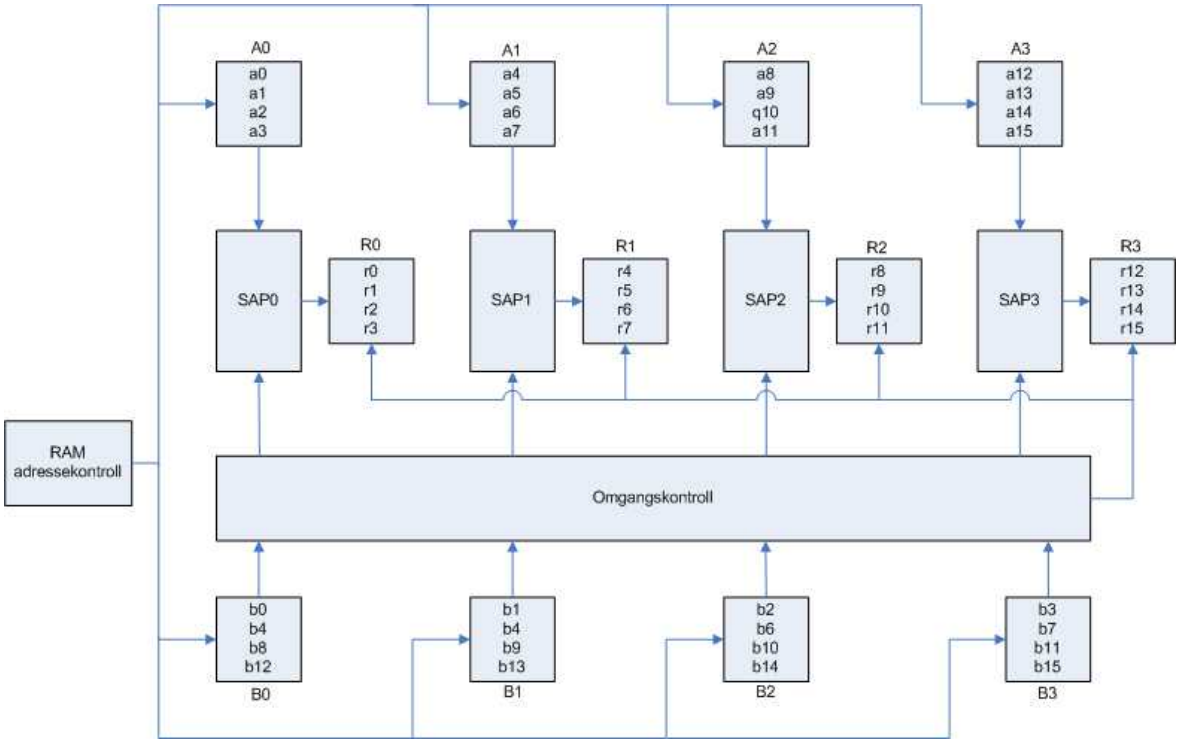
For SAP enhet tilpasset 12x12 matriser består *en omgang* av et steg som består av 12 parallelle multiplikasjoner etterfulgt av 11 addisjoner i fire steg. En *omgang* = et steg multiplikasjoner + fire steg addisjoner = fem steg.

I systemene med både universell og tilpasset SAP lagres både matrisene som skal multipliseres og produktmatrisen i RAMer i FPGAen. Matrisestørrelsen NxN avgjør hvor mange lokasjoner RAMene må ha. Hver RAM må være på N lokasjoner á 18 bit for matrisene som skal multipliseres (18 bit fordi det brukes 18bit multiplikasjonsblokker i systemene) og N lokasjoner á 36 bit for produktmatrisen (36 bit fordi multiplikasjonsblokkene i systemene har 36 bit utgang).

3.2 Multiplikasjon av 4x4 matriser.

Det er laget løsninger for å multiplisere 4x4 matriser med både universell og tilpasset SAP.

3.2.1 Universell SAP



Figur 3-7: System for multiplikasjon av 4x4 matriser med universell SAP.

Systemet i Figur 3-7 beregner produktmatrisen R ved å multiplisere 4x4 matrisene A og B. Matrisene A og B lagres i RAMene A0-A3 og B0-B3. Elementene i R beregnes av universelle SAP enheter, se avsnitt 3.1.3, og lagres i RAMene R0-R11. Organiseringen av matrisene A, B og R er beskrevet i avsnitt 3.1.1.

En universell SAP enhet kan beregne et element i R av gangen. De fire SAP enhetene i systemet beregner til sammen fire elementer (en rad) i R av gangen. Hele R kan derfor ikke beregnes samtidig, men må beregnes i omganger. I hver omgang beregnes et element av hver SP enhet til sammen fire elementer. En omgang for universelle SAP enheter er beskrevet i 3.1.9. De 16 elementene i R beregnes i løpet av fire omganger, se Tabell 3-4.

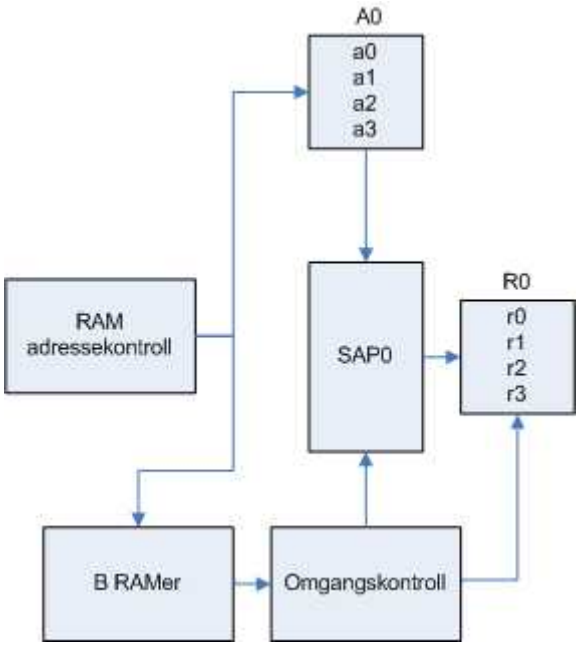
Multiplikasjonene styres av to kontrollenheter, RAM adressekontroll og omgangskontroll. RAM adressekontroll kontrollerer hvilke adresser som leses av SAP enhetene fra RAM A0-A3 og B0-B3. Omgangskontroll kontrollerer hvilken B RAM som er tilknyttet hvilken SAP enhet. Kildekoden for systemet finnes i vedlegg B.1.

Omgang	Element i R			
1	r0	r4	r8	r12
2	r1	r5	r9	r13
3	r2	r6	r10	r14
4	r3	r7	r11	r15

Tabell 3-4: Beregning av elementene i R

SAP FUNKSJON

SAP enhetene i systemet i Figur 3-7 er tilknyttet en A RAM, en B RAM og en R RAM. A RAMen inneholder en rad fra matrise A og B RAMen inneholder en kolonne fra B. I R RAMen lagres de elementene i produktmatrisen R som beregnes av SAP enheten. Figur 3-8 viser SAP0 i systemet med tilknyttede RAMer og kontrollenheter.



Figur 3-8: SAP0 fra *Figur 3-7*

For å vise hvordan SAP enhetene beregner elementene i R ser vi på SAP0 og hvordan den fungerer, se Figur 3-8.

SAP0 er fast tilknyttet RAMene A0 og R0, se Tabell 3-5 og tilknyttes en ny RAM som inneholder en kolonne fra matrise B for hver omgang, se Tabell 3-8. For hver omgang lagrer SAP0 produktet av SAP operasjonen i RAM R0. Adressen elementet lagres i avgjøres av hvilken omgang systemet er i, se Tabell 3-9.

	RAM	
SAP	**A RAM**	**R RAM**
0	A0	R0
1	A1	R1
2	A2	R2
3	A3	R3

Tabell 3-5: Fast tilknyttede RAMer

SAP0 beregner elementene r0, r4, r8 og r12 i R, se Tabell 3-4.

Elementene beregnes ved at innholdet i samme adresse i A0 og B RAMen som er tilknyttet for omgangen multipliseres og produktet legges sammen med produktet av innholdet i resten av adressene i RAMene. For å illustrere hvordan elementene beregnes skal vi se på hvordan element r8 i R beregnes, se Figur 3-9.

Element r8 beregnes i omgang 3, se Tabell 3-4. SAP0 er fast tilknyttet A0. I omgang 3 er SAP0 tilknyttet B RAMen B2, se Tabell 3-8. r8 lagres i R0[2], se Tabell 3-9.

Innholdet i første adresse i A2 og B0 multipliseres i SAP enhetens multiplikasjonsblokk og produktet legges inn i SAP enhetens akkumulator. Innholdet i de to neste adressene multipliseres og legges til resultatet i akkumulatoren. Dette gjentas til innholdet i alle fire RAM adressene er multiplisert og lagt sammen. Nå er r8 beregnet og kan lagres i RAM R0 adresse 2 (R0[2]), se Figur 3-9.

$$\begin{bmatrix} a0 & a1 & a2 & a3 \\ a4 & a5 & a6 & a7 \\ \mathbf{a8} & & & \\ a12 & a13 & a14 & a15 \end{bmatrix} \cdot \begin{bmatrix} & b1 & b2 & b3 \\ & b5 & b6 & b7 \\ & b9 & b10 & b11 \\ & b13 & b14 & b15 \end{bmatrix} = \begin{bmatrix} r0 & r1 & r2 & r3 \\ r4 & r5 & r6 & r7 \\ & r9 & r10 & r11 \\ r12 & r13 & r14 & r15 \end{bmatrix}$$

Figur 3-9: Beregning av elementet r8 i produktmatrisen

$$r8 = a8 \cdot b0 + a9 \cdot b4 + a10 \cdot b8 + a11 \cdot b12$$
$$= A3[0] \cdot B0[0] + A3[1] \cdot B0[1] + A3[2] \cdot B0[2] + A3[3] \cdot B0[3]$$
$$R0[2] = \sum_{i=0}^{3} A0[i] \cdot B2[i]$$

SYSTEM FUNKSJON

Elementene i produktmatrisen R beregnes som elementet r8. I hver SAP enhet multipliseres elementene i en rad fra matrise A med elementene i en ny kolonne fra matrise B i hver omgang. Tabell 3-6 viser hvilke elementer i R som beregnes av hvilken SAP enhet i hvilken omgang.

	SAP0	**SAP1**	**SAP2**	**SAP3**
Omgang1	r0	r4	r8	r12
Omgang2	r1	r5	r9	r13
Omgang3	r2	r6	r10	r14
Omgang4	r3	r7	r11	r15

Tabell 3-6: Elementer R som beregnes av de ulike SAP enhetene.

Systemet i Figur 3-7 består av fire universelle SAP enheter som arbeider i parallell, dvs de leser fra RAM samtidig, multipliserer innholdet i RAMene og akkumulerer produktene samtidig og de lagrer sluttproduktene (elementene i R) samtidig.

Oppbygning

RAM

RAMene i systemet er Blokk RAM, disse er beskrevet i avsnitt 2.3.2. A og B RAMene har 4 lokasjoner á 18bit og R RAMene er 4 lokasjoner á 36bit. Lokasjonene har adresser fra 0-3.

RAM adressekontroll

RAM adressekontrollen kontrollerer hvilke RAM adresser i RAM A0-A3 og B0-B3 som leses av SAP enhetene. Adressekontrollen er basert på variabelen i. Verdien på i avgjør hvilken adresse som leses, se Tabell 3-7.

i	**Adresse**
00	Adresse 0
01	Adresse 1
10	Adresse 2
11	Adresse 3

Tabell 3-7: RAM adresse avhengig av variabelen i.

Omgangskontroll

Omgangskontrollen er en 2-4 dekoder. Omgangskontrollen styrer skiftet mellom omganger basert på verdien til variabelen j. For hver omgang tilknyttes hver SAP enhet en ny B RAM og en ny adresse i R RAMen den er tilknyttet for lagring av det beregnede elementet, se Tabell 3-8 og Tabell 3-9

		SAP enhet			
Omgang	**j**	**0**	**1**	**2**	**3**
1	**00**	B0	B1	B2	B3
2	**01**	B1	B2	B3	B0
3	**10**	B2	B3	B0	B1
4	**11**	B3	B0	B1	B2

Tabell 3-8: Sannhetstabell for omgangskontroll, tilordning av B RAM

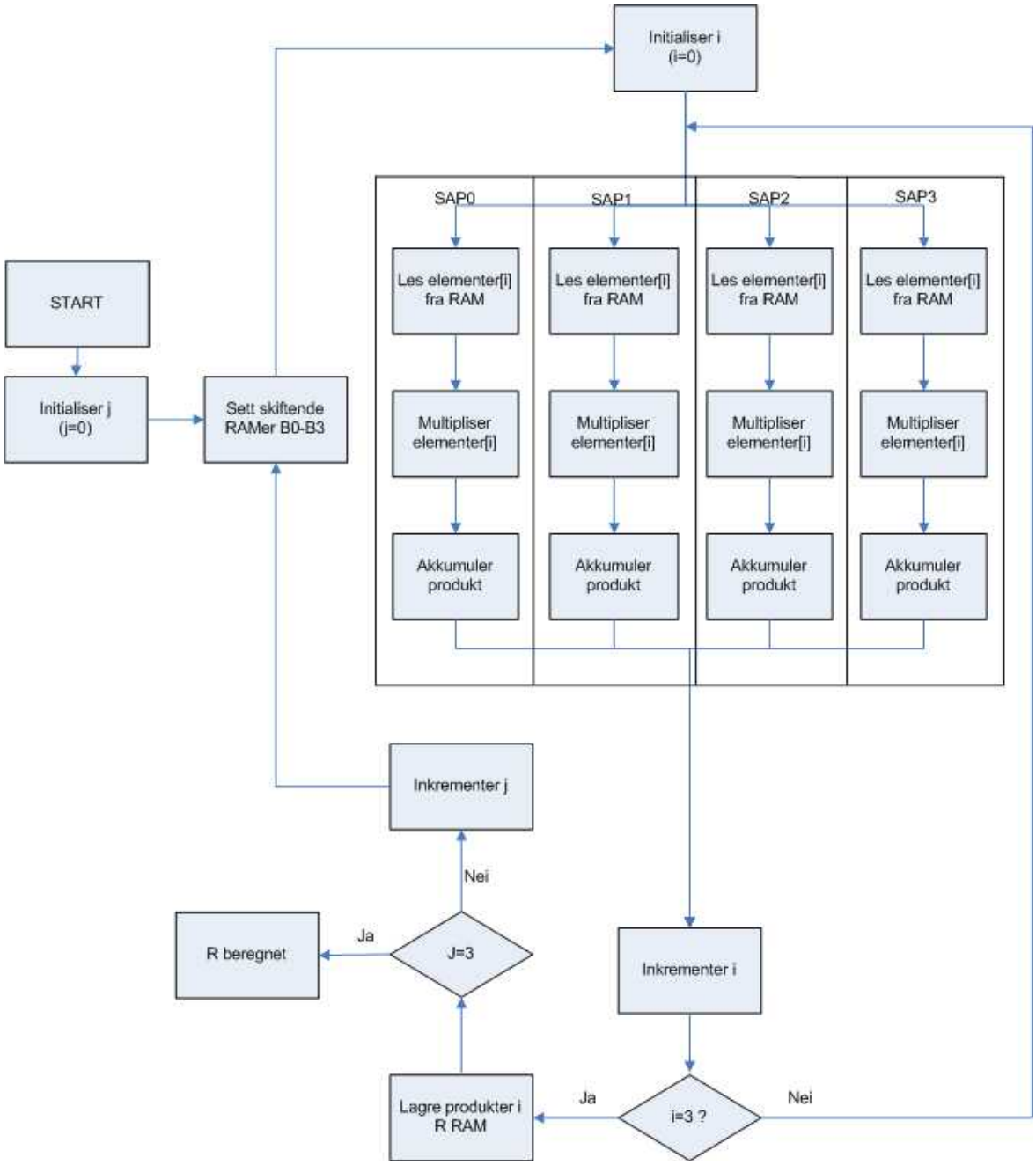
		SAP enhet			
Omgang	**j**	**0**	**1**	**2**	**3**
1	**00**	R0[0]	R1[0]	R2[0]	R3[0]
2	**01**	R0[1]	R1[1]	R2[1]	R3[1]
3	**10**	R0[2]	R1[2]	R2[2]	R3[2]
4	**11**	R0[3]	R1[3]	R2[3]	R3[3]

Tabell 3-9: Sannhetstabell omgangskontroll, R RAM adresse

SAP enheter

SAP enhetene er universelle SAP enheter, se 3.1.3 for beskrivelse.

Flyt for systemet



Figur 3-10: Flytskjema for systemet i *Figur 3-7*

Hver SAP enhet i Figur 3-7 er tilknyttet en A og en R RAM, Tabell 3-5 viser hvilke RAMer som er tilknyttet hvilken SAP enhet. I hver omgang tilknyttes hver SAP enhet en ny B RAM. Hvilken omgang systemet er i avhenger av verdien til variabelen j. Systemet begynner i omgang en og beregner elementene som vist i Tabell 3-4. Når elementene er beregnet og lagret går systemet videre til neste omgang hvor nye elementer beregnes. Dette gjentas til omgang fire er gjennomført og R er beregnet, se Figur 3-10.

Kommentar

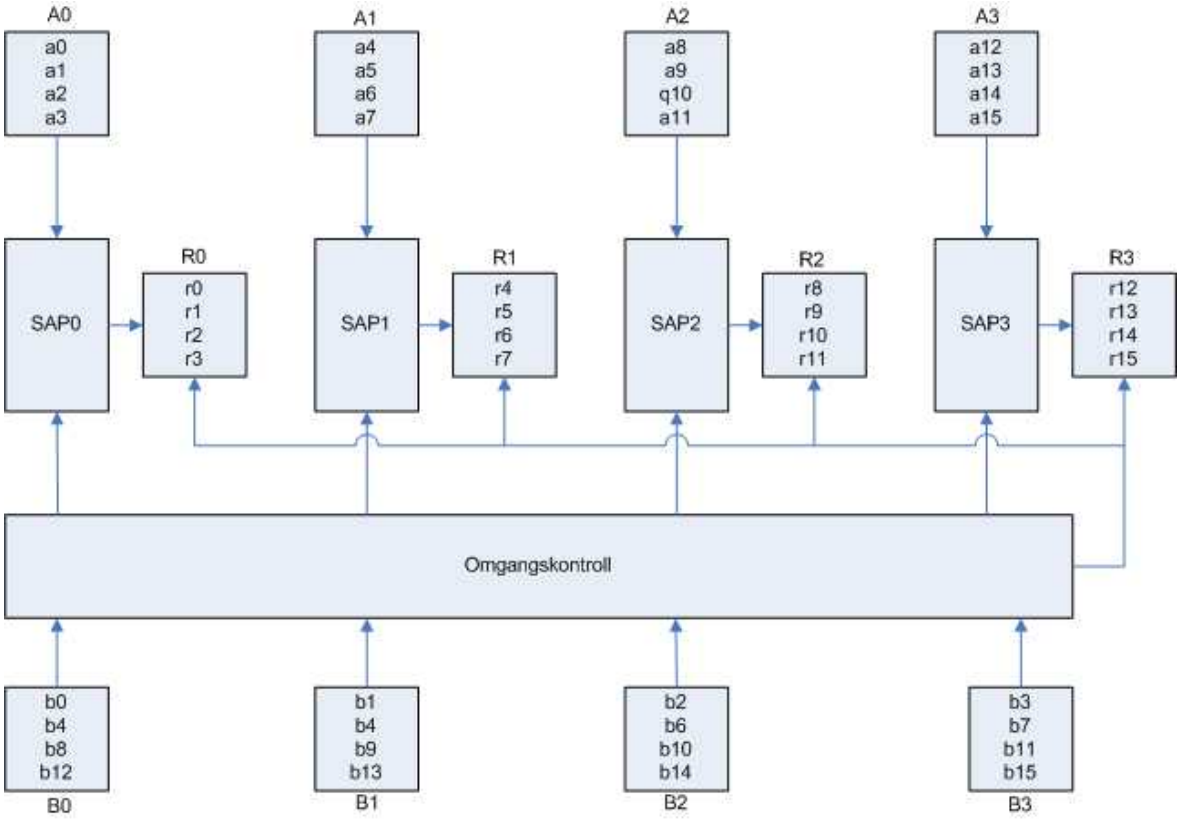
Systemet beregner produktmatrisen med universelle SAP enheter. Universelle SAP enheter krever lite ressurser og kan brukes for vilkårlige NxN matriser, men bruker N operasjoner for å beregne et element i produktmatrisen.

Estimert tidsforbruk for systemet

Fra 3.1.9 antall steg pr omgang for et system med universelle SAP enheter er 2N. I dette systemet er N = 4. Antall steg blir da 8 pr omgang. Systemet beregner produktmatrisen i løpet av fire omganger, så totalt antall steg for å beregne produktmatrisen blir 8*4 =32 steg.

Merk Dette estimatet inkluderer ikke tid som går med til I/O

3.2.2 Tilpasset SAP



Figur 3-11: System for multiplikasjon av 4x4 matriser med tilpasset SAP

Systemet i Figur 3-11 beregner produktmatrisen R av to 4x4 matriser A og B som systemet i avsnitt 3.2.1. Men i dette systemet beregnes elementene i R av tilpassede SAP enheter, se avsnitt 3.1.4. Matrisene A, B og R er delt i fire rader/kolonner og organisert i RAMer som i systemet i avsnitt 3.2.1.

R må som i systemet i kapittel 3.2.1 beregnes i omganger fordi hver tilpasset SAP enhet bare kan beregne et element i R av gangen. De fire SAP enhetene beregner fire elementer i en omgang. R inneholder 16 elementer og må beregnes i fire omganger

Multiplikasjonene i systemet styres av omgangskontroll. Omgangskontroll styrer hvilken omgang systemet befinner seg i. For hver omgang tilknyttes hver SAP enhet en ny B RAM og elementet SAP enheten beregner lagres i en ny adresse på samme måte som i avsnitt 3.2.1. Kildekoden for systemet finnes i vedlegg B.2.

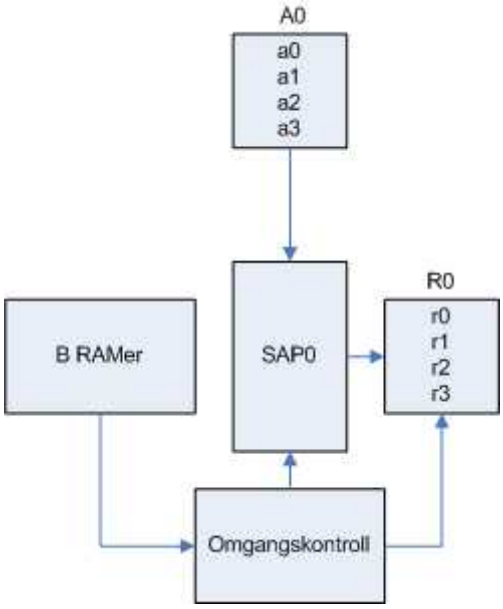
Funksjon for SAP enheten

For å vise hvordan de tilpassede SAP enhetene fungerer skal vi se på hvordan SAP0 i Figur 3-12 virker. Som i systemet i 3.2.1 er hver SAP enhet fast tilknyttet en A RAM og en R RAM, se Tabell 3-5 og tilknyttes en ny B RAM for hver omgang, se Tabell 3-8. SAP enhetene leser innholdet i RAMene adresse for adresse.

Multiplikasjonsblokk	**A adresse**	**B adresse**
0	A0[0]	B[1][0]
1	A0[1]	B[1]
2	A0[2]	B[2]
3	A0[3]	B[3]

Tabell 3-10: Tilordning av RAM innhold i SAP0

Elementene i R beregnes på tilsvarende måte som systemet i 3.2.1, men alle elementene multipliseres parallelt og produktene legges sammen til elementet som beregnes når multiplikasjonene er uført. Innholdet i RAM adressene multipliseres i multiplikasjonsblokkene og produktene adderes av addisjonsblokkene. Addisjonene utføres i to trinn. I første trinn adderes produktene fra multiplikasjonsblokk 0 og 1 og 2 og 3. I andre trinn adderes resultatene av addisjonene i første trinn. Resultatet av den siste addisjonen er elementet som beregnes.



Figur 3-12: SAP0 med tilknyttede RAMer

[1] SAP enhetene er tilknyttet en ny B RAM i hver omgang.

Funksjon for systemet

Systemet inneholder fire tilpassede SAP enheter som beregner elementene i R i parallell, fire elementer pr omgang. SAP enhetene leser innholdet de tilknyttede A og B RAMene, se Figur 3-12, en adresse av gangen. Deretter multipliseres elementene parallelt i multiplikasjonsblokkene før produktene av multiplikasjonene adderes til fire elementer i R i to steg. Systemet beregner R i fire omganger, skifte mellom omgangene styres med variabelen j, se Figur 3-14.

OPPBYGNING

RAM

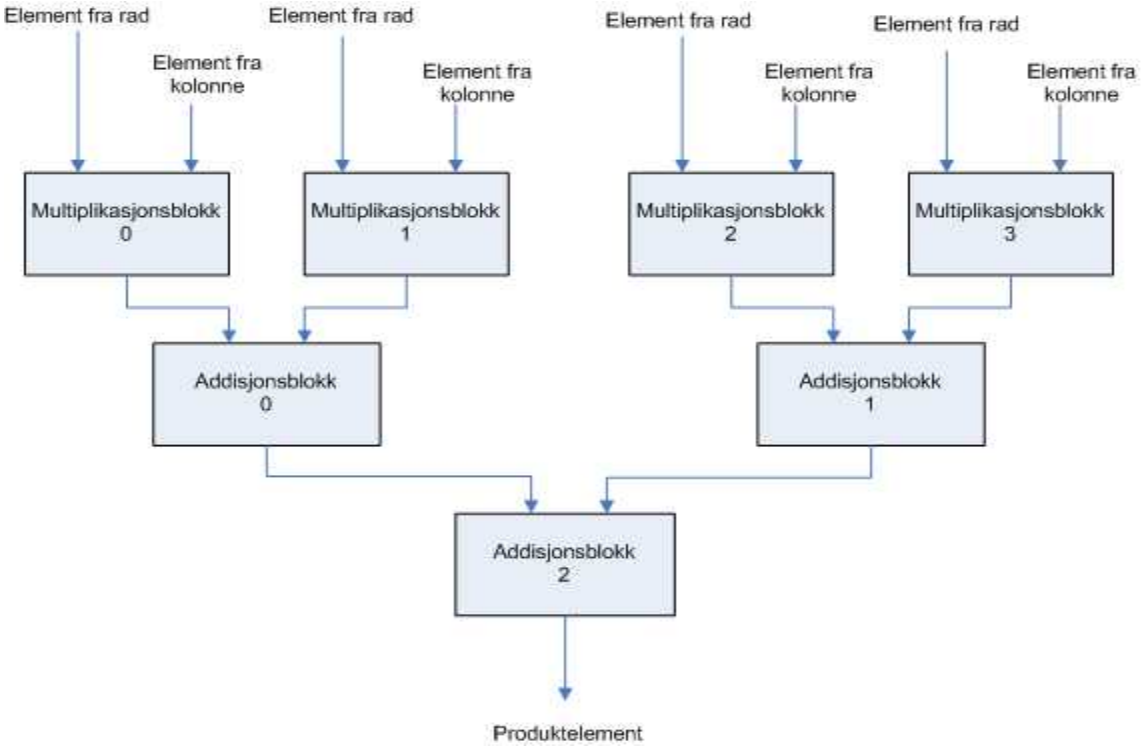
RAMene i systemet er Blokk RAM, se 2.3.2. A og B RAMene har 4 lokasjoner a18bit og R RAMene er 4 lokasjoner à 36bit. Lokasjonene har adresser fra 0-3.

Omgangskontroll

Omgangskontrollen styrer skifte mellom omgangene. For hver omgang blir SAP enhetene tilknyttet en ny B RAM og en ny adresse i den tilknyttede R RAMen for lagring av det beregnede R elementet. Omgangskontrollen er en to til fire dekoder basert på variabelen j, verdien på j avgjør hvilken omgang systemet befinner seg.

Tilpasset SAP enhet

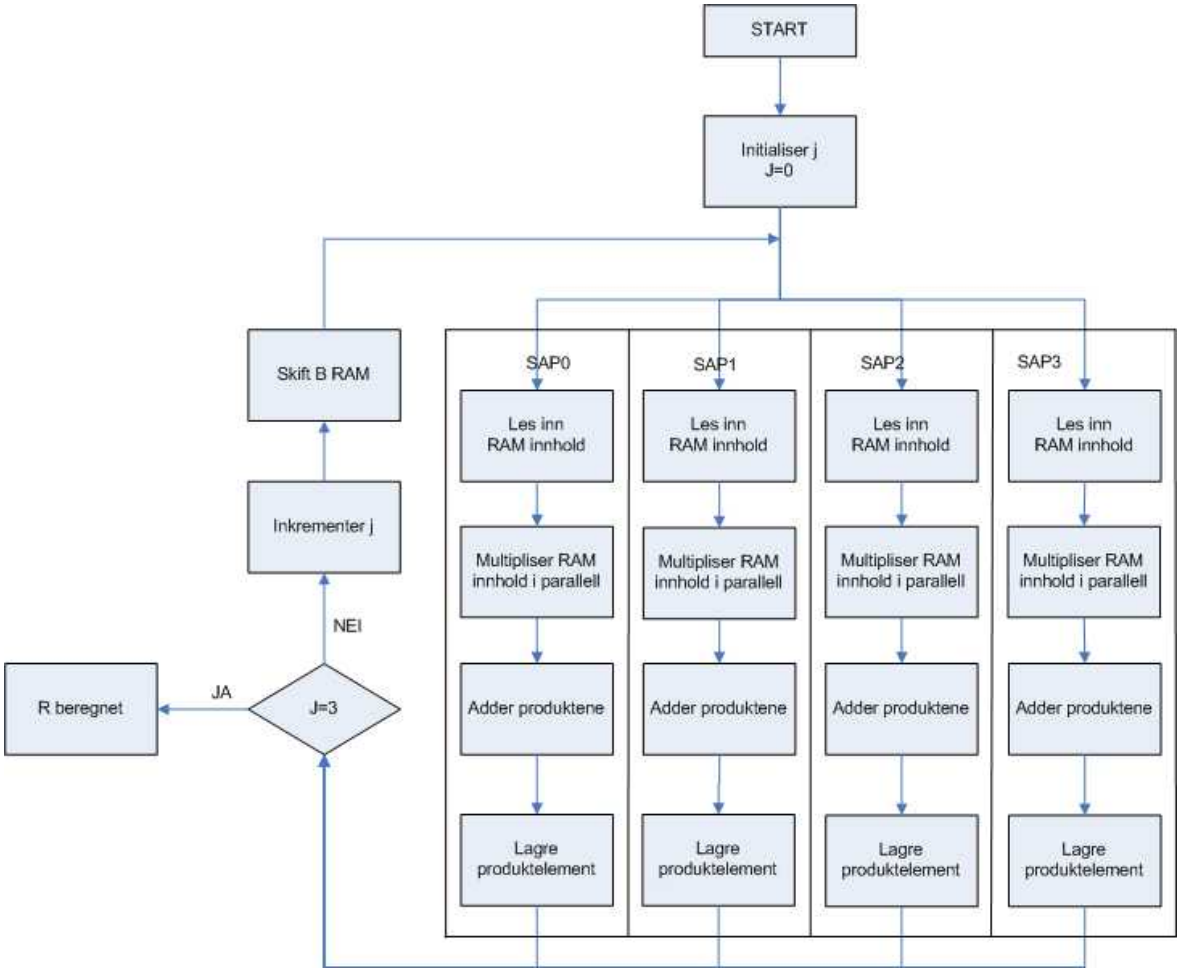
SAP enhetene er tilpassede SAP enheter, se 3.1.4



Figur 3-13: SAP0 fra systemet i Figur 3-11

SAP enhetene i systemet i er bygd opp som SAP0 i Figur 3-13. SAP enhetene er tilpasset multiplikasjon av 4x4 matriser og består av fire multiplikasjonsblokker og tre addisjonsblokker, se 3.1.5 og 3.1.6.

Flyt for systemet



Figur 3-14: Flytskjema for systemet i Figur 3-11

En A RAM og en R RAM, se Tabell 3-5, er tilknyttet hver SAP enhet. For hver omgang knyttes en ny B RAM, se Tabell 3-8 til hver SAP enhet. I hver omgang leser hver SAP enhet innholdet (elementene i matrise A og B) i alle adressene i de tilknyttede A og B RAMene før elementene multipliseres. Deretter adderes produktene av multiplikasjonene i hver SAP enhet til elementer i R, se Figur 3-14. Deretter lagres elementene før systemet går til neste omgang hvor SAP enhetene tilknyttes en ny B RAM, se Tabell 3-8 og fire nye elementer i R beregnes. Dette gjentas til alle elementene i R er beregnet, når omgang fire er fullført.

Kommentarer

Dette systemet beregner resultatmatrisen med tilpassede SAP enheter. Tilpassede SAP enheter krever mer ressurser enn universelle. SAP enhetene i dette systemet består av fire multiplikasjonsblokker og tre addisjonsblokker mot universelle SAP enheter som krever en multiplikasjonsblokk og en akkumulator. Som systemet i 3.2.1 beregnes produktmatrisen i fire omganger.

Estimert tidsforbruk

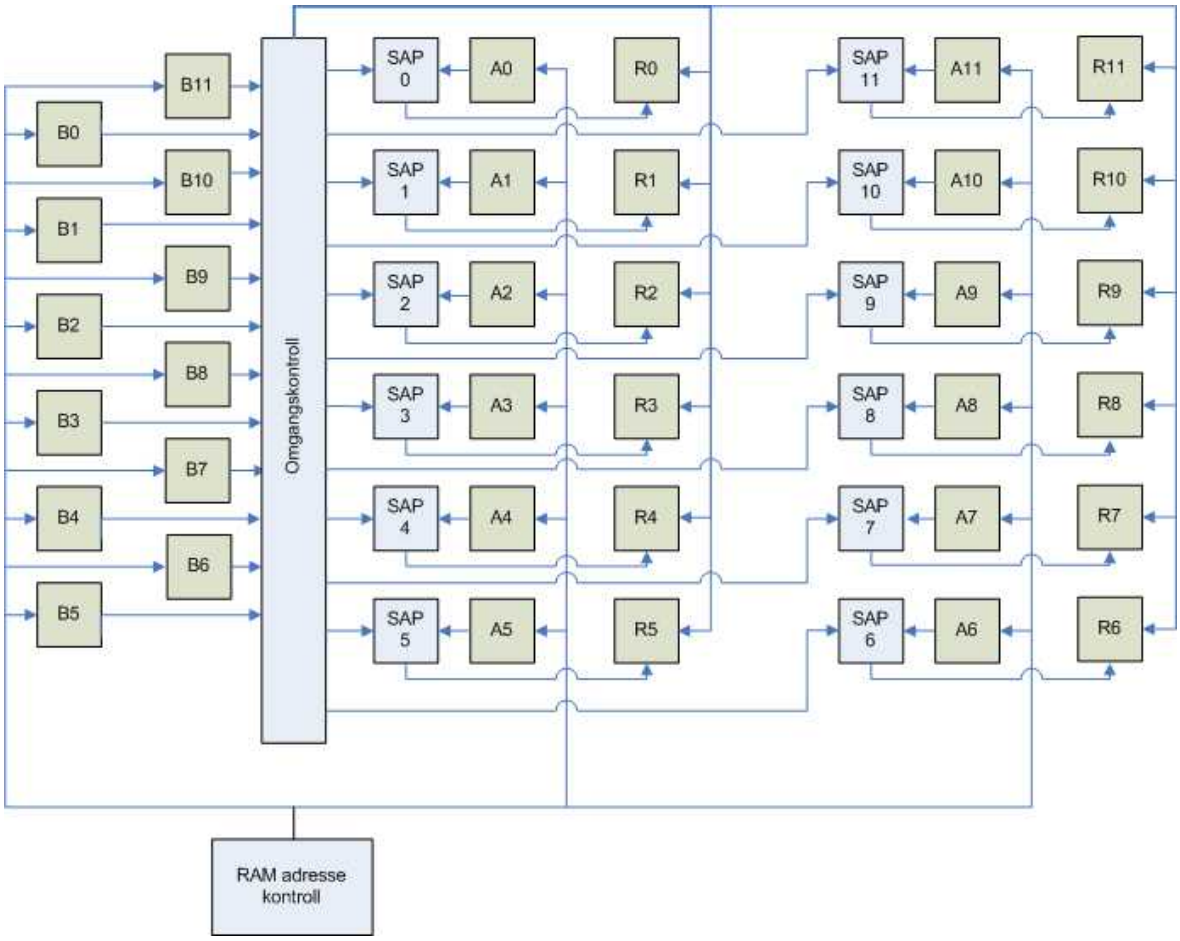
Fra 3.1.9, en omgang for en tilpasset SAP enhet for 4x4 matriser består av tre steg. Totalt antall steg for å beregne produktmatrisen blir da tre steg / omgang * fire omganger = 12 steg.

Merk Dette estimatet inkluderer ikke tid som går med til I/O

3.3 Multiplikasjon av 12x12 matriser.

Det er laget systemer for 12x12 matriser med universell og tilpasset SAP.

3.3.1 Universell SAP



Figur 3-15: System for multiplikasjon av 12x12 matriser med universelle SAP enheter

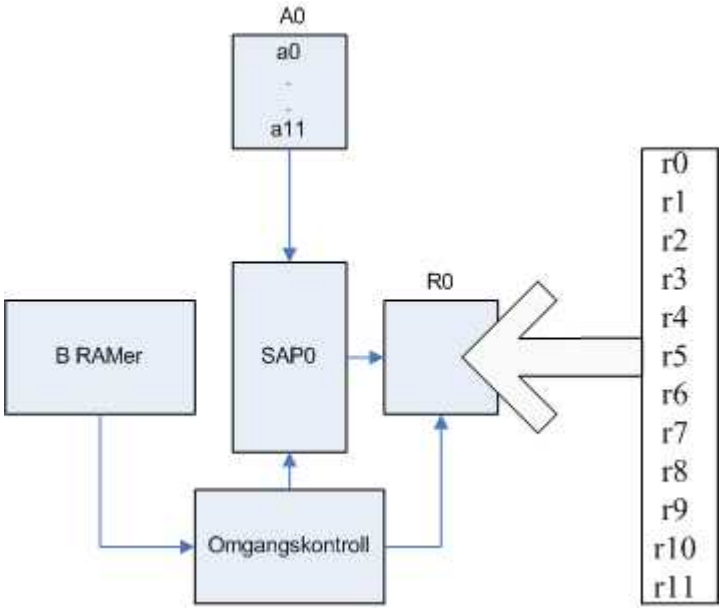
Systemet i Figur 3-15 beregner produktmatrisen R ved å multiplisere 12x12 matrisene A og B. Systemet er basert på systemet i avsnitt 3.2.1. Matrisene A og B lagres i RAMene A0-A11 og B0-B11. Elementene i R beregnes av universelle SAP enheter og lagres i RAMene R0-R11. Organiseringen av matrisene er beskrevet i avnitt 3.1.1.

Som i systemet i kapittel 3.2.1 kan hver SAP enhet bare beregne et element i R av gangen. De 12 SAP enhetene i systemet beregner til sammen 12 elementer av R av gangen. Hele R, som består av 144 elementer må derfor beregnes i løpet av 12 omganger. En omgang for en universell SAP enhet er beskrevet i avsnitt 3.1.9.

Multiplikasjonene styres av to kontrollenheter, RAM adressekontroll og omgangskontroll. RAM adressekontroll kontrollerer hvilke adresser som leses av SAP enhetene fra RAMene A0-A11 og B0-B11. Omgangskontroll kontrollerer hvilken B RAM som er tilknyttet hvilken SAP enhet og i hvilken adresse i R RAM elementene som er beregnet i omgangen lagres, se vedlegg A. Kildekoden for systemet finnes i vedlegg B.3

Funksjon SAP enhet

SAP enhetene er universelle SAP enheter, se 3.1.3 og fungerer på samme måte som SAP enhetene i 3.2.1. Men siden matrisene som skal multipliseres har 12 elementer i rad/kolonne må hvert element i produktmatrisen beregnes med 12 multiplikasjoner og 12 akkumulasjoner.



Figur 3-16: SAP0 i Figur 3-15

For å illustrere hvordan SAP enhetene fungerer skal vi se på hvordan elementet *r0* i produktmatrisen beregnes. *r0* beregnes av SAP0, se Figur 3-16, ved å multiplisere innholdet i RAM A0 med innholdet i RAM B0, se vedlegg A. Innholdet i RAMene multipliseres adresse for adresse og delproduktene legges sammen til *r0* som lagres i RAM R0 i adresse 0.

$$r0 = R0[0] = \sum_{i=0}^{11} A0[i] \cdot B0[i]$$

Fra tabell 1-1 og 1-2 vedlegg A, RAM A0 inneholder første rad i matrise A, RAM B0 første kolonne i matrise B

A0= a0, a1, a2, a3, a4, a5, a6, a7, a8, a9, a10, a11

B0= b0, b12, b24, b36, b48, b60, b72, b84, b96, b108, b120, b132

$$r0 = a0 \cdot b0 + a1 \cdot b12 + a2 \cdot b24 + a3 \cdot b36 + a4 \cdot b48 + a5 \cdot b60 + a6 \cdot b72 + a7 \cdot b84 + a8 \cdot b96 + a9 \cdot b108 + a10 \cdot b120 + a11 \cdot b132$$

Funksjon for systemet

Systemet i Figur 3-15 fungerer på tilsvarende måte som systemet i kapittel 3.2.1, men er utvidet for å multiplisere 12x12 matriser og beregner 12 elementer av R i hver omgang.

Oppbygning

RAM

RAMene i systemet er Blokk RAM som i systemet i kapittel 3.2.1, men A og B RAMene inneholder 12 lokasjoner á 18 bit og R RAMene 12 lokasjoner à 36 bit. Lokasjonene har adresser fra 0 til 11.

RAM adressekontroll

RAM adressekontroll er basert på variabelen i. Verdien på i bestemmer hvilken adresse i A og B RAMene som leses av SAP enhetene, se Tabell 3-11.

Adresse	i verdi
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011

Tabell 3-11: RAM adresse for i-verdier

Omgangskontroll

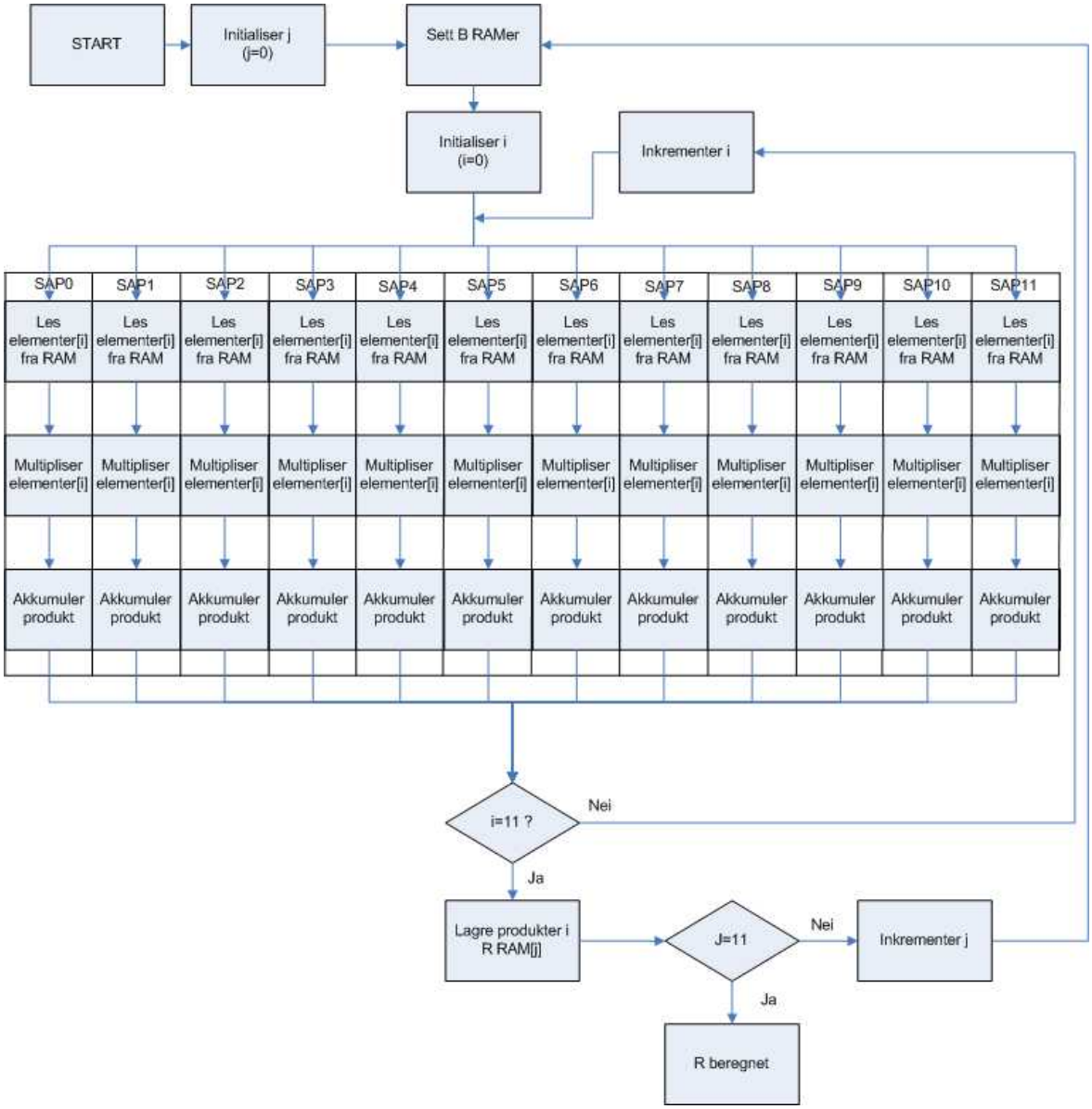
Omgangskontrollen er en 4-12 dekoder basert på variabelen j. Verdien på j avgjør hvilken omgang systemet er i. Hvilken omgang systemet er i avgjør hvilke B RAMer som

tilknyttes hvilke SAP enheter og i hvilken adresse i R RAMene elementene som beregnes i omgangen lagres, se tabell 1.4 og 1.7 i vedlegg A.

SAP enheter

SAP enhetene er universelle SAP enheter, se 3.1.3.

Flyt for systemet



Figur 3-17: Flytskjema for systemet i Figur 3-15

Hver SAP enhet er fast tilknyttet en A Ram og en R RAM, se Figur 3-15. SAP enhetene tilknyttes en B RAMen for hver omgang. Variabelen j kontrollerer hvilken omgang systemet befinner seg i. I hver omgang leser SAP enhetene inn innholdet i den første adressen fra A og B RAMene som er tilknyttet i omgangen og multipliserer dem før

produktet av multiplikasjonen akkumuleres, se Figur 3-17. Dette gjentas til innholdet i alle 12 RAM adressene er lest inn, multiplisert og produktene akkumulert til elementer i R som lagres i R RAMene i adressen bestemt av verdien på variabelen j.

Kommentar

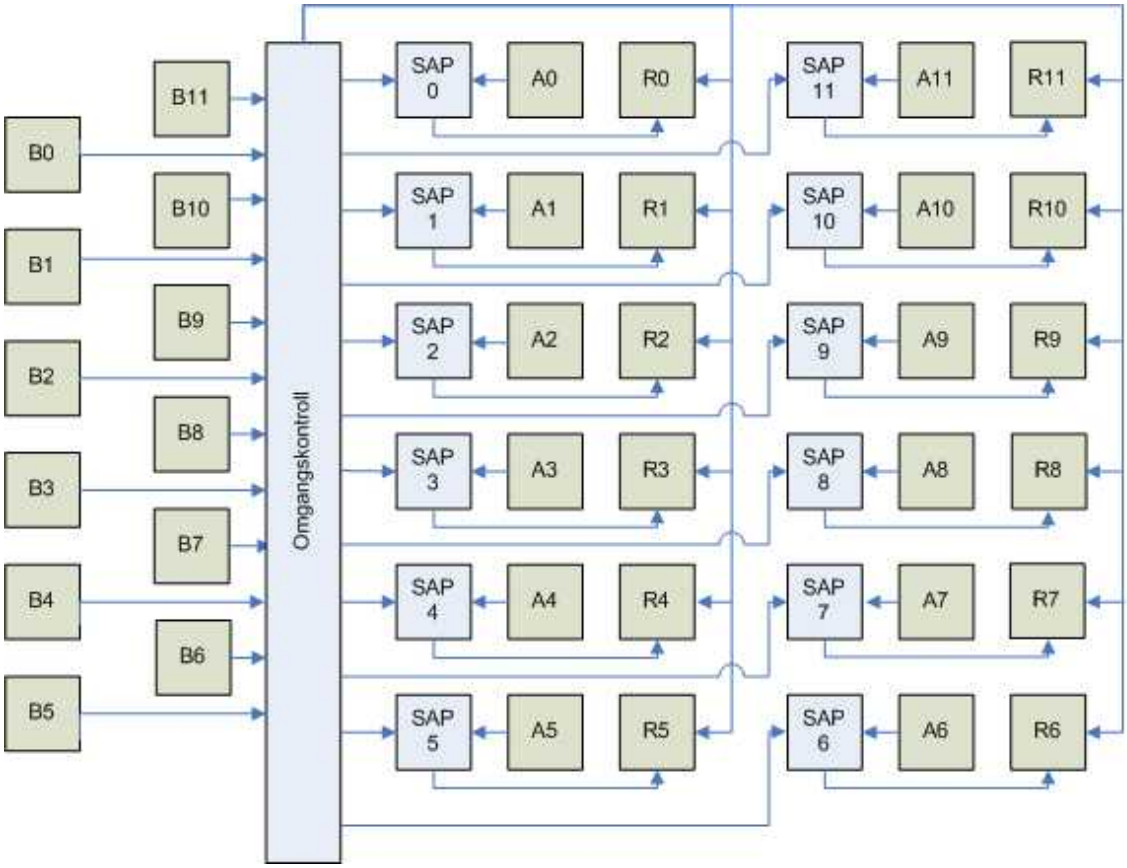
Dette systemet er basert på systemet i 3.2.1. Men er utvidet for å multiplisere 12x12 matriser. Systemet beregner produktmatrisen i løpet av 12 omganger.

Estimert tidsforbruk

Fra 3.1.9 en omgang for et system med universelle SAP enheter er på 2N steg. Med N = 12 blir antall steg pr omgang lik 24. Totalt antall steg for å beregne resultatmatrisen blir 24 steg/omgang*12 omganger = 288.

Merk Dette estimatet inkluderer ikke tid som går med til I/O

3.3.2 Tilpasset SAP



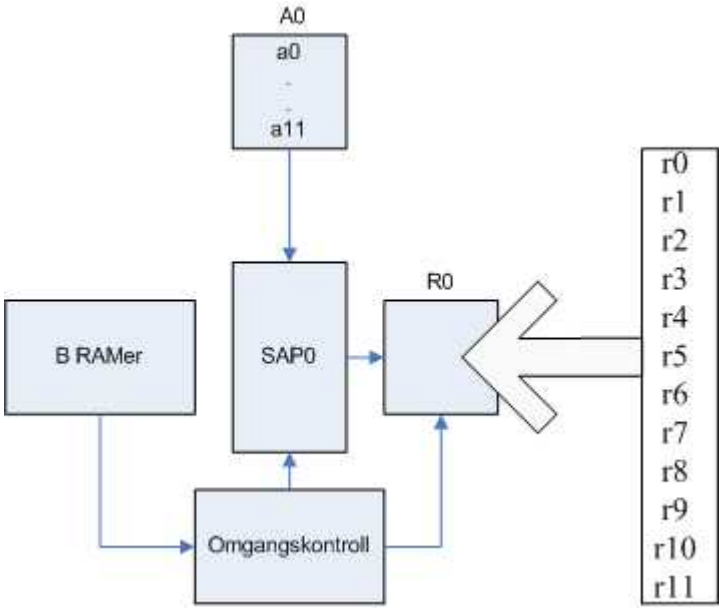
Figur 3-18: System for multiplikasjon av 12x12 matriser med tilpassede SAP enheter

Systemet i Figur 3-18 beregner produktmatrisen R ved å multiplisere to 12x12 matriser A og B. Systemet er basert på systemet i avsnitt 3.2.2, men er utvidet for å multiplisere 12x12 matriser. Matrisene A, B og R er delt i 12 og lagres på samme måte som i avsnitt 3.3.1.

Systemet inneholder 12 tilpassede SAP enheter, disse er tilpasset multiplikasjon av 12x12 matriser. R inneholder 144 elementer og må beregnes i 12 omganger som i systemet i 3.3.1.

Multiplikasjonene styres av omgangskontroll. Omgangskontroll styrer hvilken B Ram som er tilknyttet hvilken SAP enhet og i hvilken adresse i R RAMen det beregnede elementet lagres, se vedlegg A. Kildekoden for systemet finnes i vedlegg B.4

Funksjon for SAP enhetene



Figur 3-19: SAP0 i Figur 3-18

For å illustrere hvordan SAP enhetene i systemet i Figur 3-18 fungerer skal vi se på hvordan elementet r0 i produktmatrisen beregnes. r0 beregnes av SAP0 Figur 3-19.

Systemet beregner r0 på samme måte som systemet i 3.3.1, men multiplikasjonene utføres parallelt før delproduktene legges sammen.

Hver multiplikasjonsblokk leser innholdet i en adresse fra A0 og B0, se Tabell 3-12. Innholdet i adressene multipliseres til 12 delprodukter som legges sammen i addisjonsblokkene til r48.

$$r0 = a0 \cdot b0 + a1 \cdot b12 + a2 \cdot b24 + a3 \cdot b36 + a4 \cdot b48 + a5 \cdot b60 + a6 \cdot b72$$
$$+ a7 \cdot b84 + a8 \cdot b96 + a9 \cdot b108 + a10 \cdot b120 + a11 \cdot b132$$

Addisjonene utføres av addere i en trestruktur.

	Multiplikasjonsblokk											
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
adresse A0	0	1	2	3	4	5	6	7	8	9	10	11
adresse B0	0	1	2	3	4	5	6	7	8	9	10	11

Tabell 3-12: Hvilken multiplikasjonsblokk leser hvilken RAM adresse

Hver SAP enhet er tilknyttet tre RAMer, en A RAM med en rad fra matrise A, en B RAM med en kolonne fra matrise B, denne skiftes for hver omgang og en R RAM for lagring av beregnede R elementer.

Elementene i R beregnes på tilsvarende måte som for systemet i 3.3.1, men innholdet i alle RAM adressene leses til multiplikasjonsblokkene før de multipliseres i parallell. Deretter adderes produktene til R elementet.

Funksjon for systemet

Systemet beregner elementene i produktmatrisen R ved å multiplisere 12x12 matrisene A og B. R beregnes med 12 SAP enheter tilpasset multiplikasjon av 12x12 matriser. Systemet inneholder 12 SAP enheter som hver er fast tilknyttet en A RAM, som inneholder en rad fra matrise A, og en R RAM hvor elementene som beregnes av SAP enheten lagres. Produktmatrisen R består av 144 elementer (12x12), systemet kan beregne en rad (12 elementer) av R i hver omgang. Dette medfører at R må beregnes i 12 omganger. Systemet skifter hvilken B RAM som er tilknyttet hvilken SAP enhet avhengig av verdien for variabelen j, se vedlegg A.

Oppbygning

RAM

RAMene er Blokk RAM, se 2.3.2, som i systemet i 3.2.1, men A og B RAMene inneholder 12 lokasjoner a18 bit og R RAMene 12 lokasjoner à 36 bit. Lokasjonene er adressert fra 0-11

SAP enheter

SAP enhetene er tilpassede SAP enheter, se kapittel 3.1 Tilpasset SAP enhet og 3.3.2 Tilpasset SAP enhet, se Figur 3-19.

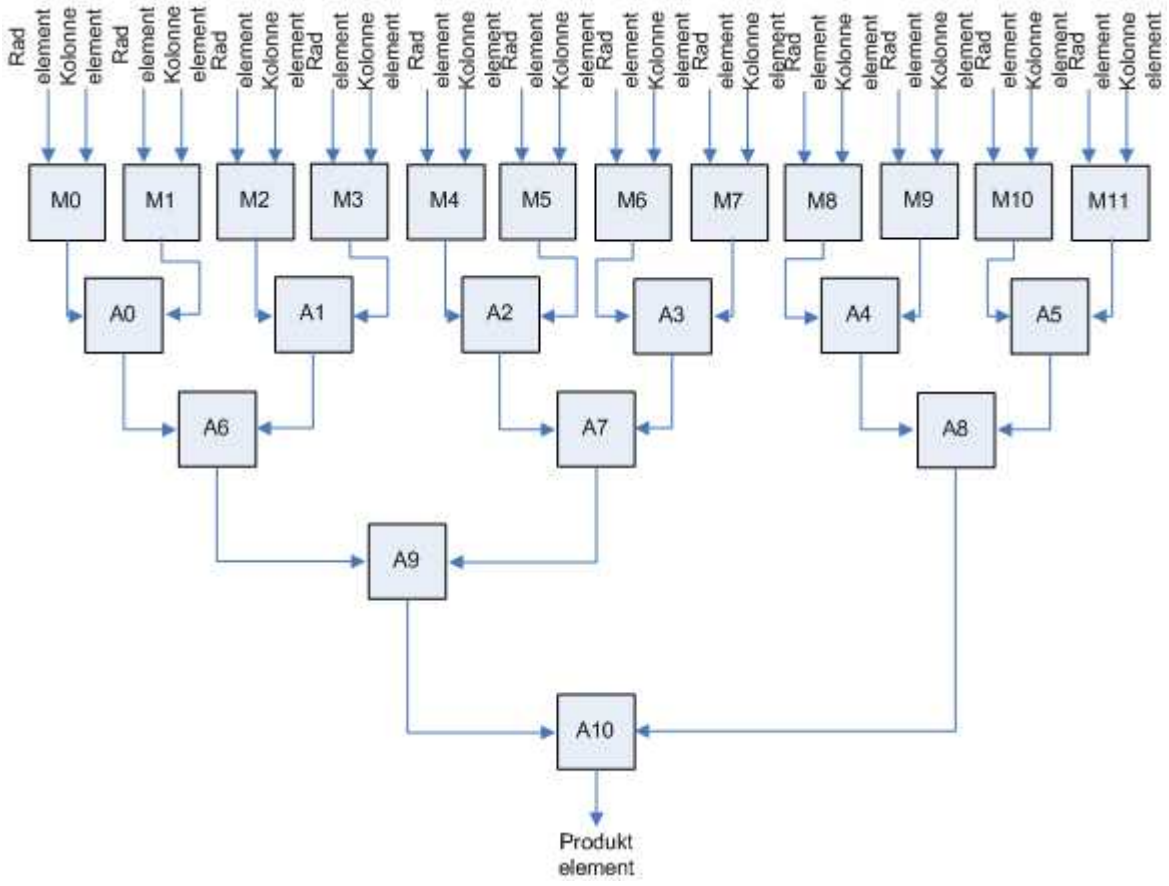
Omgangskontroll

Omgangskontrollen er basert på variabelen j. verdien på j avgjør hvilken omgang systemet befinner seg i og hvilken adresse i R RAMene R elementet beregnet i omgangen lagres i.

Resultatelementene lagres i 12 RAMer à 12x36bit. Tabell 3.14 viser hvordan elementene organiseres i RAMene.

Tilpasset SAP enhet

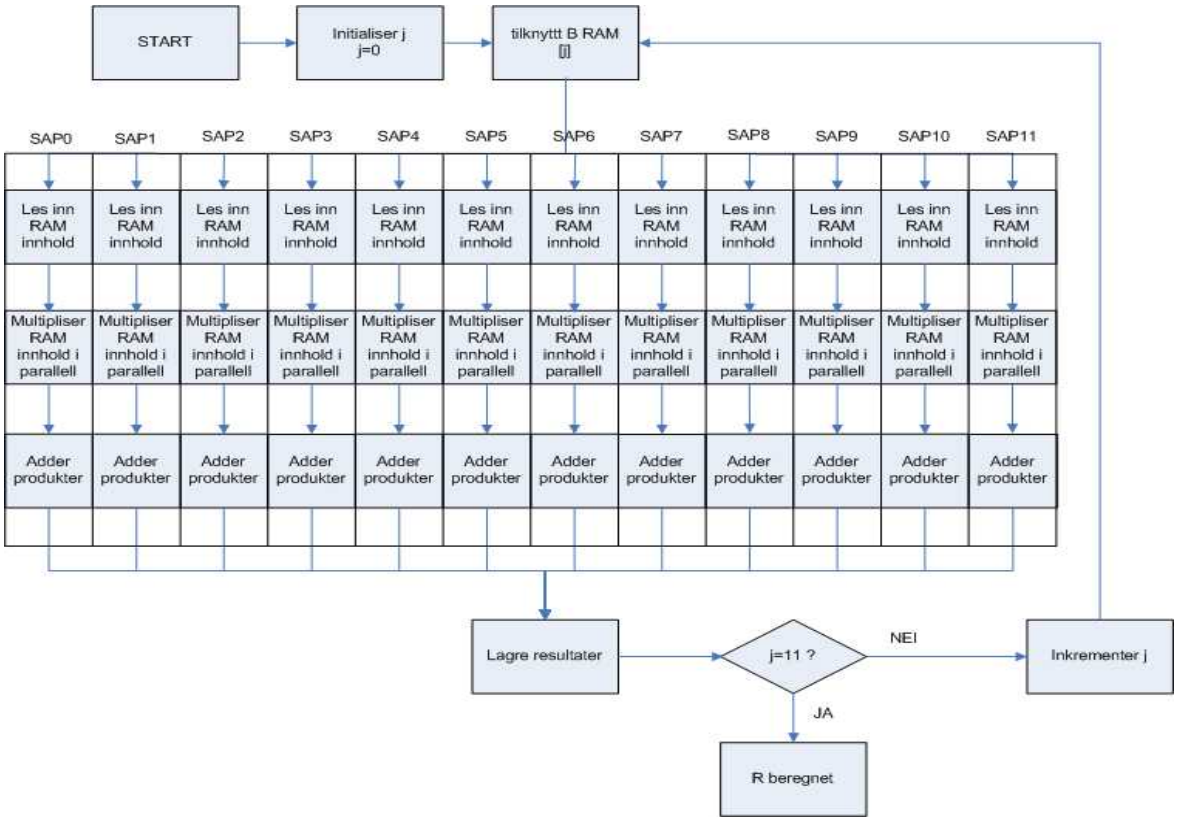
SAP enhetene er tilpassede SAP enheter, se 3.1.4. SAP enhetene er bygd opp som vist i Figur 3-20. Blokkene M0-M11 er 18bit multiplikasjonsblokker, se og blokkene A0-A10 er 36 bits addisjonsblokker, se 3.1.6.



Figur 3-20: Tilpasset SAP enhet for multiplikasjon av 12x12 matriser.

SAP enhetene i systemet i er bygd opp som SAP0 i Figur 3-18 SAP enhetene er tilpasset multiplikasjon av 12x12 matriser og består av 12 multiplikasjonsblokker og 11 addisjonsblokker, se Figur 3-20.

Flyt for systemet



Figur 3-21: Flytskjema for systemet i Figur 3-18.

I hver omgang leser enhetene innholdet i adressene i A og B RAMene til multiplikasjonsblokkene. Innholdet i første adresse leses av første multiplikasjonsblokk, innholdet i andre adresse av andre blokk osv. Innholdet i RAMene multipliseres til delprodukter i multiplikasjonsblokkene før de legges sammen til elementet i R som beregnes av enheten i addisjonsblokkene. Resultatet av addisjonene, produktelementet lagres i den tilknyttede R RAMen i adresse samsvarende med omgangen systemet befinner seg i, se Figur 3-21. Dette gjentas til alle elementene i produktmatrisen er beregnet.

Kommentarer

Dette systemet beregner resultatmatrisen med tilpassede SAP enheter. Produktmatrisen beregnes i løpet av 12 omganger som for systemet i 3.3.1.

Estimert tidsforbruk

En omgang for SAP enheter tilpasset 12x12 matriser inneholder fem steg, se 3.1.9. Totalt antall steg for å beregne produktmatrisen blir da: fem steg/omgang*12 omganger = 60 steg.

Merk Dette estimatet inkluderer ikke tid som går med til I/O

4 RESULTATER

Systemene i kapittel 3 er laget for en XC2V6000 FPGA. Det er kun foretatt test av funksjon og om systemene får plass på FPGAen.

Grunnet tidsnød i slutten av prosjektet og problemer med programvare er I/O for systemene ikke implementert og testet. Systemene heller ikke testet mot mikroprosessor som kjører et C program for multiplikasjon av matriser.

Resultatene som presenteres her er forbruk av ressurser i FPGAen mot totale tilgjengelige ressurser.

Fra 2.3 ressurser i XC2V6000 FPGA

LUT	67584 stk
Vipper	67584 stk
Block RAM	2654208 bit
Multiplikasjonsblokker	144 stk
Max hastighet[2]	180 MHz

I kapittel 4.1 presenteres resultatene for systemene i kapittel 3.2. Og i 4.2 presenteres resultatene for systemene i 3.3. Resultatene diskuteres i kapittel 4.3.

Merk Forbruket av ressurser for systemene er svært lavt og det tas forbehold om at deler av systemene er blitt borte under kompilering.

Merk Det beregnede tidsforbruket inkluderer ikke tidsforbruk til I/O. I/O vil trolig kreve mer tid enn selve beregningen av elementene i produktmatrisen. Men da I/O ikke er implementert og testet er tidsforbruk til I/O ikke tatt med i estimert tidsforbruk.

4.1 Resultater for systemer for multiplikasjon av 4x4 matriser

Ressursforbruket for systemene i 3.2 presenteres i Tabell 4-1

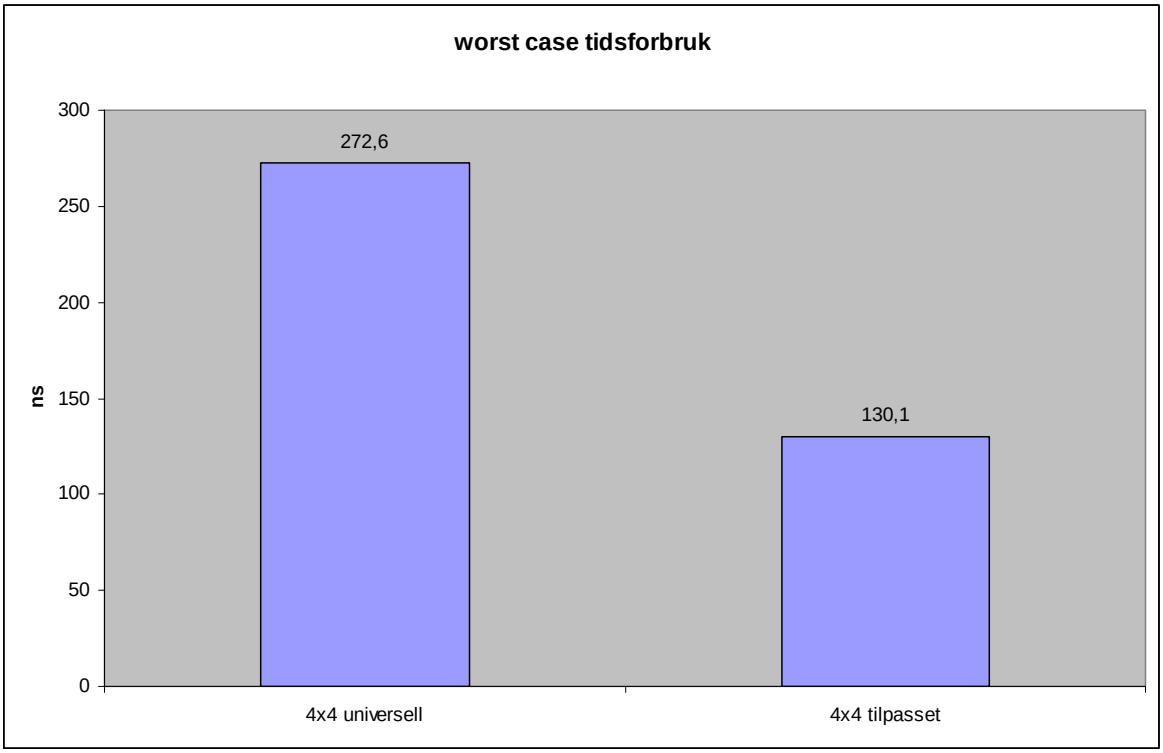
	LUT	**vipper**	**forsinkelse**(ns)	**RAM**(bit)
4x4 universell SAP	19	52	8.53	1152
4x4 tilpasset SAP	31	181	10.91	1152

Tabell 4-1: Forbruk for 4x4 systemene

[2] Gjelder multiplikasjonsblokker i XC2V6000-4 (speed grade 4) 22. Xilinx, *High-Speed Data Serialization and Deserialization (840 Mb/s LVDS)*, N. Sawyer, Editor. 2002.

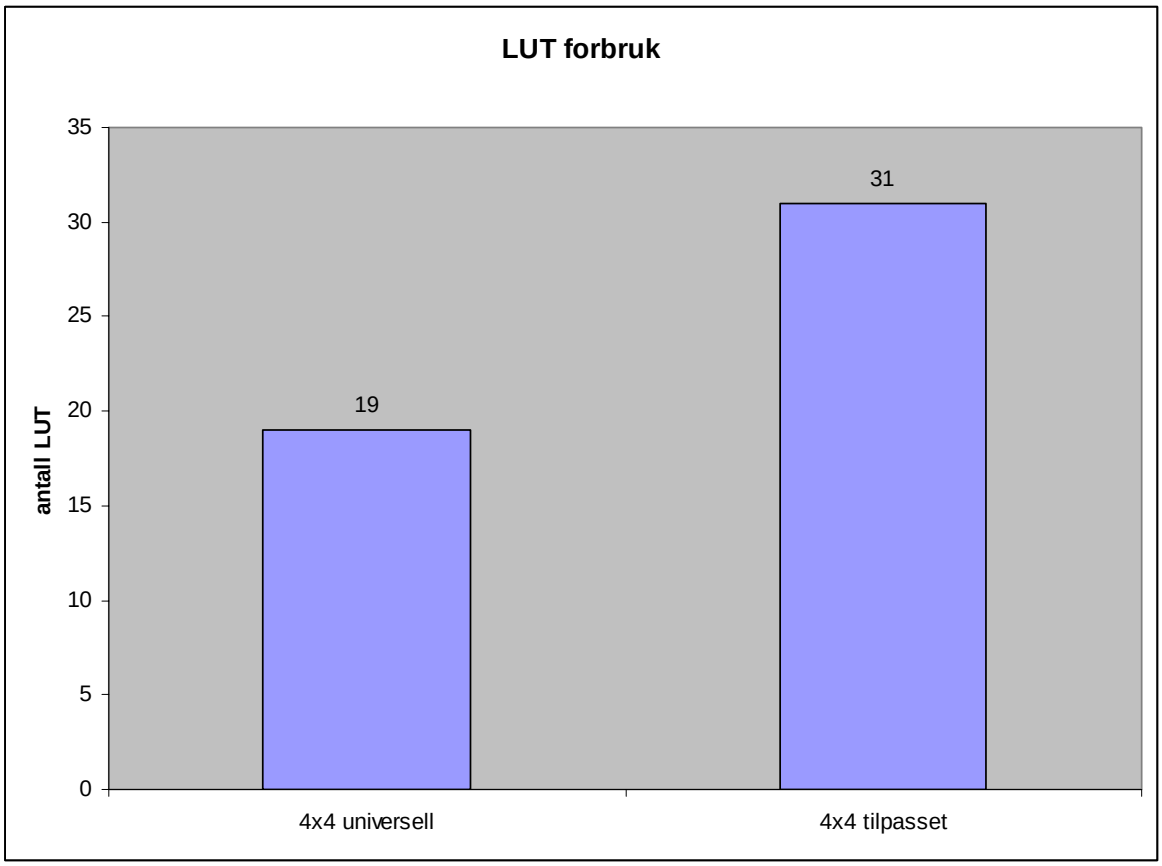
Tidsforbruk

Worst Case" tidsforbruk for 4x4 systemene kan estimeres ved å bruke forsinkelsene i Tabell 4-1 (disse er maksimal forsinkelse i systemet) og antall steg som trengs for å beregne produktmatrisen for systemene. For universell SAP 32 steg og tilpasset SAP 12 steg. Tidsforbruk for universell SAP blir da 32*8.53 ns = *272.6* ns og for tilpasset SAP 12*10.91 ns = *130.1* ns.



Figur 4-1:Worst case tidsforbruk for 4x4 systemene

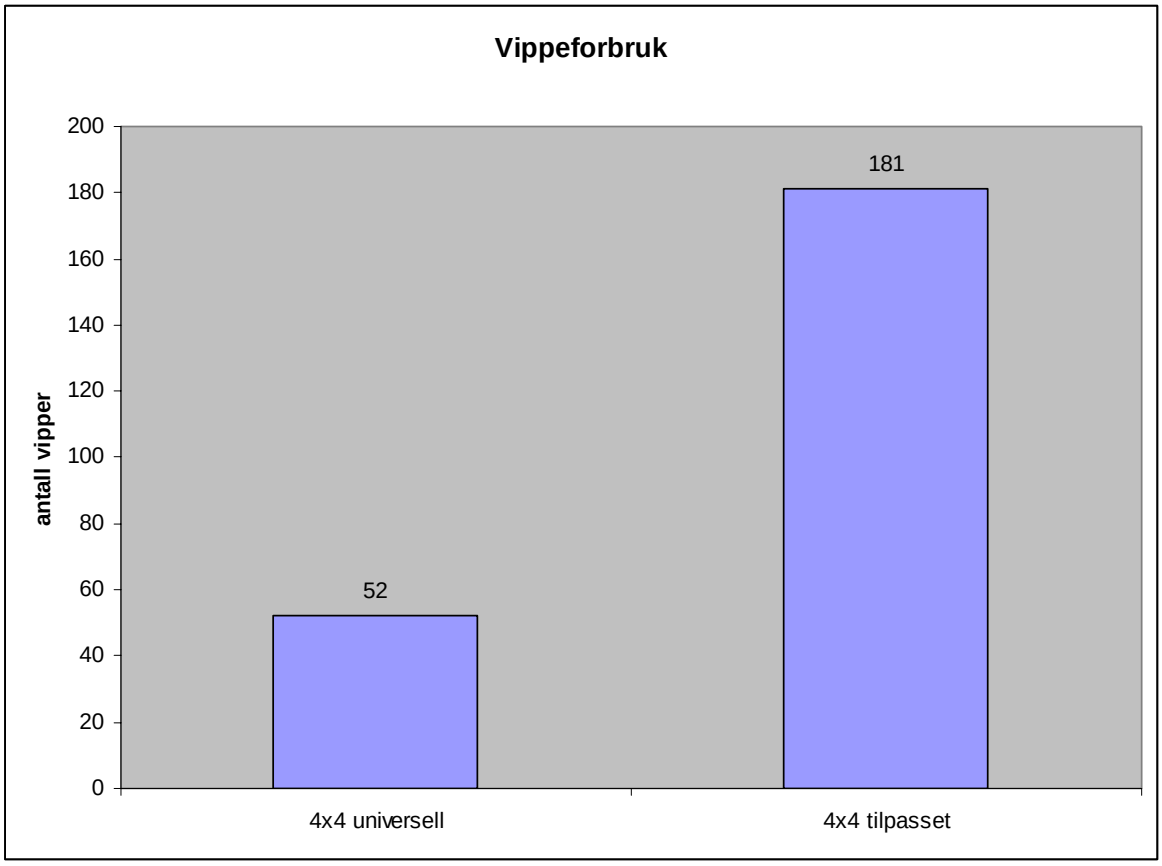
LUT forbruk



Figur 4-2:LUT forbruk for systemene i kapittel 3.2

Systemet med universelle SAP enheter bruker 0,03 % og systemet med tilpassede SAP enheter0,045 % av totalt antall LUT i FPGAen.

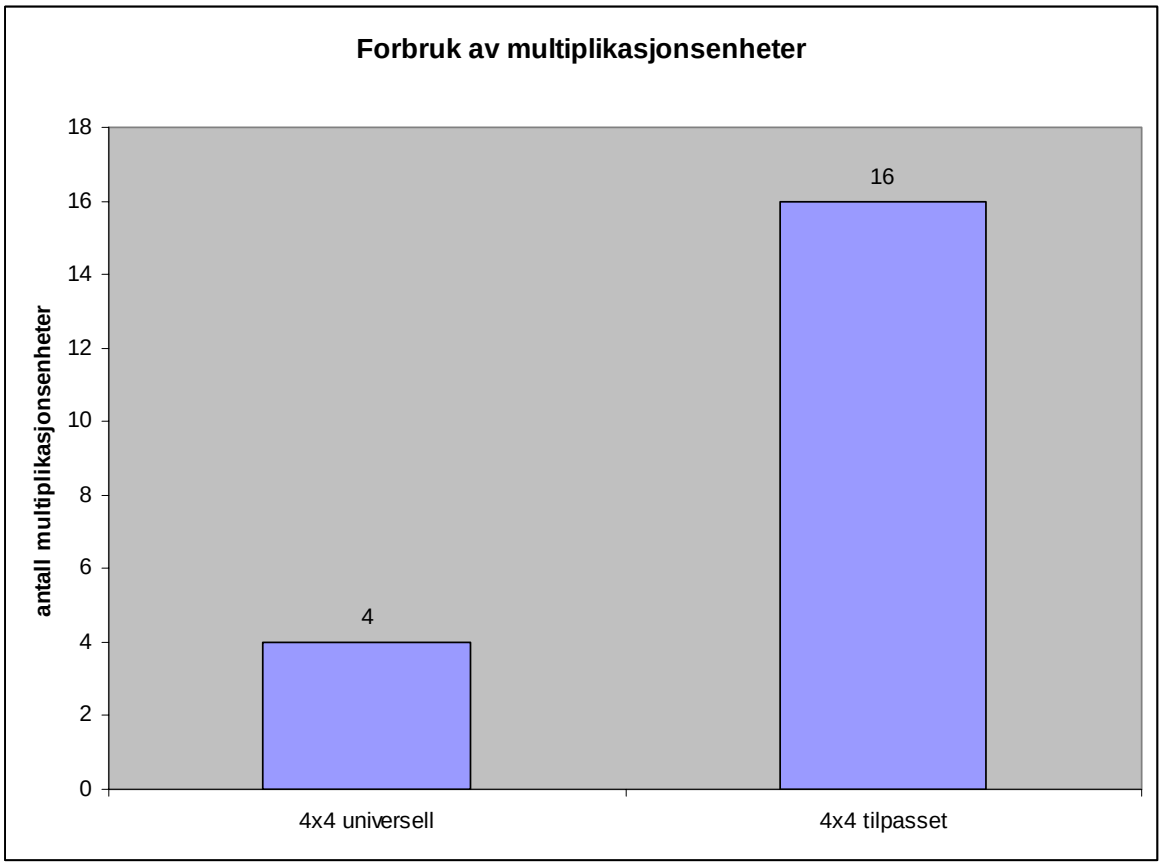
Vippeforbruk



Figur 4-3:Vippe forbruk for systemene i kapittel 3.2

Systemet med universelle SAP enheter bruker 0,076 % og systemet med tilpassede SAP enheter 0,3 % av totalt antall vipper i FPGAen.

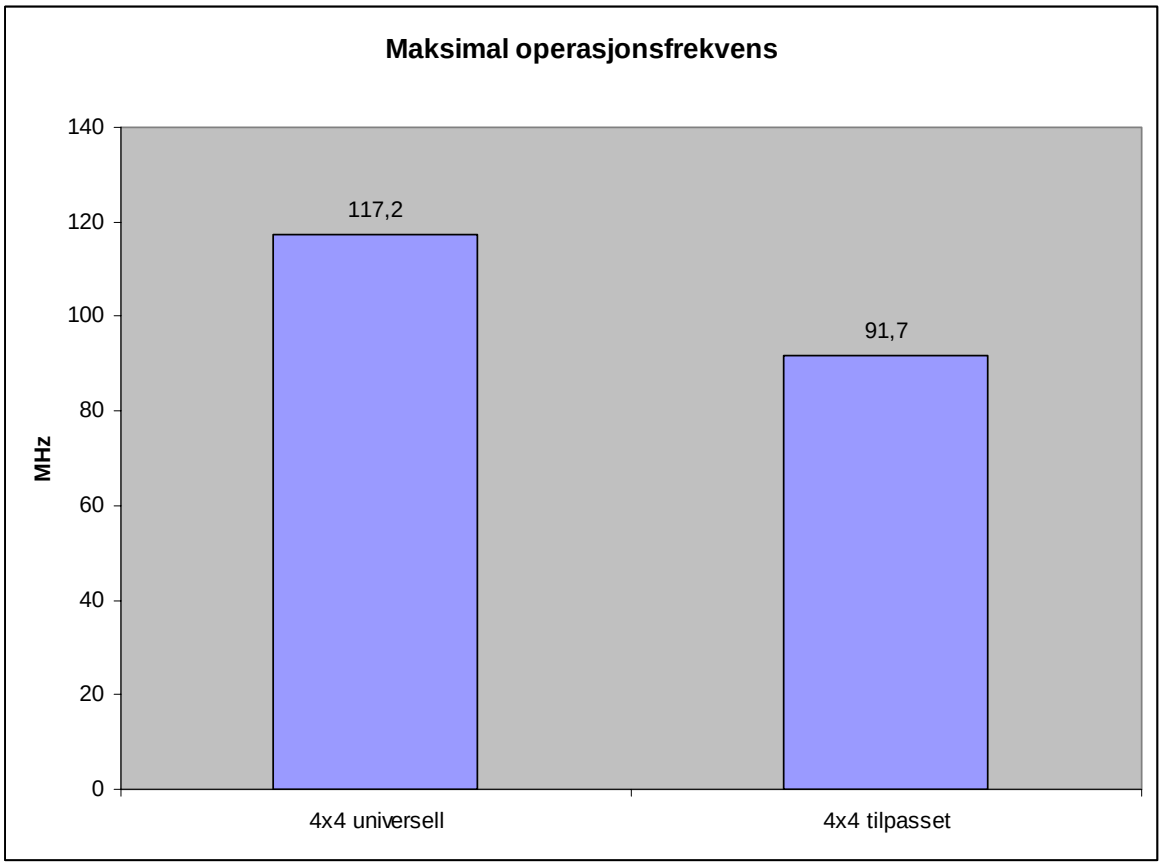
Forbruk av multiplikasjonsblokker



Figur 4-4: Forbruk av multiplikasjonsblokker for systemene i kapittel 3.2

Systemet med universelle SAP enheter bruker 2,8 % og systemet med tilpassede SAP enheter 11 % av totalt antall multiplikasjonsblokker i FPGAen.

Maksimal operasjonsfrekvens



Figur 4-5:Maksimal operasjonsfrekvens for systemene i 3.2

Disse operasjonsfrekvensene er innenfor maksimal ytelse for multiplikasjonsenhetene i FPGAen [23].

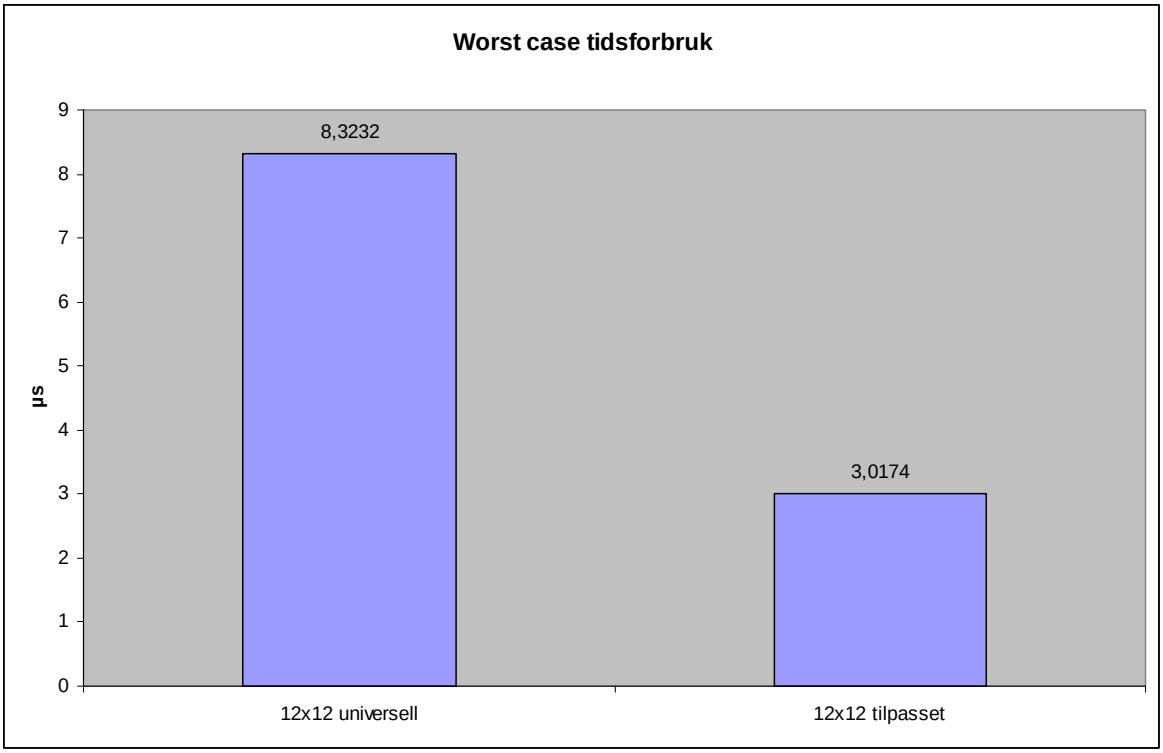
4.2 Resultater for systemer for multiplikasjon av 12x12 matriser.

	LUT	vipper	delay (ns)	RAM (bit)
12x12 universell	80	88	28,91	10368
12x12 tilpasset	170	690	50,29	10368

Tabell 4-2: Forbruk for 12x12 systemene

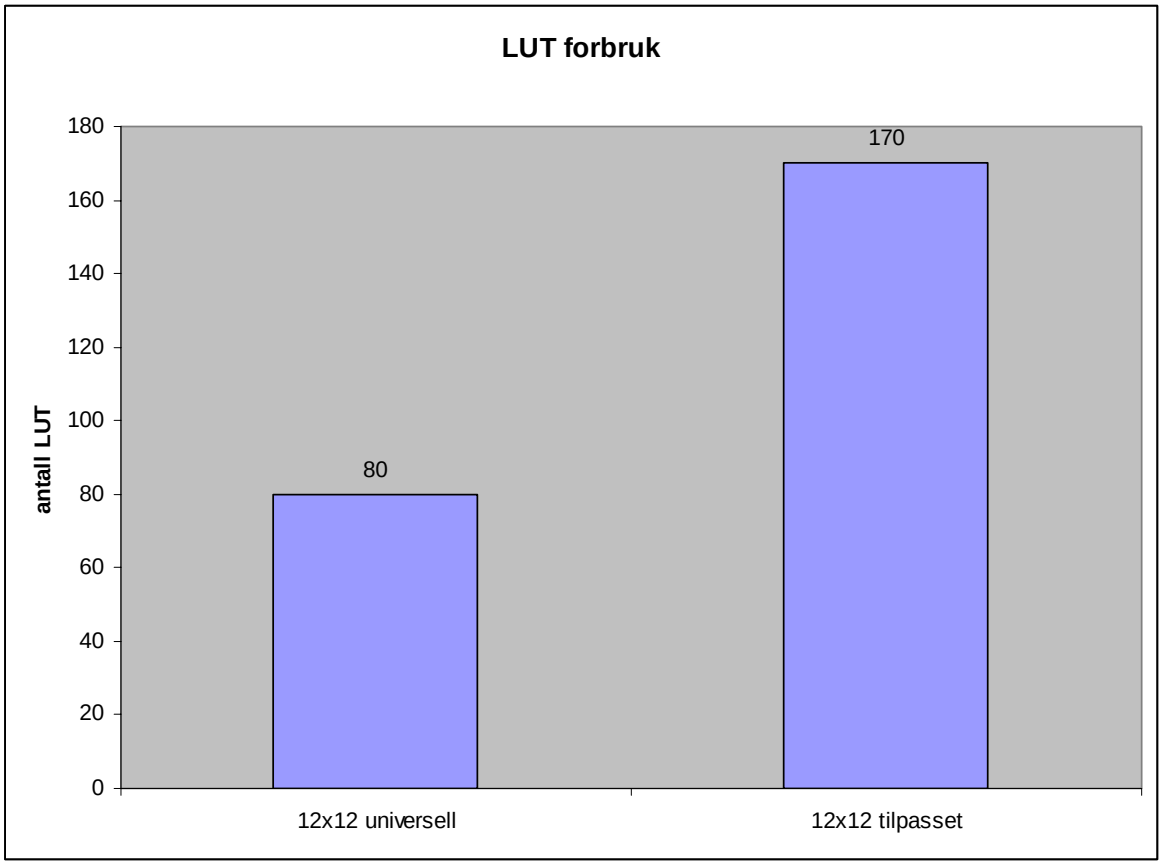
Tidsforbruk

Worst Case tidsforbruk for 12x12 systemene kan estimeres ved å bruke forsinkelsene i Tabell 4-2 (maksimale forsinkelser i systemene) og antall steg som trengs for å beregne produktmatrisen for systemene fra avsnitt 3.3.1 og 3.3.2. Antall steg for universell SAP 288 steg og for tilpasset SAP 60 steg. Tidsforbruk for universell SAP blir da 288*28,91 ns = *8323,2* ns (8,32 µs) og for tilpasset SAP 60*50,29 ns = *3017.4* ns (3.02 µs).



Figur 4-6: Worst case tidsforbruk for 12x12 systemene

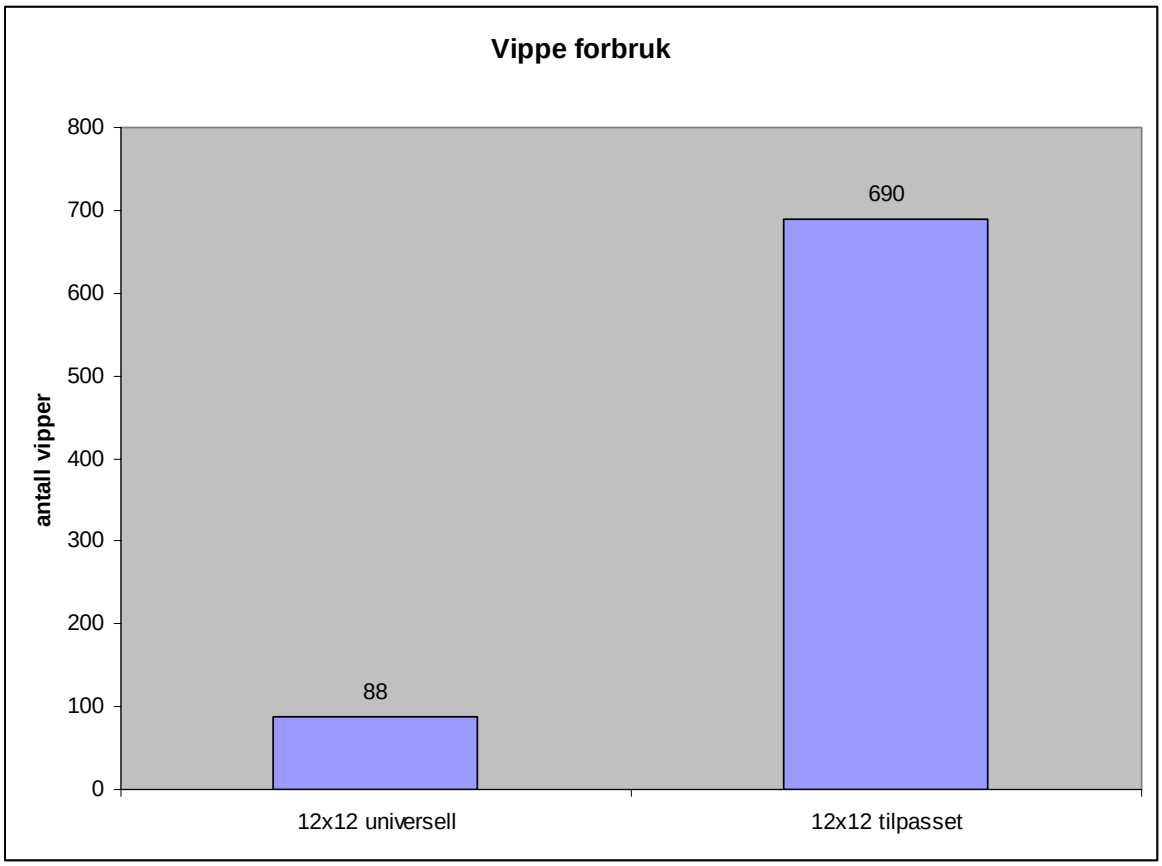
LUT forbruk



Figur 4-7: LUT forbruk for systemene i kapittel 3.3

Systemet med universelle SAP enheter bruker 0,11 % og systemet med tilpassede SAP enheter 0,25 % av totalt antall LUT i FPGAen.

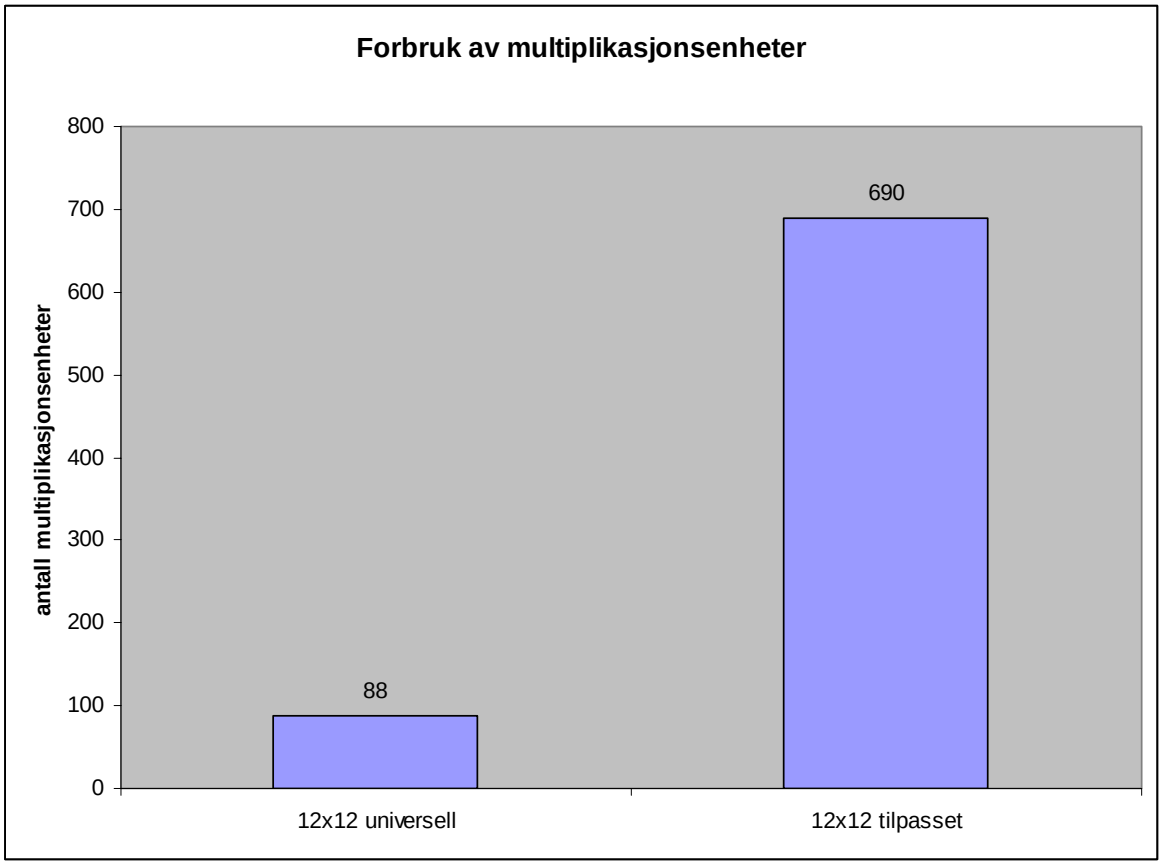
Vippe forbruk



Figur 4-8:Vippe forbruk for systemene i kapittel 3.3

Systemet med universelle SAP enheter bruker 0,13 % og systemet med tilpassede SAP enheter 1 % av totalt antall vipper i FPGAen.

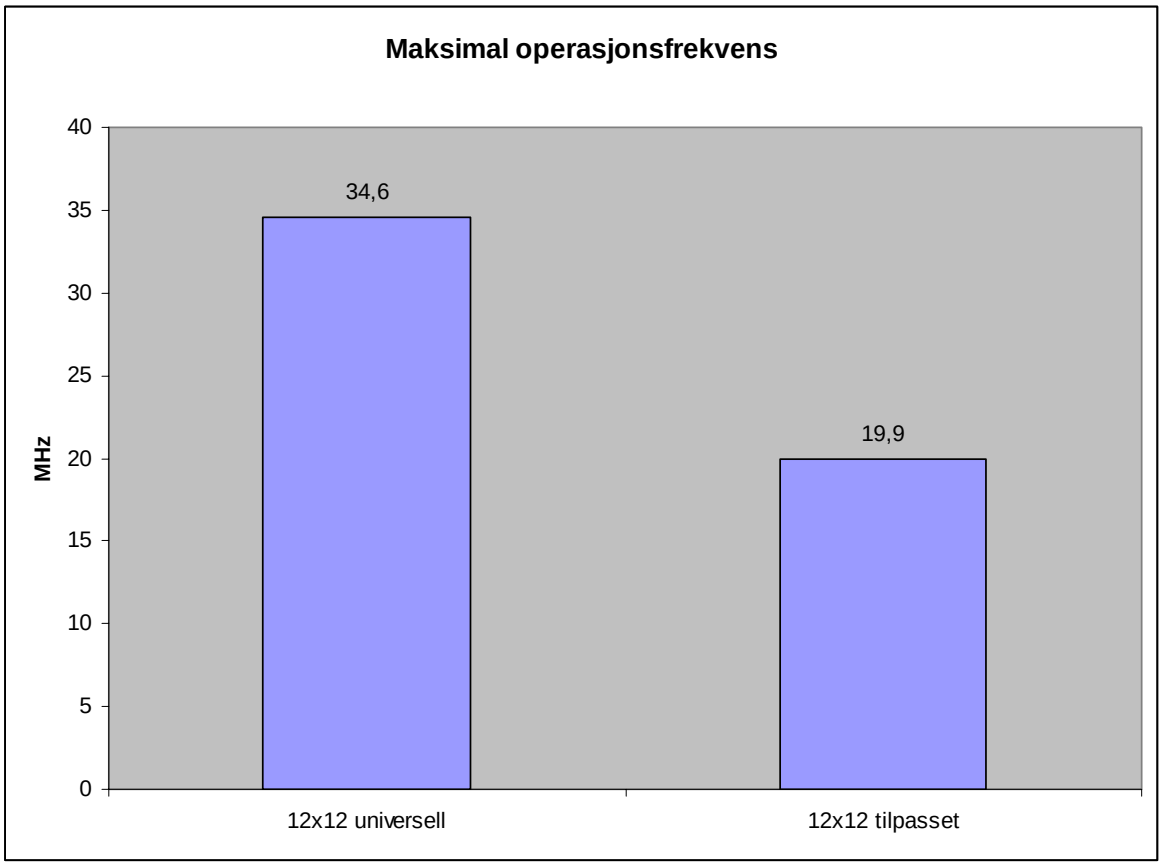
Forbruk av multiplikasjonsblokker



Figur 4-9: Forbruk av multiplikasjonsblokker for systemene i kapittel 3.3

Systemet med universelle SAP enheter bruker 8,3 % og systemet med tilpassede SAP enheter 100 % av totalt antall multiplikasjonsblokker i FPGAen.

Maksimal operasjonsfrekvens



Figur 4-10: Maksimal operasjonsfrekvens for systemene i 3.3.

4.3 Diskusjon

Systemene i kapittel 3 er konstruert for multiplikasjon av NxN matriser med størrelse 4x4 og 12x12. I systemene er det benyttet to metoder for å beregne elementene i produktmatrisen, universell og tilpasset SAP.

Som bemerket i begynnelsen av kapittel 4 er resultatene for systemene i kapittel 3 presentert med all mulig forbehold om at deler av systemene er falt bort under kompilasjon og at tidsforbruk for I/O ikke er tatt med i det estimerte tidsforbruket. Det må og bemerkes at en omgang for

Systemet for multiplikasjon av 4x4 matriser med universelle SAP enheter bruker lite av ressursene i FPGAen. Av LUT og vipper brukes 0,028 og 0,077 %, av interne multiplikasjonsblokker 2,78 % og 0,043 % av Blokk RAM. Systemet oppnår en maksimal operasjonsfrekvens på 117,2 MHz og estimert tidsforbruk for å beregne produktmatrisen på 272,6 ns.

Sammenliknes resultatene for systemet med resultatene for systemene i [11] og [12] for multiplikasjon av 4x4 matriser oppnår dette systemet høyere operasjonsfrekvens enn systemet [12], men lavere enn systemet i [11]. Disse systemene oppnår operasjonsfrekvenser på 60 og 166.5 MHz.

At systemet oppnår høyere operasjonsfrekvens enn systemet i [12] er trolig grunnet bruk av nyere FPGA og interne multiplikasjonsblokker. At systemet oppnår lavere operasjonsfrekvens enn systemet i [11] kan tilskrives flere faktorer, systemet multipliserer matriser med større bitbredde pr element (8 Vs 18 bit elementer)og at systemet er skrevet i Händel C og derfor trolig kan optimaliseres ved konvertering til VHDL eller liknende.

Systemet for multiplikasjon av 4x4 matriser med tilpassede SAP enheter bruker noe mer av ressursene i FPGAen enn systemet som bruker universelle SAP enheter, men bruker 0,046 % av LUT i FPGAen og 0,27 % av vipper, 11,1 % interne multiplikasjonsblokker og 0,043 % av Blokk RAM. Systemet oppnår en maksimal operasjonsfrekvens på 91,7MHz og estimert tidsforbruk for å beregne produktmatrisen er 130,1 ns.

Om systemet sammenliknes med systemet for multiplikasjon av 4x4 matriser med universelle SAP enheter bruker dette systemet 4 ganger så mange interne multiplikasjonsblokker, men beregner produktmatrisen med 20 færre steg.

Sammenliknes systemet med systemene i [11] og [12] oppnår systemet i likhet med systemet med universelle SAP enheter høyere operasjonsfrekvens enn systemet i [12] og

lavere enn systemet i [11]. Grunnene til dette er de samme som for systemet med universelle SAP enheter da begge systemene er laget for samme FPGA i samme verktøy (DK 3.1).

Tidsforbruket for systemene er ikke sammenliknet med systemene i [11] og [12] fordi detaljer om tidsforbruket ikke er oppgitt for disse systemene.

Systemet for multiplikasjon av 12x12 matriser med universelle SAP enheter bruker som systemene for multiplikasjon av 4x4 matriser lite av ressursene i FPGAen. Av LUTer brukes 0,12 % og av vipper brukes 0,13 %, multiplikasjonsenhetene 8,3 % og 0,39 % av Blokk RAM. Systemet oppnår en maksimal operasjonsfrekvens på 34,6MHz og estimert tidsforbruk for å beregne produktmatrisen er 8,3 µs.

Om systemet sammenliknes med systemet i [14] for multiplikasjon av 12x12 matriser oppnås lavere operasjonsfrekvens (34,6 mot 74 MHz). Estimert tidsforbruk for systemet er lavere enn for systemet i [14] (307,2 µs mot 8,3 µs). Forbruk av LUT er også lavere (80 mot 223).

Systemet for multiplikasjon av 12x12 matriser med tilpassede SAP enheter bruker mer av ressursene i FPGAen enn systemet med universelle SAP enheter. Systemet bruker 0,25 % av LUT, 1,02 % av vipper, 100 % av interne multiplikasjonsblokker og 0,39 % av Blokk RAM. Systemet oppnår en operasjonsfrekvens på 19,9 MHz og estimert tidsforbruk for å beregne produktmatrisen er 3,02 µs.

Om systemet sammenliknes med systemet for multiplikasjon av 12x12 matriser med universelle SAP enheter bruker dette systemet 12 ganger flere interne multiplikasjonsblokker, men beregner produktmatrisen med 228 færre steg.

Om systemet sammenliknes med systemet i [14] oppnår det lavere operasjonsfrekvens (19,9 mot 74MHz), men estimert tidsforbruk er lavere enn for systemet i [14] (307,2 µs mot 3,02 µs). Forbruket av LUT er også lavere (170 mot 223).

Operasjonsfrekvensene for alle systemene i kapittel 3 samsvarer med [23] hvor de interne multiplikasjonsenhetene i en Virtex-II XC2V6000-4 (speed grade 4) oppnår en maksimal operasjonsfrekvens på 180 MHz for 18x18 bit multiplikasjon med maksimal "pipelining".

Forbruket av ressurser i FPGAen er for alle systemene lavt, på det meste 1 og 4 % for LUT og vipper. Systemene bruker lite av den tilgjengelige Blokk RAMen i FPGAen 0,4 og 4 ‰ for 4x4 og 12x12 systemene.

Men det må igjen bemerkes at ressursforbruket i systemene kan være så lavt som det er på grunn av at deler av systemene kan ha falt bort under kompilasjon og at det estimerte

tidsforbruket for systemene ikke inkluderer tidsforbruk for I/O. Det faktiske tidsforbruket vil sannsynligvis være høyere.

En grunn til at operasjonsfrekvensene er i den størrelsesorden de er kan være at systemene er laget med Händel C. I [15] sammenliknes to like systemer for matrisemultiplikasjon, et laget i Händel C og et laget i VHDL. Systemet laget i VHDL har omlag 15 ganger høyere operasjonsfrekvens (182MHz mot 12.6 MHz) enn systemet laget i Händel C. Det må bemerkes at systemet i [15] er laget i en tidlig utgave av Händel C, mens systemene i denne rapporten er laget i nest siste utgave av Händel C. Men bruken av Händel C antas å ha hatt innvirkning på ytelsen for systemene.

5 VIDERE ARBEID

Arbeidet med bruk av FPGA til matrisemultiplikasjon er for min del avsluttet, men langt fra ferdig. I dette kapittelet skisseres noen mulige utvidelser og forbedringer av systemene i kapittel 3. Jeg vil fremheve at dette på ingen måte er en komplett liste over de forbedringer som er mulige, men håper de gir en oversikt over mulige utvidelser og forbedringer.

Det første som bør gjøres er å implementere en effektiv I/O løsning, dette er ikke gjort pga tidsnød i avslutningen av oppgaven og problemer med programvaren som er brukt.

Med I/O løsningen på plass kan tidsforbruket for matrisemultiplikasjonen måles og sammenliknes med tidsforbruket for matrisemultiplikasjon av matriser med samme størrelse i en mikroprosessor.

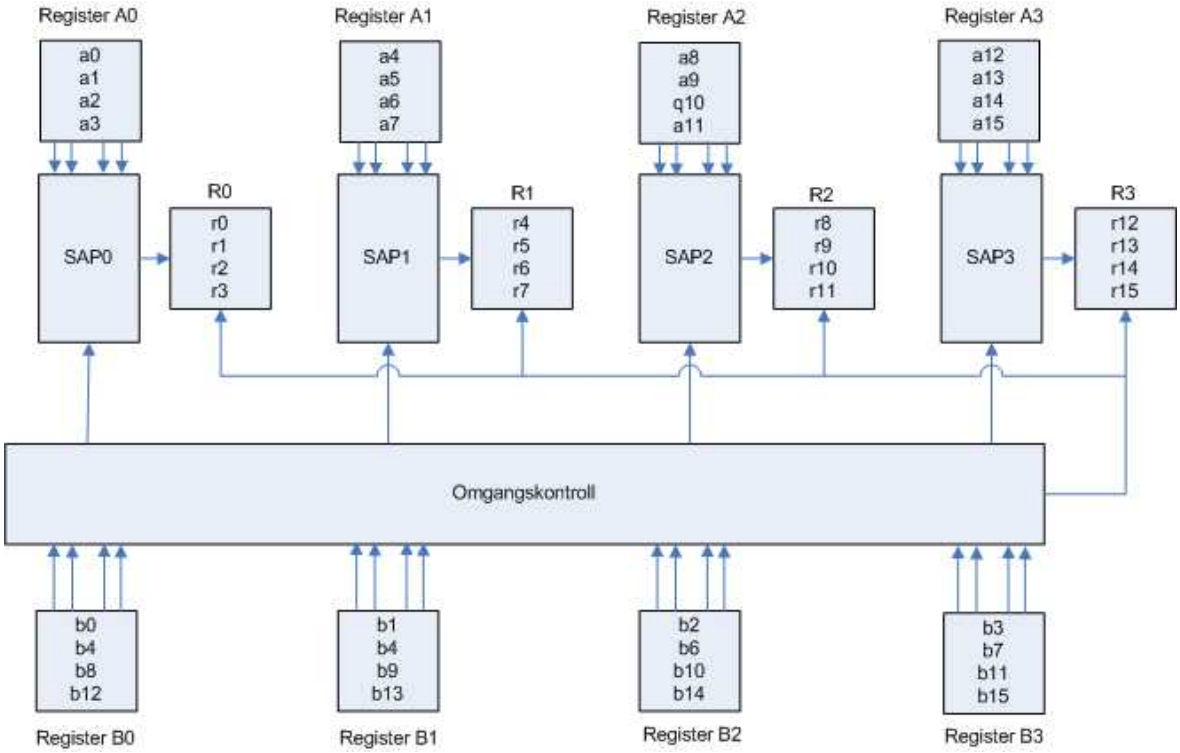
5.1 Forslag til utvidelser og forbedringer

VHDL

For å kunne optimalisere ytelsen for systemene er det ønskelig å konvertere dem til VHDL. En slik konvertering vil gi større kontroll over systemets oppbygning og større muligheter til å justere systemets ytelse. Ulempen med å konvertere systemene til VHDL er at fokuset flyttes fra overordnet nivå (dataflyt og multiplikasjonsoperasjonen) til konsekvenser av endringer gjort på lavt (port) nivå i FPGAen.

Bytte ut RAM med registre i systemene med tilpasset SAP

Systemene med tilpasset SAP leser inn alle elementene i rad/kolonne som skal multipliseres før elementene multipliseres. Når RAM brukes må elementene i rad/kolonne leses et og et inn i SAP enhetene. Om RAM byttes ut med registre hvor elle elementene kan leses samtidig som vist i Figur 2-1 kan tidsforbruket for innlesing av elementene reduseres.



Figur 5-1:System for multiplikasjon av 4x4 matriser med tilpassede SAP enheter hvor RAM er byttet ut med registre

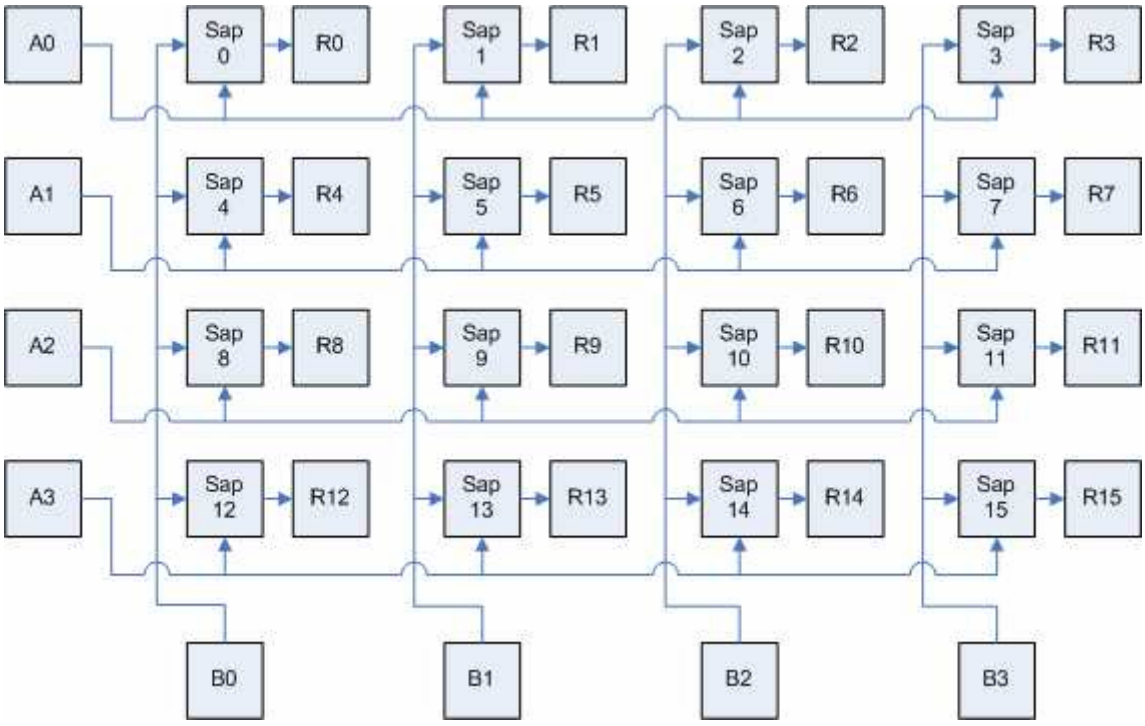
Om elementene lagres i registre kan alle elementene leses inn samtidig istedenfor å lese elementene et og et

Flyttall

Flyttalsmultiplikasjon av matriser er en grunnleggende operasjon i mange vitenskaplige kalkulasjoner [15]. Å konvertere systemene til å kunne multiplisere matriser med flyttallselementer vil medføre at elementene må lagres på en annen måte, de vil kreve mer plass pr element og SAP enhetene må endres for å håndtere flyttall.

Utvidelse av system for multiplikasjon av 4x4 matriser med tilpassede SAP enheter

Systemet for multiplikasjon av 4x4 matriser med tilpassede SAP enheter i kapittel 3.2.2 kan utvides fra fire SAP enheter til 16, se Figur 5-2. Systemet vil da inneholde en SAP enhet pr element i produktmatrisen og denne kan beregnes i løpet av en omgang.



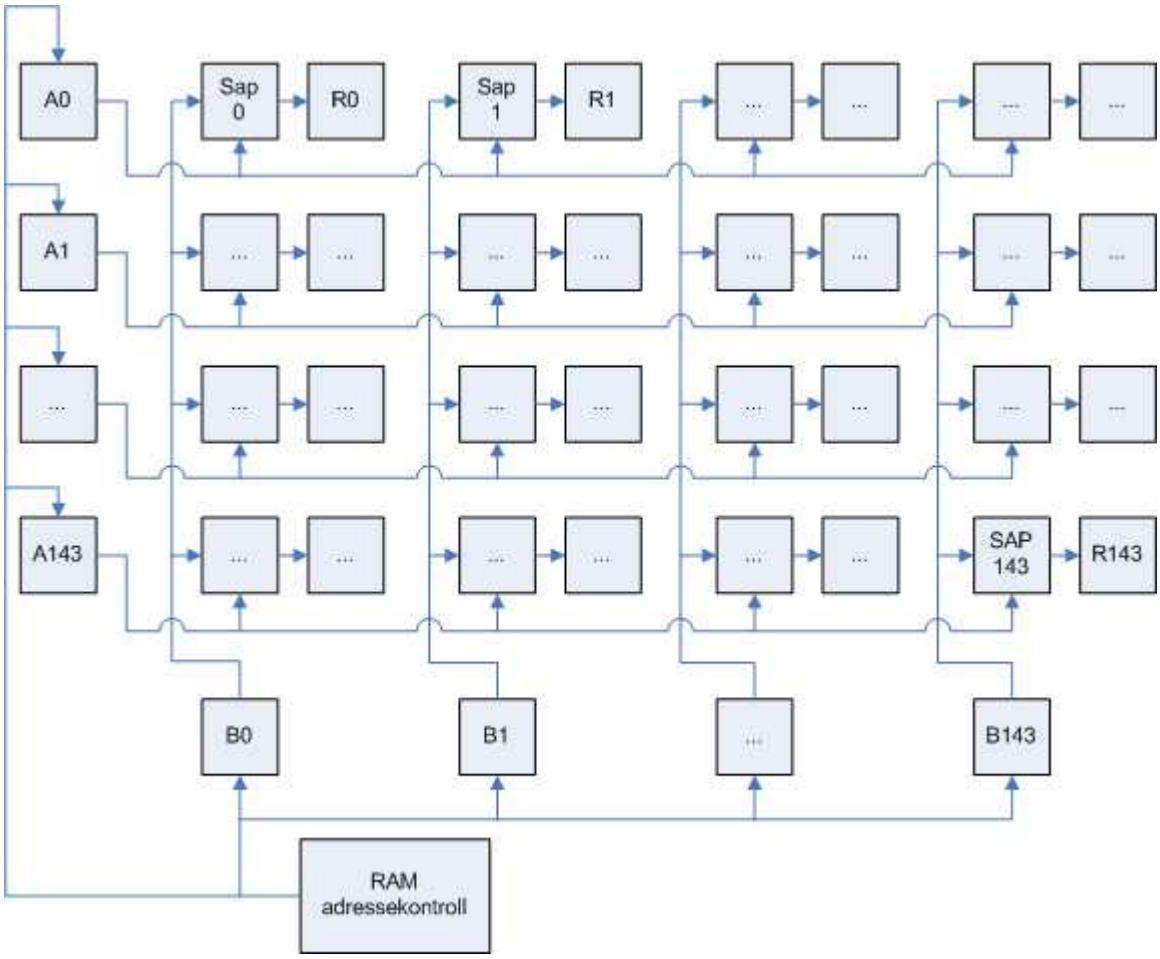
Figur 5-2: Mulig utvidelse av system for multiplikasjon av 4x4 matriser

Effektiv I/O blir avgjørende for om systemet kan brukes effektivt. Bruk av LVDS eller PCI-X I/O som skissert i avsnitt 3.1.8 bør kunne oppnå høy nok overføringsrate til at I/O ikke blir en for stor flaskehals i systemet.

Å implementere multiplikasjon av NxN matriser på denne måten vil kreve mye ressurser. Hver SAP enhet inneholder N interne multiplikasjonsblokker og systemet inneholder N^2 SAP enheter. Totalt vil systemet inneholde N^3 interne multiplikasjonsblokker. Systemet vil også kreve 2N RAMer som hver inneholder N lokasjoner for matrisene som skal multipliseres og N^2 RAMer som hver inneholder 2N lokasjoner for produktmatrisen.

Utvidelse av system for multiplikasjon av 12x12 matriser med universelle SAP enheter

Systemet for multiplikasjon av 12x12 matriser med universelle SAP enheter kan utvides på samme måte som det foreslåtte systemet i Figur 5-2 til å beregne produktmatrisen i løpet av en omgang. Systemet vil inneholde en universell SAP enhet pr element i produktmatrisen.



Figur 5-3: Mulig utvidelse av system for multiplikasjon av 12x12 matriser med universelle SAP enheter

Å implementere multiplikasjon av 12x12 matriser på denne måten benytter alle de interne multiplikasjonsenhetene i FPGAen. For lagring av matrisene som skal multipliseres og produktmatrisen trengs det som for systemet i Figur 5-2 $2N + N^2$ RAMer. Med N=12 vil systemet trenge 24+144 RAMer hver med 144 lokasjoner. FPGAen inneholder 144 18Kbit RAMer, hver RAM har to porter

Utvidelse for å kunne multiplisere 144x144 matriser

Systemet for 12x12 matriser med universelle SAP enheter kan utvides til å multiplisere 144x144 matriser. Denne matrisestørrelsen er den største som kan behandles i FPGAen og samtidig ha begge matrisene som skal multipliseres i FPGAen. Produktmatrisen vil da måtte beregnes i løpet av 144 omganger tilsvarende som for systemene med universelle SAP enheter i avsnitt 3.2.1 og 3.3.1. Hver omgang vil inneholde 144 multiplikasjoner og 144 akkumulasjoner til sammen 288 operasjoner (steg).

Utvidelse for å kunne multiplisere NxN matriser med N>144

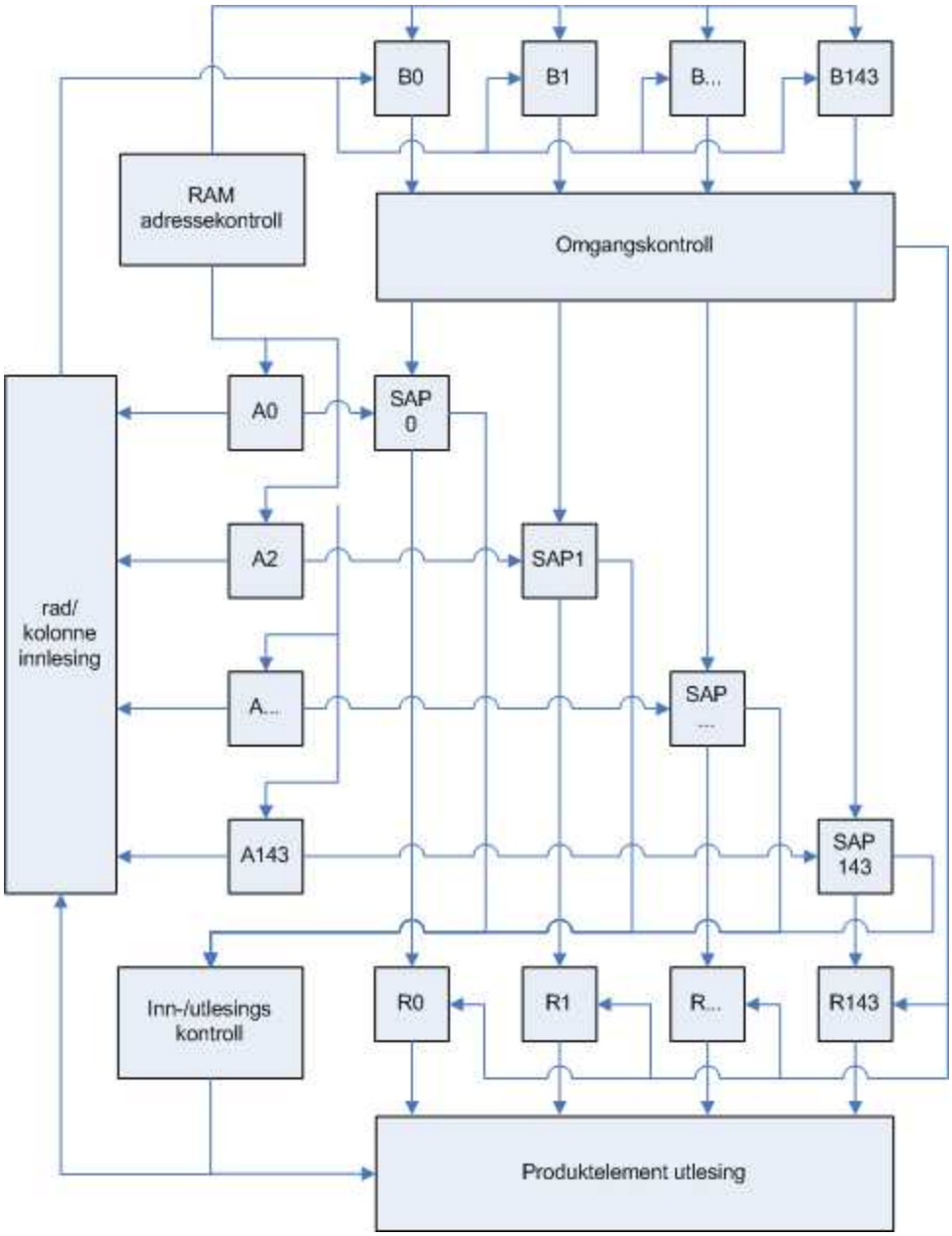
Matrisene som skal multipliseres deles opp i rader og kolonner som i systemene i kapittel 3. Produktmatrisen deles opp i undermatriser og beregnes en undermatrise av gangen. For å beregne en undermatrise lastes de radene og kolonnene som trengs for å beregne denne inn i FPGAen.

$$\begin{bmatrix} \mathbf{a_0} & \mathbf{a_1} & \mathbf{a_2} & \mathbf{a_3} \\ \mathbf{a_4} & \mathbf{a_5} & \mathbf{a_6} & \mathbf{a_7} \\ a_8 & a_9 & a_{10} & a_{11} \\ a_{12} & a_{13} & a_{14} & a_{15} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{b_0} & \mathbf{b_1} & b_2 & b_3 \\ \mathbf{b_4} & \mathbf{b_5} & b_6 & b_7 \\ \mathbf{b_8} & \mathbf{b_9} & b_{10} & b_{11} \\ \mathbf{b_{12}} & \mathbf{b_{13}} & b_{14} & b_{15} \end{bmatrix} = \begin{bmatrix} \mathbf{r_0} & \mathbf{r_1} & r_2 & r_3 \\ \mathbf{r_4} & \mathbf{r_5} & r_6 & r_7 \\ r_8 & r_9 & r_{10} & r_{11} \\ r_{12} & r_{13} & r_{14} & r_{15} \end{bmatrix}$$

Figur 5-4: Inndeling av produktmatrise i undermatriser

Metoden er skissert i Figur 5-4. For å beregne elementene i undermatrisen som består av elementene r_0, r_1, r_4, r_5 trengs radene a_0, a_1, a_2, a_3 og a_4, a_5, a_6, a_7 og kolonnene b_0, b_4, b_8, b_{12} og b_1, b_5, b_8, b_{13}. Matriser med andre størrelser kan deles opp på tilsvarende måte og elementene i undermatrisene beregnes av de tilhørende rader/kolonner.

For å kunne lage et slikt system trengs mekanismer for innlesning av rader/kolonner til RAM og utlesning av beregnede produktelementer i tillegg til den delen som forestår selve beregningen av elementene i produktmatrisen. I Figur 5-5 er et slikt system skissert. Systemet er basert på universelle SAP enheter, da bruk av tilpasset SAP vil medføre at rader/kolonner må deles opp for å beregne produktelementene, dette fordi tilpassede SAP enheter inneholder en multiplikasjonsblokk pr element i rad/kolonne. Inn- og utlesing av rader/kolonner og elementene i produktmatrisene gjøres med rad/kolonne innlesing og produktelement utlesing. Kontroll av inn-/utlesing gjøres med inn-/utlesingskontroll.



Figur 5-5:Mulig system for multiplikasjon av NxN matriser med N>144

6 OPPSUMMERING

Matrisemultiplikasjon er kjernen i algoritmer innen datagrafikk, signal- og bildebehandling. Matrisemultiplikasjon er en krevende operasjon siden beregning av et element inkluderer flere multiplikasjoner og addisjoner (antall avhenger av matrisestørrelsen som multipliseres). Å bruke FPGA til matrisemultiplikasjon er et alternativ som kan gi økt ytelse i forhold til å bruke mikroprosessor.

I arbeidet med denne rapporten er det er laget systemer i Händel C for multiplikasjon av NxN matriser i FPGA. Systemene er laget for en Xilinx Virtex-II XC2V6000 FPGA og for to matrisestørrelser, 4x4 og 12x12. Kjernen i systemene er sum av produkter (SAP) operasjoner. Operasjonene er realisert på to måter; universell og tilpasset SAP. Universell SAP kan brukes uansett matrisestørrelse og bruker lite ressurser, men bruker 2N steg på å beregne ett element. Tilpasset SAP må lages spesielt for hver matrisestørrelse og bruker mer ressurser enn universal SAP. Antall steg er forskjellig for hver utgave av tilpasset SAP, men felles er at alle multiplikasjonene gjøres parallelt i ett steg.

Funksjon og om systemene får plass i FPGAen er testet, men det ble ikke tid til å implementere en effektiv I/O-løsning så tidsforbruket er ikke målt, men estimert. Systemene som laget krever så lite ressurser at kompilering for en mindre FPGA kan være mulig.

Det estimerte tidsforbruket indikerer at tilpasset SAP er den raskeste løsningen for begge matrisestørrelsene.

Systemene bruker lite av ressursene i FPGAen, forbruket av LUT, vipper og blokk RAM er i størrelsesorden < 1 %. Multiplikasjonsblokkene i FPGAen er utnyttet fra underkant av 10 % for det minste systemet (multiplikasjon av 4x4 matriser med universal SAP) til 100 % for det største systemet (multiplikasjon av 12x12 matriser med tilpasset SAP). Systemene er sammenliknet med tidligere systemer for matrisemultiplikasjon (fra kapittel 2.5). Sammenliknet med tidligere system laget for samme FPGA (Xilinx Virtex-II XC2V6000) kreves mindre ressurser for systemene i denne rapporten. Om den estimerte ytelsen for systemene i denne rapporten sammenliknes med ytelsen til systemer for multiplikasjon av matriser med samme størrelse oppnår systemene i rapporten høyere ytelse.

Videre arbeid inkluderer implementasjon av en effektiv I/O løsning, konvertering av systemet fra Händel C til VHDL og å bytte ut RAM med registre i systemene med tilpasset SAP for å øke ytelsen.

Mulige utvidelser for systemene inkluderer å bygge systemene om slik at hele produktmatrisen kan beregnes i en omgang og å utvide systemene til å behandle matriser med størrelse opptil 144 internt i FPGA eller med størrelse større enn 144 ved å dele opp

matrisene som skal multipliseres og laste inn delene etter hvert som de trengs for å beregne produktmatrisen.

FIGURLISTE

TABELLISTE

REFERANSELISTE

1. Lay, D.C., *Linear Algebra and its applications*. 1997.
2. Skahill, K., *VHDL for programmable Logic*. 1996.
3. Xilinx, *Xilinx Virtex-II Datasheet*. 2003: Xilinx Corporation.
4. Celoxica, *RC2000 Datasheet*. 2005.
5. Celoxica, *Handel-C Language Reference Manual*. 2004.
6. Peter, C., *Overview: Hardware Compilation and the Händel-C language*. 2000.
7. Wikipedia, *Discrete cosine transform*. 2005.
8. Wikipedia, *Discrete Fourier transform*. 2005.
9. Kung, S.Y., *VLSI Array Processors.* IEEE ASSP Magazine, 1985.
10. Baugh, C.R., Wooley, B.A, *A two's complement parallel array multiplication algorithm.* IEEE-Transactions-on-Computers, 1973. **C-22(12)**(December 1973): p. 1045-1047.
11. Amira, A., Bensaali, F, *An FPGA based parameterisable system for matrix product implementation.* EEE Workshop on Signal Processing Systems, 2002. (SIPS '02). I, 2002: p. 625-628.
12. Amira, A., Bouridane, A.,Miligan, P., Sage, P., *A high throughput FPGA implementation of a bit-level matrix-matrix product.* Proceedings of the 43rd IEEE Midwest Symposium on Circuits and systems, 2000.
13. Wiatr, K., Jamro, Ernest, *Implementation of multipliers in FPGA structures.* 2001.
14. Jianwen, L., Chuen, Jong Ching, *Partially Reconfigurable Matrix multiplication for area and time efficiency on FPGAs.* Proceedings of the 2004 Euromicro Symposium on Digital System Design, 2004.
15. Zhuo, L., Prasanna, Viktor K., *Scalable and modular algorithms for floating-point matrix multiplication on FPGAs.* Proceedings of the 18th International Parallel and Distributed Processing Symposium, 2004.
16. Xilinx, *DSP Co-processing in FPGAs: Embedding high-performance, low-cost DSP functions*, S. Zack, Dhanani, Suhel, Editor. 2004.
17. McGhee, R.B., Bachmann, E. R & Zyda M. J, *Rigid Body Dynamics, Inertial Reference Frames, and Graphics Coordinate Systems: A Resolution of Conflicting Conventions and Terminology.* Journal of Computers and Graphics, 2000.
18. Pique, M.E., *Semantics of Interactive Rotations.* Proceedings of the 1986 workshop on Interactive 3D graphics, 1986.
19. Karczmarek, M., Thies, William & Amarasinghe, Saman, *Phased Scheduling of Stream Programs.* Proceedings of the 2003 ACM SIGPLAN conference on Language, compiler, and tool for embedded systems., 2003.
20. Xilinx, *LogiCORE PCI64 Interface v3.0*. 2005.
21. Wikipedia, *Low Voltage Differential Signaling*. 2005.
22. Xilinx, *High-Speed Data Serialization and Deserialization (840 Mb/s LVDS)*, N. Sawyer, Editor. 2002.
23. Xilinx, *Optimal Pipelining of I/O Ports of the Virtex-II Multiplier*, M. Adhiwiyogo, Editor. 2004.

VEDLEGSLISTE

A: Organisering av matriser, tilknyttede faste og vekslende RAMer for 12x12 systemene i kapittel 3.3.1 og 3.3.2.

B: Kildekode for systemene i kapittel 3.

A ORGANISERING AV MATRISENE OG SAP ENHETENE I SYSTEMENE FOR MULTIPLIKASJON AV 12X12 MATRISER

Matrise A

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} & a_{17} & a_{18} & a_{19} & a_{110} & a_{111} & a_{112} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & a_{26} & a_{27} & a_{28} & a_{29} & a_{210} & a_{211} & a_{212} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & a_{36} & a_{37} & a_{38} & a_{39} & a_{310} & a_{311} & a_{312} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} & a_{46} & a_{47} & a_{48} & a_{49} & a_{410} & a_{411} & a_{412} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} & a_{56} & a_{57} & a_{58} & a_{59} & a_{510} & a_{511} & a_{512} \\ a_{61} & a_{62} & a_{63} & a_{64} & a_{65} & a_{66} & a_{67} & a_{68} & a_{69} & a_{610} & a_{611} & a_{612} \\ a_{71} & a_{72} & a_{73} & a_{74} & a_{75} & a_{76} & a_{77} & a_{78} & a_{79} & a_{710} & a_{711} & a_{712} \\ a_{81} & a_{82} & a_{83} & a_{84} & a_{85} & a_{86} & a_{87} & a_{88} & a_{89} & a_{810} & a_{811} & a_{812} \\ a_{91} & a_{92} & a_{93} & a_{94} & a_{95} & a_{96} & a_{97} & a_{98} & a_{99} & a_{910} & a_{911} & a_{911} \\ a_{101} & a_{102} & a_{103} & a_{104} & a_{105} & a_{106} & a_{107} & a_{108} & a_{109} & a_{1010} & a_{1011} & a_{1012} \\ a_{111} & a_{112} & a_{113} & a_{114} & a_{115} & a_{116} & a_{117} & a_{118} & a_{119} & a_{1110} & a_{1111} & a_{1112} \\ a_{121} & a_{122} & a_{123} & a_{124} & a_{125} & a_{126} & a_{127} & a_{128} & a_{129} & a_{1210} & a_{1211} & a_{1212} \end{bmatrix}$$

Figur A-1: 12x12 matrisen A med konvensjonelle indekser

$$A = \begin{bmatrix} a0 & a1 & a2 & a3 & a4 & a5 & a6 & a7 & a8 & a9 & a10 & a11 \\ a12 & a13 & a14 & a15 & a16 & a17 & a18 & a19 & a20 & a21 & a22 & a23 \\ a24 & a25 & a26 & a27 & a28 & a29 & a30 & a31 & a32 & a33 & a34 & a35 \\ a36 & a37 & a38 & a39 & a40 & a41 & a42 & a43 & a44 & a45 & a46 & a47 \\ a48 & a49 & a50 & a51 & a52 & a53 & a54 & a55 & a56 & a57 & a58 & a59 \\ a60 & a61 & a62 & a63 & a64 & a65 & a66 & a67 & a68 & a69 & a70 & a71 \\ a72 & a73 & a74 & a75 & a76 & a77 & a78 & a79 & a80 & a81 & a82 & a83 \\ a84 & a85 & a86 & a87 & a88 & a89 & a90 & a91 & a92 & a93 & a94 & a95 \\ a96 & a97 & a98 & a99 & a100 & a101 & a102 & a103 & a104 & a105 & a106 & a107 \\ a108 & a109 & a110 & a111 & a112 & a113 & a114 & a115 & a116 & a117 & a118 & a119 \\ a120 & a121 & a122 & a123 & a124 & a125 & a126 & a127 & a128 & a129 & a130 & a131 \\ a132 & a133 & a134 & a135 & a136 & a137 & a138 & a139 & a140 & a141 & a142 & a143 \end{bmatrix}$$

Figur A-2: 12x12 matrise A med indekser brukt i systemene

	RAM											
Adresse	**A0**	**A1**	**A2**	**A3**	**A4**	**A5**	**A6**	**A7**	**A8**	**A9**	**A10**	**A11**
0	a0	a12	a24	a36	a48	a60	a72	a84	a96	a108	a120	a132
1	a1	a13	a25	a37	a49	a61	a73	a85	a97	a109	a121	a133
2	a2	a14	a26	a38	a50	a62	a74	a86	a98	a110	a122	a134
3	a3	a15	a27	a39	a51	a63	a75	a87	a99	a111	a123	a135
4	a4	a16	a28	a40	a52	a64	a76	a88	a100	a112	a124	a136
5	a5	a17	a29	a41	a53	a65	a77	a89	a101	a113	a125	a137
6	a6	a18	a30	a42	a54	a66	a78	a90	a102	a114	a126	a138
7	a7	a19	a31	a43	a55	a67	a79	a91	a103	a115	a127	a139
8	a8	a20	a32	a44	a56	a68	a80	a92	a104	a116	a128	a140
9	a9	a21	a33	a45	a57	a69	a81	a93	a105	a117	a129	a141
10	a10	a22	a34	a46	a58	a70	a82	a94	a106	a118	a130	a142
11	a11	a23	a35	a47	a59	a71	a83	a95	a107	a119	a131	a143

Tabell A-1: Radene i 12x12 matrise A i RAMene A0-A11

Matrise B

$$B = \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} & b_{15} & b_{16} & b_{17} & b_{18} & b_{19} & b_{110} & b_{111} & b_{112} \\ b_{21} & b_{22} & b_{23} & b_{24} & b_{25} & b_{26} & b_{27} & b_{28} & b_{29} & b_{210} & b_{211} & b_{212} \\ b_{31} & b_{32} & b_{33} & b_{34} & b_{35} & b_{36} & b_{37} & b_{38} & b_{39} & b_{310} & b_{311} & b_{312} \\ b_{41} & b_{42} & b_{43} & b_{44} & b_{45} & b_{46} & b_{47} & b_{48} & b_{49} & b_{410} & b_{411} & b_{412} \\ b_{51} & b_{52} & b_{53} & b_{54} & b_{55} & b_{56} & b_{57} & b_{58} & b_{59} & b_{510} & b_{511} & b_{512} \\ b_{61} & b_{62} & b_{63} & b_{64} & b_{65} & b_{66} & b_{67} & b_{68} & b_{69} & b_{610} & b_{611} & b_{612} \\ b_{71} & b_{72} & b_{73} & b_{74} & b_{75} & b_{76} & b_{77} & b_{78} & b_{79} & b_{710} & b_{711} & b_{712} \\ b_{81} & b_{82} & b_{83} & b_{84} & b_{85} & b_{86} & b_{87} & b_{88} & b_{89} & b_{810} & b_{811} & b_{812} \\ b_{91} & b_{92} & b_{93} & b_{94} & b_{95} & b_{96} & b_{97} & b_{98} & b_{99} & b_{910} & b_{911} & b_{911} \\ b_{101} & b_{102} & b_{103} & b_{104} & b_{105} & b_{106} & b_{107} & b_{108} & b_{109} & b_{1010} & b_{1011} & b_{1012} \\ b_{111} & b_{112} & b_{113} & b_{114} & b_{115} & b_{116} & b_{117} & b_{118} & b_{119} & b_{1110} & b_{1111} & b_{1112} \\ b_{121} & b_{122} & b_{123} & b_{124} & b_{125} & b_{126} & b_{127} & b_{128} & b_{129} & b_{1210} & b_{1211} & b_{1212} \end{bmatrix}$$

Figur A-3: 12x12 matrise B konvensjonelle indekser

$$B = \begin{bmatrix} b0 & b1 & b2 & b3 & b4 & a5 & b6 & b7 & b8 & b9 & b10 & b11 \\ b12 & b13 & b14 & b15 & b16 & b17 & b18 & b19 & b20 & b21 & b22 & b23 \\ b24 & b25 & b26 & b27 & b28 & b29 & b30 & b31 & b32 & b33 & b34 & b35 \\ b36 & b37 & b38 & b39 & b40 & b41 & b42 & b43 & b44 & b45 & b46 & b47 \\ b48 & b49 & b50 & b51 & b52 & b53 & b54 & b55 & b56 & b57 & b58 & b59 \\ b60 & b61 & b62 & b63 & b64 & b65 & b66 & b67 & b68 & b69 & b70 & b71 \\ b72 & b73 & b74 & b75 & b76 & b77 & b78 & b79 & b80 & b81 & b82 & b83 \\ b84 & b85 & b86 & b87 & b88 & b89 & b90 & b91 & b92 & b93 & b94 & b95 \\ b96 & b97 & b98 & b99 & b100 & b101 & b102 & b103 & b104 & b105 & b106 & b107 \\ b108 & b109 & b110 & b111 & b112 & b113 & b114 & b115 & b116 & b117 & b118 & b119 \\ b120 & b121 & b122 & b123 & b124 & b125 & b126 & b127 & b128 & b129 & b130 & b131 \\ b132 & b133 & b134 & b135 & b136 & b137 & b138 & b139 & b140 & b141 & b142 & b143 \end{bmatrix}$$

Figur A-4: 12x12 matrise B med indekser brukt i systemene

	RAM											
adresse	**B0**	**B1**	**B2**	**B3**	**B4**	**B5**	**B6**	**B7**	**B8**	**B9**	**B10**	**B11**
0	b0	b1	b2	b3	b4	b5	b6	b7	b8	b9	b10	b11
1	b12	b13	b14	b15	b16	b17	b18	b19	b20	b21	b22	b23
2	b24	b25	b26	b27	b28	b29	b30	b31	b32	b33	b34	b35
3	b36	b37	b38	b39	b40	b41	b42	b43	b44	b45	b46	b47
4	b48	b49	b50	b51	b52	b53	b54	b55	b56	b57	b58	b59
5	b60	b61	b62	b63	b64	b65	b66	b67	b68	b69	b70	b71
6	b72	b73	b74	b75	b76	b77	b78	b79	b80	b81	b82	b83
7	b84	b85	b86	b87	b88	b89	b90	b91	b92	b93	b94	b95
8	b96	b97	b98	b99	b100	b101	b102	b103	b104	b105	b106	b107
9	b108	b109	b110	b111	b112	b113	b114	b115	b116	b117	b118	b119
10	b120	b121	b122	b123	b124	b125	b126	b127	b128	b129	b130	b131
11	b132	b133	b134	b135	b136	b137	b138	b139	b140	b141	b142	b143

Tabell A-2: Kolonnene i 12x12 matrise B i RAMene B0-B11

Produktmatrisen R

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} & r_{15} & {}_{16} & r_{17} & r_{18} & r_{19} & r_{110} & r_{111} & r_{112} \\ r_{21} & r_{22} & r_{23} & r_{24} & r_{25} & r_{26} & r_{27} & r_{28} & r_{29} & r_{210} & r_{211} & r_{212} \\ r_{31} & r_{32} & r_{33} & r_{34} & r_{35} & r_{36} & r_{37} & r_{38} & r_{39} & r_{310} & r_{311} & r_{312} \\ r_{41} & r_{42} & r_{43} & r_{44} & r_{45} & r_{46} & r_{47} & r_{48} & r_{49} & r_{410} & r_{411} & r_{412} \\ r_{51} & r_{52} & r_{53} & r_{54} & r_{55} & r_{56} & r_{57} & r_{58} & r_{59} & r_{510} & r_{511} & r_{512} \\ r_{61} & r_{62} & r_{63} & r_{64} & r_{65} & r_{66} & r_{67} & r_{68} & r_{69} & r_{610} & r_{611} & r_{612} \\ r_{71} & r_{72} & r_{73} & r_{74} & r_{75} & r_{76} & r_{77} & r_{78} & r_{79} & r_{710} & r_{711} & r_{712} \\ r_{81} & r_{82} & r_{83} & r_{84} & r_{85} & r_{86} & r_{87} & r_{88} & r_{89} & r_{810} & r_{811} & r_{812} \\ r_{91} & r_{92} & r_{93} & r_{94} & r_{95} & r_{96} & r_{97} & r_{98} & r_{99} & r_{910} & r_{911} & r_{911} \\ r_{101} & r_{102} & r_{103} & r_{104} & r_{105} & r_{106} & r_{107} & r_{108} & r_{109} & r_{1010} & r_{1011} & r_{1012} \\ r_{111} & r_{112} & r_{113} & r_{114} & r_{115} & r_{116} & r_{117} & r_{118} & r_{119} & r_{1110} & r_{1111} & r_{1112} \\ r_{121} & r_{122} & r_{123} & r_{124} & r_{125} & r_{126} & r_{127} & r_{128} & r_{129} & r_{1210} & r_{1211} & r_{1212} \end{bmatrix}$$

Figur A-5: 12x12 produktmatrisen R konvensjonelle indekser

$$R = \begin{bmatrix} r0 & r1 & r2 & r3 & r4 & r5 & r6 & r7 & r8 & r9 & r10 & r11 \\ r12 & r13 & r14 & r15 & r16 & r17 & r18 & r19 & r20 & r21 & r22 & r23 \\ r24 & r25 & r26 & r27 & r28 & r29 & r30 & r31 & r32 & r33 & r34 & r35 \\ r36 & r37 & r38 & r39 & r40 & r41 & r42 & r43 & r44 & r45 & r46 & r47 \\ r48 & r49 & r50 & r51 & r52 & r53 & r54 & r55 & r56 & r57 & r58 & r59 \\ r60 & r61 & r62 & r63 & r64 & r65 & r66 & r67 & r68 & r69 & r70 & r71 \\ r72 & r73 & r74 & r75 & r76 & r77 & r78 & r79 & r80 & r81 & r82 & r83 \\ r84 & r85 & r86 & r87 & r88 & r89 & r90 & r91 & r92 & r93 & r94 & r95 \\ r96 & r97 & r98 & r99 & r100 & r101 & r102 & r103 & r104 & r105 & r106 & r107 \\ r108 & r109 & r110 & r111 & r112 & r113 & r114 & r115 & r116 & r117 & r118 & r119 \\ r120 & r121 & r122 & r123 & r124 & r125 & r126 & r127 & r128 & r129 & r130 & r131 \\ r132 & r133 & r134 & r135 & r136 & r137 & r138 & r139 & r140 & r141 & r142 & r143 \end{bmatrix}$$

Figur A-6: 12x12 produktmatrisen R med indekser brukt i systemene

	RAM											
Adresse	**R0**	**R1**	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**	**R10**	**R11**
0	r0	r12	r24	r36	r48	r60	r72	r84	r96	r108	r120	r132
1	r1	r13	r25	r37	r49	r61	r73	r85	r97	r109	r121	r133
2	r2	r14	r26	r38	r50	r62	r74	r86	r98	r110	r122	r134
3	r3	r15	r27	r39	r51	r63	r75	r87	r99	r111	r123	r135
4	r4	r16	r28	r40	r52	r64	r76	r88	r100	r112	r124	r136
5	r5	r17	r29	r41	r53	r65	r77	r89	r101	r113	r125	r137
6	r6	r18	r30	r42	r54	r66	r78	r90	r102	r114	r126	r138
7	r7	r19	r31	r43	r55	r67	r79	r91	r103	r115	r127	r139
8	r8	r20	r32	r44	r56	r68	r80	r92	r104	r116	r128	r140
9	r9	r21	r33	r45	r57	r69	r81	r93	r105	r117	r129	r141
10	r10	r22	r34	r46	r58	r70	r82	r94	r106	r118	r130	r142
11	r11	r23	r35	r47	r59	r71	r83	r95	r107	r119	r131	r143

Tabell A-3: kolonnene i12x12 produktmatrisen R i RAMene R0-R11

		SAP enhet											
j	**omgang**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
0000	**1**	R0[0]	R1[0]	R2[0]	R3[0]	R4[0]	R5[0]	R6[0]	R7[0]	R8[0]	R9[0]	R10[0]	R11[0]
0001	**2**	R0[1]	R1[1]	R2[1]	R3[1]	R4[1]	R5[1]	R6[1]	R7[1]	R8[1]	R9[1]	R10[1]	R11[1]
0010	**3**	R0[2]	R1[2]	R2[2]	R3[2]	R4[2]	R5[2]	R6[2]	R7[2]	R8[2]	R9[2]	R10[2]	R11[2]
0011	**4**	R0[3]	R1[3]	R2[3]	R3[3]	R4[3]	R5[3]	R6[3]	R7[3]	R8[3]	R9[3]	R10[3]	R11[3]
0100	**5**	R0[4]	R1[4]	R2[4]	R3[4]	R4[4]	R5[4]	R6[4]	R7[4]	R8[4]	R9[4]	R10[4]	R11[4]
0101	**6**	R0[5]	R1[5]	R2[5]	R3[5]	R4[5]	R5[5]	R6[5]	R7[5]	R8[5]	R9[5]	R10[5]	R11[5]
0110	**7**	R0[6]	R1[6]	R2[6]	R3[6]	R4[6]	R5[6]	R6[6]	R7[6]	R8[6]	R9[6]	R10[6]	R11[6]
0111	**8**	R0[7]	R1[7]	R2[7]	R3[7]	R4[7]	R5[7]	R6[7]	R7[7]	R8[7]	R9[7]	R10[7]	R11[7]
1000	**9**	R0[8]	R1[8]	R2[8]	R3[8]	R4[8]	R5[8]	R6[8]	R7[8]	R8[8]	R9[8]	R10[8]	R11[8]
1001	**10**	R0[9]	R1[9]	R2[9]	R3[9]	R4[9]	R5[9]	R6[9]	R7[9]	R8[9]	R9[9]	R10[9]	R11[9]
1010	**11**	R0[10]	R1[10]	R2[10]	R3[10]	R4[10]	R5[10]	R6[10]	R7[10]	R8[10]	R9[10]	R10[10]	R11[10]
1011	**12**	R0[11]	R1[11]	R2[11]	R3[11]	R4[11]	R5[11]	R6[11]	R7[11]	R8[11]	R9[11]	R10[11]	R11[11]

Tabell A-4: Adresser for lagring av beregnede elementer i de forskjellige SAP enhetene for omgang 1-12

	SAP enhet											
Omgang	**SAP0**	**SAP1**	**SAP2**	**SAP3**	**SAP4**	**SAP5**	**SAP6**	**SAP7**	**SAP8**	**SAP9**	**SAP10**	**SAP11**
0	r0	r12	r24	r36	r48	r60	r72	r84	r96	r108	r120	r132
1	r1	r13	r25	r37	r49	r61	r73	r85	r97	r109	r121	r133
2	r2	r14	r26	r38	r50	r62	r74	r86	r98	r110	r122	r134
3	r3	r15	r27	r39	r51	r63	r75	r87	r99	r111	r123	r135
4	r4	r16	r28	r40	r52	r64	r76	r88	r100	r112	r124	r136
5	r5	r17	r29	r41	r53	r65	r77	r89	r101	r113	r125	r137
6	r6	r18	r30	r42	r54	r66	r78	r90	r102	r114	r126	r138
7	r7	r19	r31	r43	r55	r67	r79	r91	r103	r115	r127	r139
8	r8	r20	r32	r44	r56	r68	r80	r92	r104	r116	r128	r140
9	r9	r21	r33	r45	r57	r69	r81	r93	r105	r117	r129	r141
10	r10	r22	r34	r46	r58	r70	r82	r94	r106	r118	r130	r142
11	r11	r23	r35	r47	r59	r71	r83	r95	r107	r119	r131	r143

Tabell A-5:Hvilke elementer i R som beregnes i hvilken omgang.

		SAP enhet											
j	**omgang**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
0000	1	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11
0001	2	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B0
0010	3	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B0	B1
0011	4	B3	B4	B5	B6	B7	B8	B9	B10	B11	B0	B1	B2
0100	5	B4	B5	B6	B7	B8	B9	B10	B11	B0	B1	B2	B3
0101	6	B5	B6	B7	B8	B9	B10	B11	B0	B1	B2	B3	B4
0110	7	B6	B7	B8	B9	B10	B11	B0	B1	B2	B3	B4	B5
0111	8	B7	B8	B9	B10	B11	B0	B1	B2	B3	B4	B5	B6
1000	9	B8	B9	B10	B11	B0	B1	B2	B3	B4	B5	B6	B7
1001	10	B9	B10	B11	B0	B1	B2	B3	B4	B5	B6	B7	B8
1010	11	B10	B11	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9
1011	12	B11	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10

Tabell A-6: Tilordning av B RAM til SAP enhetene for omgang 1-12

	RAM	
SAP	**A RAM**	**R RAM**
0	A0	R0
1	A1	R1
2	A2	R2
3	A3	R3
4	A4	R4
5	A5	R5
6	A6	R6
7	A7	R7
8	A8	R8
9	A9	R9
10	A10	R10
11	A11	R11

Tabell A-7: Fast tilknyttede RAMer (A og R) for SAP enhet 0-12

B KILDEKODE FOR SYSTEMENE I KAPITTEL 3.

B.1 System for multiplikasjon av 4x4 matriser med universelle SAP enheter

```
1   ////////////////////////////////////////////////////////////
2   //   Händel C program for multiplikasjon av 4x4 matriser    //
3   //   med universelle SAP enheter                             //
4   //                                                          //
5   //   G H Andersen 2005.                                     //
6   ////////////////////////////////////////////////////////////
7   //Definisjoner og header filer for 18x18bit Multiplier
8   //og ADMXRC2 FPGAkort.
9   #define ADMXRC2_CLOCK_RATE 100000000
10  #define ADMXRC2_USE_LCLK
11  #include "admxrc2.hch"
12  #include "xilinxmul.hch"
13
14  void main (void)
15  {
16      //deklarasjon av innganger og hjelpevariabler.
17      signed int 18 a0,a1,a2,a3,b0,b1,b2,b3;
18      signed int 36 q0,q1,q2,q3;
19      signed int 36 p0,p1,p2,p3;
20      // RAM
21      //RAMer som inneholder radene i matrise A.
22      ram signed 18 A0[4] with { block = "BlockRAM" };
23      ram signed 18 A1[4] with { block = "BlockRAM" };
24      ram signed 18 A2[4] with { block = "BlockRAM" };
25      ram signed 18 A3[4] with { block = "BlockRAM" };
26      //RAMer som innholder kolonnene i matrise B.
27      ram signed 18 B0[4] with { block = "BlockRAM" };
28      ram signed 18 B1[4] with { block = "BlockRAM" };
```

```
29        ram signed 18 B2[4] with { block = "BlockRAM" };
30        ram signed 18 B3[4] with { block = "BlockRAM" };
31        //RAMer hvor radene i produktmatrisen lagres.
32        ram signed 36 R0[4]with { block = "BlockRAM" };
33        ram signed 36 R1[4]with { block = "BlockRAM" };
34        ram signed 36 R2[4]with { block = "BlockRAM" };
35        ram signed 36 R3[4]with { block = "BlockRAM" };
36        //Deklarasjon og initialisering av kontrollvariabler
37        unsigned int 2 i; //Variabelen i kontrollerer hvilke
38                          //adresser i RAM som leses
39        unsigned int 2 j; //Variabelen j kontrollerer hvilken
40                          // omgang vi er i
41        unsigned int 1 ferdig;
42        unsigned int 1 start;
43        start =0;
44        ferdig =0;
45
46       //oppstart av systemet
47       //Initialisering av variabler og porter
48       if (start ==1)
49       {
50        par
51        {
52        i =0;
53        j =0;
54        a0=0;
55        a1=0;
56        a2=0;
57        a3=0;
58        b0=0;
59        b1=0;
60        b2=0;
61        b3=0;
```

```
62          q0=0;
63          q1=0;
64          q2=0;
65          q3=0;
66          p0=0;
67          p1=0;
68          p2=0;
69          p3=0;
70          }
71          }
72           //En rad fra matrise A legges fast til hver SAP.
73          par
74          {
75          a0 = A0[i];
76          a1 = A1[i];
77          a2 = A2[i];
78          a3 = A3[i];
79          }
80          //kolonnene fra matrise B
81          //skiftes mellom SAPenhetene.
82          par{
83              {
84              if (j == 0)//tilordning for 1. omgang
85              {
86                  par
87                  {
88                  b0 = B0[i];
89                  b1 = B1[i];
90                  b2 = B2[i];
91                  b3 = B3[i];
92                  }
93              }
94              if (j == 1)//tilordning for 2. omgang
```

```
95              {
96                  par
97                  {
98                  b0 = B1[i];
99                  b1 = B2[i];
100                 b2 = B3[i];
101                 b3 = B0[i];
102                 }
103             }
104             if (j == 2)//tilordning for 3. omgang
105             {
106                 par
107                 {
108                 b0 = B2[i];
109                 b1 = B3[i];
110                 b2 = B0[i];
111                 b3 = B1[i];
112                 }
113             }
114             if (j == 3)//tilordning for 4. omgang
115             {
116                 par
117                 {
118                 b0 = B3[i];
119                 b1 = B0[i];
120                 b2 = B1[i];
121                 b3 = B2[i];
122                 }
123             }
124            }
125         do
126         {
127             par
```

```
128             {
129             xilinxmult0(q0, a0, b0);
130             xilinxmult1(q1, a1, b1);
131             xilinxmult2(q2, a2, b2);
132             xilinxmult3(q3, a3, b3);
133             }
134             par
135             {
136             p0 = p0+q0;
137             p1 = p1+q1;
138             p2 = p2+q2;
139             p3 = p3+q3;
140             i=i+1;
141             }
142         } while (i<=3);
143         //Når i=3 er elementene beregnet
144         //og lagres
145         if (i==3)
146         {
147             par
148             {
149             R0[j] = p0;
150             R1[j] = p1;
151             R2[j] = p2;
152             R3[j] = p3;
153             }
154             // når j=3 er R beregnet
155             if (j==3 )
156             {
157             ferdig = 1;
158             }
159             par
160             {
```

```
161             i=0;
162             j=j+1;
163             }
164             //Reset av hjelpevariabler
165             par
166             {
167             q0=0;
168             q1=0;
169             q2=0;
170             q3=0;
171             p0=0;
172             p1=0;
173             p2=0;
174             p3=0;
175             }
176         }
177     }
178     }
```

B.2 System for multiplikasjon av 4x4 matriser med universelle SAP enheter

```
1   ////////////////////////////////////////////////////////////
2   //   Händel C program for multiplikasjon av 4x4 matriser    //
3   //   med tilpassede SAP enheter                             //
4   //                                                          //
5   //   G H Andersen 2005                                      //
6   ////////////////////////////////////////////////////////////
7
8   //Definisjoner og header filer for 18bit Multiplier
9   //og ADMXRC2 FPGAkort.
10  #define ADMXRC2_CLOCK_RATE 100000000
11  #define ADMXRC2_USE_LCLK
12  #include "admxrc2.hch"
13  #include "xilinxmul.hch"
14
15  void main(void)
16  {
17      ///deklarasjon av innganger og hjelpevariabler.
18      signed int 18 a0,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,
19                    a11,a12,a13,a14,a15,b0,b1,b2,b3,b4,
20                    b5,b6,b7,b8,b9,b10,b11,b12,b13,b14,b15;
21      signed int 36 q0,q1,q2,q3,q4,q5,q6,q7,q8,q9,q10,q11,
22                    q12,q13,q14,q15;
23      signed int 36 t0,t1,t2,t3,t4,t5,t6,t7;
24      //RAM
25      //Radene i Matrise A
26      ram signed 18 A0[4] with { block = "BlockRAM" };
27      ram signed 18 A1[4] with { block = "BlockRAM" };
28      ram signed 18 A2[4] with { block = "BlockRAM" };
```

```
29        ram signed 18 A3[4] with { block = "BlockRAM" };
30        //Kolonnene i matrise B
31        ram signed 18 B0[4] with { block = "BlockRAM" };
32        ram signed 18 B1[4] with { block = "BlockRAM" };
33        ram signed 18 B2[4] with { block = "BlockRAM" };
34        ram signed 18 B3[4] with { block = "BlockRAM" };
35        //kolonnene i prdouktmatrisen R
36        ram signed 36 R0[4] with { block = "BlockRAM" };
37        ram signed 36 R1[4] with { block = "BlockRAM" };
38        ram signed 36 R2[4] with { block = "BlockRAM" };
39        ram signed 36 R3[4] with { block = "BlockRAM" };
40        //deklarasjon av kontrollvariabel
41        unsigned int 2 j;
42        unsigned int 1 start;
43        unsigned int 1 ferdig;
44        //Oppstart av systemet
45        //Initialisering av variabler og porter
46        start =0;
47        ferdig =0;
48        if (start ==1)
49        {
50        par
51        {
52        j=0;
53        a0=0;
54        a1=0;
55        a2=0;
56        a3=0;
57        a4=0;
58        a5=0;
59        a6=0;
60        a7=0;
61        a8=0;
```

```
62          a9=0;
63          a10=0;
64          a11=0;
65          a12=0;
66          a13=0;
67          a14=0;
68          a15=0;
69          b0=0;
70          b1=0;
71          b2=0;
72          b3=0;
73          b4=0;
74          b5=0;
75          b6=0;
76          b7=0;
77          b8=0;
78          b9=0;
79          b10=0;
80          b11=0;
81          b12=0;
82          b13=0;
83          b14=0;
84          b15=0;
85          q0=0;
86          q1=0;
87          q2=0;
88          q3=0;
89          q4=0;
90          q5=0;
91          q6=0;
92          q7=0;
93          q8=0;
94          q9=0;
```

```
95        q10=0;
96        q11=0;
97        q12=0;
98        q13=0;
99        q14=0;
100       q15=0;
101       t0=0;
102       t1=0;
103       t2=0;
104       t3=0;
105       t4=0;
106       t5=0;
107       t6=0;
108       t7=0;
109       }
110       }
111       //En rad fra matrise A legges
112       //fast til hver SAPenhet.
113       par
114       {
115       a0 =A0[0];
116       a4 =A1[0];
117       a8 =A2[0];
118       a12 =A3[0];
119       }
120       par
121       {
122       a1 =A0[1];
123       a5=A1[1];
124       a9=A2[1];
125       a13=A3[1];
126       }
127       par
```

```
128       {
129       a2=A0[2];
130       a6=A1[2];
131       a10=A2[2];
132       a14=A3[2];
133       }
134       par
135       {
136       a3=A0[3];
137       a7=A1[3];
138       a11=A2[3];
139       a15=A3[3];
140       }
141      //kolonnene fra matrise B skiftes mellom SAPenhetene.
142       par
143           {
144           {
145           if (j == 0)//data tilordning for 1. omgang
146           {
147           par
148           {
149           b0=B0[0];
150           b4=B1[0];
151           b8=B2[0];
152           b12=B3[0];
153           }
154           par
155           {
156           b1=B0[1];
157           b5=B1[1];
158           b9=B2[1];
159           b13=B3[1];
160           }
```

```
161             par
162             {
163             b2=B0[2];
164             b6=B1[2];
165             b10=B2[2];
166             b14=B3[2];
167             }
168             par
169             {
170             b3=B0[3];
171             b7=B1[3];
172             b11=B2[3];
173             b15=B3[3];
174             }
175         }
176         if (j==1)//data tilordning for 2. omgang
177         {
178             par
179             {
180             b0=B1[0];
181             b4=B2[0];
182             b8=B3[0];
183             b12=B0[0];
184             }
185             par
186             {
187             b1=B1[1];
188             b5=B2[1];
189             b9=B3[1];
190             b13=B0[1];
191             }
192             par
193             {
```

```
194             b2=B1[2];
195             b6=B2[2];
196             b10=B3[2];
197             b14=B0[2];
198             }
199             par
200             {
201             b3=B1[3];
202             b7=B2[3];
203             b11=B3[3];
204             b15=B0[3];
205             }
206         }
207         if (j==2)//data tilordning for 3. omgang
208         {
209             par
210             {
211             b0=B2[0];
212             b4=B3[0];
213             b8=B0[0];
214             b12=B1[0];
215             }
216             par
217             {
218             b1=B2[1];
219             b5=B3[1];
220             b9=B0[1];
221             b13=B1[1];
222             }
223             par
224             {
225             b2=B2[2];
226             b6=B3[2];
```

```
227             b10=B0[2];
228             b14=B1[2];
229             }
230             par
231             {
232             b3=B2[3];
233             b7=B3[3];
234             b11=B0[3];
235             b15=B1[3];
236             }
237         }
238         if (j==3)//data tilordning for 4. omgang
239         {
240             par
241             {
242             b0=B3[0];
243             b4=B0[0];
244             b8=B1[0];
245             b12=B2[0];
246             }
247             par
248             {
249             b1=B3[1];
250             b5=B0[1];
251             b9=B1[1];
252             b13=B2[1];
253             }
254             par
255             {
256             b2=B3[2];
257             b6=B0[2];
258             b10=B1[2];
259             b14=B2[2];
```

```
260             }
261             par
262             {
263             b3=B3[3];
264             b7=B0[3];
265             b11=B1[3];
266             b15=B2[3];
267             }
268         }
269         }
270         do
271         {
272             par//elementene multipliseres i parallell
273             {
274             xilinxmult0(q0, a0, b0);
275             xilinxmult1(q1, a1, b1);
276             xilinxmult2(q2, a2, b2);
277             xilinxmult3(q3, a3, b3);
278             xilinxmult4(q4, a4, b4);
279             xilinxmult5(q5, a5, b5);
280             xilinxmult6(q6, a6, b6);
281             xilinxmult7(q7, a7, b7);
282             xilinxmult8(q8, a8, b8);
283             xilinxmult9(q9, a9, b9);
284             xilinxmult10(q10, a10, b10);
285             xilinxmult11(q11, a11, b11);
286             xilinxmult12(q12, a12, b12);
287             xilinxmult13(q13, a13, b13);
288             xilinxmult14(q14, a14, b14);
289             xilinxmult15(q15, a15, b15);
290             }
291             //delproduktene adderes i to omganger
292             par
```

```
293             {
294             t0=q0+q1;
295             t1=q2+q3;
296             t2=q4+q5;
297             t3=q6+q7;
298             t4=q8+q9;
299             t5=q10+q11;
300             t6=q12+q13;
301             t7=q14+q15;
302             }
303             //til elementer i produktmatrisen R
304             //som lagres
305             par
306             {
307             R0[j]=t0+t1;
308             R1[j]=t2+t3;
309             R2[j]=t4+t5;
310             R3[j]=t6+t7;
311             }
312             //når j=3 er R beregnet
313             if (j==3)
314             {
315             ferdig = 1;
316             }
317             j=j+1;
318             //reset av hjelpvariabler
319             par
320             {
321             q0=0;
322             q1=0;
323             q2=0;
324             q3=0;
325             q4=0;
```

```
326             q5=0;
327             q6=0;
328             q7=0;
329             q8=0;
330             q9=0;
331             q10=0;
332             q11=0;
333             q12=0;
334             q13=0;
335             q14=0;
336             q15=0;
337             t0=0;
338             t1=0;
339             t2=0;
340             t3=0;
341             t4=0;
342             t5=0;
343             t6=0;
344             t7=0;
345             }
346         }while (j<=3);
347         }
348     }
```

B.3 System for multiplikasjon av 4x4 matriser med universelle SAP enheter

```
1   ////////////////////////////////////////////////////////
2   //   HändelC program for multiplikasjon av 12x12 matriser //
3   //    med universelle SAPenheter                        //
4   //                                                      //
5   //   G H Andersen 2005                                  //
6   ////////////////////////////////////////////////////////
7   //Definisjoner og header filer for 18bit Multiplier
8   // og ADMXRC2 FPGAkort.
9   #define ADMXRC2_CLOCK_RATE 100000000
10  #define ADMXRC2_USE_LCLK
11  #include "admxrc2.hch"
12  #include "xilinxmul.hch"
13  void main (void)
14  {
15      //deklarasjon av innganger og hjelpevariabler.
16      signed int 18 a0,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,
17                    b0,b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11;
18      signed int 36 q0,q1,q2,q3,q4,q5,q6,q7,q8,q9,q10,q11;
19      signed int 36 p0,p1,p2,p3,p4,p5,p6,p7,p8,p9,p10,p11;
20       // RAM
21      //Radene i matrise A lagres i disse RAMene.
22      ram signed 18  A0[12]  with { block = "BlockRAM" };
23      ram signed 18  A1[12]  with { block = "BlockRAM" };
24      ram signed 18  A2[12]  with { block = "BlockRAM" };
25      ram signed 18  A3[12]  with { block = "BlockRAM" };
26      ram signed 18  A4[12]  with { block = "BlockRAM" };
27      ram signed 18  A5[12]  with { block = "BlockRAM" };
28      ram signed 18  A6[12]  with { block = "BlockRAM" };
```

```
29        ram signed 18  A7[12]  with { block = "BlockRAM" };
30        ram signed 18  A8[12]  with { block = "BlockRAM" };
31        ram signed 18  A9[12]  with { block = "BlockRAM" };
32        ram signed 18 A10[12]  with { block = "BlockRAM" };
33        ram signed 18 A11[12]  with { block = "BlockRAM" };
34       //Kolonnene i matrise B lagres i disse RAMene.
35        ram signed 18  B0[12]  with { block = "BlockRAM" };
36        ram signed 18  B1[12]  with { block = "BlockRAM" };
37        ram signed 18  B2[12]  with { block = "BlockRAM" };
38        ram signed 18  B3[12]  with { block = "BlockRAM" };
39        ram signed 18  B4[12]  with { block = "BlockRAM" };
40        ram signed 18  B5[12]  with { block = "BlockRAM" };
41        ram signed 18  B6[12]  with { block = "BlockRAM" };
42        ram signed 18  B7[12]  with { block = "BlockRAM" };
43        ram signed 18  B8[12]  with { block = "BlockRAM" };
44        ram signed 18  B9[12]  with { block = "BlockRAM" };
45        ram signed 18 B10[12]  with { block = "BlockRAM" };
46        ram signed 18 B11[12]  with { block = "BlockRAM" };
47       //Kolonnene i resultatmatrisen R lagres
48       //i disse RAMene
49        ram signed 36  R0[12] with { block = "BlockRAM" };
50        ram signed 36  R1[12] with { block = "BlockRAM" };
51        ram signed 36  R2[12] with { block = "BlockRAM" };
52        ram signed 36  R3[12] with { block = "BlockRAM" };
53        ram signed 36  R4[12] with { block = "BlockRAM" };
54        ram signed 36  R5[12] with { block = "BlockRAM" };
55        ram signed 36  R6[12] with { block = "BlockRAM" };
56        ram signed 36  R7[12] with { block = "BlockRAM" };
57        ram signed 36  R8[12] with { block = "BlockRAM" };
58        ram signed 36  R9[12] with { block = "BlockRAM" };
59        ram signed 36 R10[12] with { block = "BlockRAM" };
60        ram signed 36 R11[12] with { block = "BlockRAM" };
61         //Deklarasjon og initialisering av kontrollvariabler
```

```
62        unsigned int 4 i; //Elementkontroll
63        unsigned int 4 j; //Omgangskontroll
64        unsigned int 1 ferdig;
65        unsigned int 1 start;
66        ferdig = 0;
67        start = 0;
68        //Oppstart av systemet
69        //Initialisering av variabler og porter
70        if (start ==1)
71        {
72        par
73        {
74        i =0;
75        j =0;
76        a0=0;
77        a1=0;
78        a2=0;
79        a3=0;
80        a4=0;
81        a5=0;
82        a6=0;
83        a7=0;
84        a8=0;
85        a9=0;
86        a10=0;
87        a11=0;
88        b0=0;
89        b1=0;
90        b2=0;
91        b3=0;
92        b4=0;
93        b5=0;
94        b6=0;
```

```
95          b7=0;
96          b8=0;
97          b9=0;
98          b10=0;
99          b11=0;
100         q0=0;
101         q1=0;
102         q2=0;
103         q3=0;
104         q4=0;
105         q5=0;
106         q6=0;
107         q7=0;
108         q8=0;
109         q9=0;
110         q10=0;
111         q11=0;
112         p0=0;
113         p1=0;
114         p2=0;
115         p3=0;
116         p4=0;
117         p5=0;
118         p6=0;
119         p7=0;
120         p8=0;
121         p9=0;
122         p10=0;
123         p11=0;
124         }
125         }
126         //En rad fra matrise A legges fast
127         //til hver SAPenhet.
```

```
128       par
129       {
130       a0  =  A0[i];
131       a1  =  A1[i];
132       a2  =  A2[i];
133       a3  =  A3[i];
134       a4  =  A4[i];
135       a5  =  A5[i];
136       a6  =  A6[i];
137       a7  =  A7[i];
138       a8  =  A8[i];
139       a9  =  A9[i];
140       a10 = A10[i];
141       a11 = A11[i];
142       }
143       par
144       {
145       {
146       //data tilordning for 1. syklus
147       if (j == 0)
148       {
149           par
150           {
151           b0  =  B0[i];
152           b1  =  B1[i];
153           b2  =  B2[i];
154           b3  =  B3[i];
155           b4  =  B4[i];
156           b5  =  B5[i];
157           b6  =  B6[i];
158           b7  =  B7[i];
159           b8  =  B8[i];
160           b9  =  B9[i];
```

```
161             b10 = B10[i];
162             b11 = B11[i];
163             }
164         }
165         //data tilordning for 2. syklus
166         if (j == 1)
167         {
168             par
169             {
170             b0  = B1[i];
171             b1  = B2[i];
172             b2  = B3[i];
173             b3  = B4[i];
174             b4  = B5[i];
175             b5  = B6[i];
176             b6  = B7[i];
177             b7  = B8[i];
178             b8  = B9[i];
179             b9  = B10[i];
180             b10 = B11[i];
181             b11 = B0[i];
182             }
183         }
184         //data tilordning for 3. syklus
185         if (j == 2)
186         {
187             par
188             {
189             b0  = B2[i];
190             b1  = B3[i];
191             b2  = B4[i];
192             b3  = B5[i];
193             b4  = B6[i];
```

```
194             b5  = B7[i];
195             b6  = B8[i];
196             b7  = B9[i];
197             b8  = B10[i];
198             b9  = B11[i];
199             b10 = B0[i];
200             b11 = B1[i];
201             }
202         }
203         //data tilordning for 4. syklus
204         if (j == 3)
205         {
206             par
207             {
208             b0  = B3[i];
209             b1  = B4[i];
210             b2  = B5[i];
211             b3  = B6[i];
212             b4  = B7[i];
213             b5  = B8[i];
214             b6  = B9[i];
215             b7  = B10[i];
216             b8  = B11[i];
217             b9  = B0[i];
218             b10 = B1[i];
219             b11 = B2[i];
220             }
221         }
222         //data tilordning for 5. syklus
223         if (j == 4)
224         {
225             par
226             {
```

```
227             b0  = B4[i];
228             b1  = B5[i];
229             b2  = B6[i];
230             b3  = B7[i];
231             b4  = B8[i];
232             b5  = B9[i];
233             b6  = B10[i];
234             b7  = B11[i];
235             b8  = B0[i];
236             b9  = B1[i];
237             b10 = B2[i];
238             b11 = B3[i];
239             }
240         }
241         //data tilordning for 6. syklus
242         if (j == 5)
243         {
244             par
245             {
246             b0  = B5[i];
247             b1  = B6[i];
248             b2  = B7[i];
249             b3  = B8[i];
250             b4  = B9[i];
251             b5  = B10[i];
252             b6  = B11[i];
253             b7  = B0[i];
254             b8  = B1[i];
255             b9  = B2[i];
256             b10 = B3[i];
257             b11 = B4[i];
258             }
259         }
```

```
260       //data tilordning for 7. syklus
261       if (j == 6)
262       {
263           par
264           {
265           b0  = B6[i];
266           b1  = B7[i];
267           b2  = B8[i];
268           b3  = B9[i];
269           b4  = B10[i];
270           b5  = B11[i];
271           b6  = B0[i];
272           b7  = B1[i];
273           b8  = B2[i];
274           b9  = B3[i];
275           b10 = B4[i];
276           b11 = B5[i];
277           }
278       }
279       //data tilordning for 8. syklus
280       if (j == 7)
281       {
282           par
283           {
284           b0  = B7[i];
285           b1  = B8[i];
286           b2  = B9[i];
287           b3  = B10[i];
288           b4  = B11[i];
289           b5  = B0[i];
290           b6  = B1[i];
291           b7  = B2[i];
292           b8  = B3[i];
```

```
293             b9  = B4[i];
294             b10 = B5[i];
295             b11 = B6[i];
296             }
297         }
298         //data tilordning for 9. syklus
299         if (j == 8)
300         {
301             par
302             {
303             b0  = B8[i];
304             b1  = B9[i];
305             b2  = B10[i];
306             b3  = B11[i];
307             b4  = B0[i];
308             b5  = B1[i];
309             b6  = B2[i];
310             b7  = B3[i];
311             b8  = B4[i];
312             b9  = B5[i];
313             b10 = B6[i];
314             b11 = B7[i];
315             }
316         }
317         //data tilordning for 10. syklus
318         if (j == 9)
319         {
320             par
321             {
322             b0  = B9[i];
323             b1  = B10[i];
324             b2  = B11[i];
325             b3  = B0[i];
```

```
326             b4  = B1[i];
327             b5  = B2[i];
328             b6  = B3[i];
329             b7  = B4[i];
330             b8  = B5[i];
331             b9  = B6[i];
332             b10 = B7[i];
333             b11 = B8[i];
334             }
335         }
336         //data tilordning for 11. syklus
337         if (j == 10)
338         {
339             par
340             {
341             b0  = B10[i];
342             b1  = B11[i];
343             b2  = B0[i];
344             b3  = B1[i];
345             b4  = B2[i];
346             b5  = B3[i];
347             b6  = B4[i];
348             b7  = B5[i];
349             b8  = B6[i];
350             b9  = B7[i];
351             b10 = B8[i];
352             b11 = B9[i];
353             }
354         }
355         //data tilordning for 12. syklus
356         if (j == 11)
357         {
358             par
```

```
359             {
360             b0  = B11[i];
361             b1  = B0[i];
362             b2  = B1[i];
363             b3  = B2[i];
364             b4  = B3[i];
365             b5  = B4[i];
366             b6  = B5[i];
367             b7  = B6[i];
368             b8  = B7[i];
369             b9  = B8[i];
370             b10 = B9[i];
371             b11 = B10[i];
372             }
373         }
374         }
375         do
376         {   //Elementene multipliseres til
377             //delprodukter
378             par
379             {
380             xilinxmult0(q0,a0,b0);
381             xilinxmult1(q1,a1,b1);
382             xilinxmult2(q2,a2,b2);
383             xilinxmult3(q3,a3,b3);
384             xilinxmult4(q4,a4,b4);
385             xilinxmult5(q5,a5,b5);
386             xilinxmult6(q6,a6,b6);
387             xilinxmult7(q7,a7,b7);
388             xilinxmult8(q8,a8,b8);
389             xilinxmult9(q9,a9,b9);
390             xilinxmult10(q10,a10,b10);
391             xilinxmult11(q11,a11,b11);
```

```
392           }
393           //Akkumulasjon av delproduktene
394           par
395           {
396           p0  = p0+q0;
397           p1  = p1+q1;
398           p2  = p2+q2;
399           p3  = p3+q3;
400           p4  = p4+q4;
401           p5  = p5+q5;
402           p6  = p6+q6;
403           p7  = p7+q7;
404           p8  = p8+q8;
405           p9  = p9+q9;
406           p10 = p10+q10;
407           p11 = p11+q11;
408           }
409           i=i+1;
410           } while (i<=11);
411           //til elementer i R som lagres
412       if (i==11)
413           {
414           par
415           {
416           R0[j]  = p0;
417           R1[j]  = p1;
418           R2[j]  = p2;
419           R3[j]  = p3;
420           R4[j]  = p4;
421           R5[j]  = p5;
422           R6[j]  = p6;
423           R7[j]  = p7;
424           R8[j]  = p8;
```

```
425             R9[j]  = p9;
426             R10[j] = p10;
427             R11[j] = p11;
428             }
429             //når j=11 er R beregnet
430             if (j==11)
431             {
432             ferdig = 1;
433             }
434             //Kontrollvariabelen j inkrementeres
435             // og neste syklus igangsettes.
436             i=0;
437             j = j+1;
438         }
439         //Hjelpevariabelene resettes
440         par
441         {
442         q0=0;
443         q1=0;
444         q2=0;
445         q3=0;
446         q4=0;
447         q5=0;
448         q6=0;
449         q7=0;
450         q8=0;
451         q9=0;
452         q10=0;
453         q11=0;
454         p0=0;
455         p1=0;
456         p2=0;
457         p3=0;
```

```
458         p4=0;
459         p5=0;
460         p6=0;
461         p7=0;
462         p8=0;
463         p9=0;
464         p10=0;
465         p11=0;
466         }
467     }
468     }
```

B.4 System for multiplikasjon av 4x4 matriser med universelle SAP enheter

```
1   ////////////////////////////////////////////////////////////
2   //   HändelC program for multiplikasjon av 12x12 matriser    //
3   //   med tilpassede SAPenheter                               //
4   //                                                           //
5   //   G H Andersen 2005                                       //
6   ////////////////////////////////////////////////////////////
7   //Definisjoner og header filer for 18bit Multiplier
8   // og ADMXRC2 FPGAkort.
9   #define ADMXRC2_CLOCK_RATE 100000000
10  #define ADMXRC2_USE_LCLK
11  #include "admxrc2.hch"
12  #include "xilinxmul.hch"
13  void main(void)
14  {
15      //deklarasjon av innganger og hjelpevariabler
16      signed int 18 a0,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,
17                    a12,a13,a14,a15,a16,a17,a18,a19,a20,a21
18                    ,a22,a23,a24,a25,a26,a27,a28,a29,a30,a31
19                    ,a32,a33,a34,a35,a36,a37,a38,a39,a40,a41
20                    ,a42,a43,a44,a45,a46,a47,a48,a49,a50,a51
21                    ,a52,a53,a54,a55,a56,a57,a58,a59,a60,a61
22                    ,a62,a63,a64,a65,a66,a67,a68,a69,a70,a71
23                    ,a72,a73,a74,a75,a76,a77,a78,a79,a80,a81
24                    ,a82,a83,a84,a85,a86,a87,a88,a89,a90,a91
25                    ,a92,a93,a94,a95,a96,a97,a98,a99,a100,a101
26                    ,a102,a103,a104,a105,a106,a107,a108,a109
27                    ,a110,a111,a112,a113,a114,a115,a116,a117
28                    ,a118,a119,a120,a121,a122,a123,a124,a125
29                    ,a126,a127,a128,a129,a130,a131,a132,a133
```

```
30                      ,a134,a135,a136,a137,a138,a139,a140,a141
31                      ,a142,a143;
32          signed int 18 b0,b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11,b12
33                      ,b13,b14,b15,b16,b17,b18,b19,b20,b21,b22
34                      ,b23,b24,b25,b26,b27,b28,b29,b30,b31,b32
35                      ,b33,b34,b35,b36,b37,b38,b39,b40,b41,b42
36                      ,b43,b44,b45,b46,b47,b48,b49,b50,b51,b52
37                      ,b53,b54,b55,b56,b57,b58,b59,b60,b61,b62
38                      ,b63,b64,b65,b66,b67,b68,b69,b70,b71,b72
39                      ,b73,b74,b75,b76,b77,b78,b79,b80,b81,b82
40                      ,b83,b84,b85,b86,b87,b88,b89,b90,b91,b92
41                      ,b93,b94,b95,b96,b97,b98,b99,b100,b101,b102
42                      ,b103,b104,b105,b106,b107,b108,b109,b110
43                      ,b111,b112,b113,b114,b115,b116,b117,b118
44                      ,b119,b120,b121,b122,b123,b124,b125,b126
45                      ,b127,b128,b129,b130,b131,b132,b133,b134
46                      ,b135,b136,b137,b138,b139,b140,b141,b142,b143;
47          signed int 36 q0,q1,q2,q3,q4,q5,q6,q7,q8,q9,q10,q11,q12
48                      ,q13,q14,q15,q16,q17,q18,q19,q20,q21,q22
49                      ,q23,q24,q25,q26,q27,q28,q29,q30,q31,q32
50                      ,q33,q34,q35,q36,q37,q38,q39,q40,q41,q42
51                      ,q43,q44,q45,q46,q47,q48,q49,q50,q51,q52
52                      ,q53,q54,q55,q56,q57,q58,q59,q60,q61,q62
53                      ,q63,q64,q65,q66,q67,q68,q69,q70,q71,q72
54                      ,q73,q74,q75,q76,q77,q78,q79,q80,q81,q82
55                      ,q83,q84,q85,q86,q87,q88,q89,q90,q91,q92
56                      ,q93,q94,q95,q96,q97,q98,q99,q100,q101,q102
57                      ,q103,q104,q105,q106,q107,q108,q109,q110
58                      ,q111,q112,q113,q114,q115,q116,q117,q118
59                      ,q119,q120,q121,q122,q123,q124,q125,q126
60                      ,q127,q128,q129,q130,q131,q132,q133,q134
61                      ,q135,q136,q137,q138,q139,q140,q141,q142,q143;
62          signed int 36 s0,s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12
```

```
63                      ,s13,s14,s15,s16,s17,s18,s19,s20,s21,s22
64                      ,s23,s24,s25,s26,s27,s28,s29,s30,s31,s32
65                      ,s33,s34,s35,s36,s37,s38,s39,s40,s41,s42
66                      ,s43,s44,s45,s46,s47,s48,s49,s50,s51,s52
67                      ,s53,s54,s55,s56,s57,s58,s59,s60,s61,s62
68                      ,s63,s64,s65,s66,s67,s68,s69,s70,s71;
69        signed int 36 t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,t10,t11,t12
70                      ,t13,t14,t15,t16,t17,t18,t19,t20,t21,t22
71                      ,t23,t24,t25,t26,t27,t28,t29,t30,t31,t32
72                      ,t33,t34,t35;
73        signed int 36 u0,u1,u2,u3,u4,u5,u6,u7,u8,u9,u10,u11
74                      ,u12,u13,u14,u15,u16,u17;
75        signed int 36 v0,v1,v2,v3,v4,v5,v6,v7,v8,v9,v10,v11;
76        signed int 36 p0,p1,p2,p3,p4,p5,p6,p7,p8,p9,p10,p11;
77        //RAM
78          //Radene i matrise A lagres i disse RAMene.
79        ram signed 18  A0[12]  with { block = "BlockRAM" };
80        ram signed 18  A1[12]  with { block = "BlockRAM" };
81        ram signed 18  A2[12]  with { block = "BlockRAM" };
82        ram signed 18  A3[12]  with { block = "BlockRAM" };
83        ram signed 18  A4[12]  with { block = "BlockRAM" };
84        ram signed 18  A5[12]  with { block = "BlockRAM" };
85        ram signed 18  A6[12]  with { block = "BlockRAM" };
86        ram signed 18  A7[12]  with { block = "BlockRAM" };
87        ram signed 18  A8[12]  with { block = "BlockRAM" };
88        ram signed 18  A9[12]  with { block = "BlockRAM" };
89        ram signed 18 A10[12]  with { block = "BlockRAM" };
90        ram signed 18 A11[12]  with { block = "BlockRAM" };
91       //Kolonnene i matrise B lagres i disse RAMene.
92        ram signed 18  B0[12]  with { block = "BlockRAM" };
93        ram signed 18  B1[12]  with { block = "BlockRAM" };
94        ram signed 18  B2[12]  with { block = "BlockRAM" };
95        ram signed 18  B3[12]  with { block = "BlockRAM" };
```

```
96        ram signed 18  B4[12]  with { block = "BlockRAM" };
97        ram signed 18  B5[12]  with { block = "BlockRAM" };
98        ram signed 18  B6[12]  with { block = "BlockRAM" };
99        ram signed 18  B7[12]  with { block = "BlockRAM" };
100       ram signed 18  B8[12]  with { block = "BlockRAM" };
101       ram signed 18  B9[12]  with { block = "BlockRAM" };
102       ram signed 18 B10[12]  with { block = "BlockRAM" };
103       ram signed 18 B11[12]  with { block = "BlockRAM" };
104       //Kolonnene i resultatmatrisen R lagres
105       //i disse RAMene
106       ram signed 36  R0[12] with { block = "BlockRAM" };
107       ram signed 36  R1[12] with { block = "BlockRAM" };
108       ram signed 36  R2[12] with { block = "BlockRAM" };
109       ram signed 36  R3[12] with { block = "BlockRAM" };
110       ram signed 36  R4[12] with { block = "BlockRAM" };
111       ram signed 36  R5[12] with { block = "BlockRAM" };
112       ram signed 36  R6[12] with { block = "BlockRAM" };
113       ram signed 36  R7[12] with { block = "BlockRAM" };
114       ram signed 36  R8[12] with { block = "BlockRAM" };
115       ram signed 36  R9[12] with { block = "BlockRAM" };
116       ram signed 36 R10[12] with { block = "BlockRAM" };
117       ram signed 36 R11[12] with { block = "BlockRAM" };
118       //deklarasjon og initialisering av kontrollvariabel
119       unsigned int 4 j;//Omgangskontroll
120       unsigned int 1 ferdig;
121       unsigned int 1 start;
122       start = 0;
123       ferdig = 0;
124       //Oppstart av systemet
125       //initialisering av hjelpevariabler og porter
126       if (start ==1)
127       {
128       par
```

```
129       {
130           j = 0;
131           q0=0;
132           q1=0;
133           q2=0;
134           q3=0;
135           q4=0;
136           q5=0;
137           q6=0;
138           q7=0;
139           q8=0;
140           q9=0;
141           q10=0;
142           q11=0;
143           q12=0;
144           q13=0;
145           q14=0;
146           q15=0;
147           q16=0;
148           q17=0;
149           q18=0;
150           q19=0;
151           q20=0;
152           q21=0;
153           q22=0;
154           q23=0;
155           q24=0;
156           q25=0;
157           q26=0;
158           q27=0;
159           q28=0;
160           q29=0;
161           q30=0;
```

```
162             q31=0;
163             q32=0;
164             q33=0;
165             q34=0;
166             q35=0;
167             q36=0;
168             q37=0;
169             q38=0;
170             q39=0;
171             q40=0;
172             q41=0;
173             q42=0;
174             q43=0;
175             q44=0;
176             q45=0;
177             q46=0;
178             q47=0;
179             q48=0;
180             q49=0;
181             q50=0;
182             q51=0;
183             q52=0;
184             q53=0;
185             q54=0;
186             q55=0;
187             q56=0;
188             q57=0;
189             q58=0;
190             q59=0;
191             q60=0;
192             q61=0;
193             q62=0;
194             q63=0;
```

```
195             q64=0;
196             q65=0;
197             q66=0;
198             q67=0;
199             q68=0;
200             q69=0;
201             q70=0;
202             q71=0;
203             q72=0;
204             q73=0;
205             q74=0;
206             q75=0;
207             q76=0;
208             q77=0;
209             q78=0;
210             q79=0;
211             q80=0;
212             q81=0;
213             q82=0;
214             q83=0;
215             q84=0;
216             q85=0;
217             q86=0;
218             q87=0;
219             q88=0;
220             q89=0;
221             q90=0;
222             q91=0;
223             q92=0;
224             q93=0;
225             q94=0;
226             q95=0;
227             q96=0;
```

```
228             q97=0;
229             q98=0;
230             q99=0;
231             q100=0;
232             q101=0;
233             q102=0;
234             q103=0;
235             q104=0;
236             q105=0;
237             q106=0;
238             q107=0;
239             q108=0;
240             q109=0;
241             q110=0;
242             q111=0;
243             q112=0;
244             q113=0;
245             q114=0;
246             q115=0;
247             q116=0;
248             q117=0;
249             q118=0;
250             q119=0;
251             q120=0;
252             q121=0;
253             q122=0;
254             q123=0;
255             q124=0;
256             q125=0;
257             q126=0;
258             q127=0;
259             q128=0;
260             q129=0;
```

```
261             q130=0;
262             q131=0;
263             q132=0;
264             q133=0;
265             q134=0;
266             q135=0;
267             q136=0;
268             q137=0;
269             q138=0;
270             q139=0;
271             q140=0;
272             q141=0;
273             q142=0;
274             q143=0;
275             p0=0;
276             p1=0;
277             p2=0;
278             p3=0;
279             p4=0;
280             p5=0;
281             p6=0;
282             p7=0;
283             p8=0;
284             p9=0;
285             p10=0;
286             p11=0;
287             }
288         }
289             //En rad fra matrise A legges fast
290             //til hver SAPenhet.
291             par
292             {
293             a0   = A0[0];
```

```
294             a12  = A1[0];
295             a24  = A2[0];
296             a36  = A3[0];
297             a48  = A4[0];
298             a60  = A5[0];
299             a72  = A6[0];
300             a84  = A7[0];
301             a96  = A8[0];
302             a108 = A9[0];
303             a120 = A10[0];
304             a132 = A11[0];
305             }
306             par
307             {
308             a1   = A0[1];
309             a13  = A1[1];
310             a25  = A2[1];
311             a37  = A3[1];
312             a49  = A4[1];
313             a61  = A5[1];
314             a73  = A6[1];
315             a85  = A7[1];
316             a97  = A8[1];
317             a109 = A9[1];
318             a121 = A10[1];
319             a133 = A11[1];
320             }
321             par
322             {
323             a2   = A0[2];
324             a14  = A1[2];
325             a26  = A2[2];
326             a38  = A3[2];
```

```
327           a51  = A4[2];
328           a62  = A5[2];
329           a74  = A6[2];
330           a86  = A7[2];
331           a98  = A8[2];
332           a110 = A9[2];
333           a122 = A10[2];
334           a134 = A11[2];
335           }
336           par
337           {
338           a3   = A0[3];
339           a15  = A1[3];
340           a27  = A2[3];
341           a39  = A3[3];
342           a51  = A4[3];
343           a63  = A5[3];
344           a75  = A6[3];
345           a87  = A7[3];
346           a99  = A8[3];
347           a111 = A9[3];
348           a123 = A10[3];
349           a135 = A11[3];
350           }
351           par
352           {
353           a4   = A0[4];
354           a16  = A1[4];
355           a28  = A2[4];
356           a40  = A3[4];
357           a52  = A4[4];
358           a64  = A5[4];
359           a76  = A6[4];
```

```
360             a88  = A7[4];
361             a100 = A8[4];
362             a112 = A9[4];
363             a124 = A10[4];
364             a136 = A11[4];
365             }
366             par
367             {
368             a5   = A0[5];
369             a17  = A1[5];
370             a29  = A2[5];
371             a41  = A3[5];
372             a53  = A4[5];
373             a65  = A5[5];
374             a77  = A6[5];
375             a89  = A7[5];
376             a101 = A8[5];
377             a113 = A9[5];
378             a125 = A10[5];
379             a137 = A11[5];
380             }
381             par
382             {
383             a6   = A0[6];
384             a18  = A1[6];
385             a30  = A2[6];
386             a42  = A3[6];
387             a54  = A4[6];
388             a66  = A5[6];
389             a78  = A6[6];
390             a90  = A7[6];
391             a102 = A8[6];
392             a114 = A9[6];
```

```
393           a126 = A10[6];
394           a138 = A11[6];
395           }
396           par
397           {
398           a7   = A0[7];
399           a19  = A1[7];
400           a31  = A2[7];
401           a43  = A3[7];
402           a55  = A4[7];
403           a67  = A5[7];
404           a79  = A6[7];
405           a91  = A7[7];
406           a103 = A8[7];
407           a115 = A9[7];
408           a127 = A10[7];
409           a139 = A11[7];
410           }
411           par
412           {
413           a8   = A0[8];
414           a20  = A1[8];
415           a32  = A2[8];
416           a44  = A3[8];
417           a56  = A4[8];
418           a68  = A5[8];
419           a80  = A6[8];
420           a92  = A7[8];
421           a104  = A8[8];
422           a126 = A9[8];
423           a128 = A10[8];
424           a140 = A11[8];
425           }
```

```
426             par
427             {
428             a9   = A0[9];
429             a21  = A1[9];
430             a33  = A2[9];
431             a45  = A3[9];
432             a57  = A4[9];
433             a69  = A5[9];
434             a81  = A6[9];
435             a93  = A7[9];
436             a105 = A8[9];
437             a117 = A9[9];
438             a129 = A10[9];
439             a141 = A11[9];
440             }
441             par
442             {
443             a10  = A0[10];
444             a22  = A1[10];
445             a34  = A2[10];
446             a46  = A3[10];
447             a58  = A4[10];
448             a70  = A5[10];
449             a82  = A6[10];
450             a94  = A7[10];
451             a106 = A8[10];
452             a118 = A9[10];
453             a130 = A10[10];
454             a142 = A11[10];
455             }
456             par
457             {
458             a11  = A0[11];
```

```
459             a23  = A1[11];
460             a35  = A2[11];
461             a47  = A3[11];
462             a59  = A4[11];
463             a71  = A5[11];
464             a83  = A6[11];
465             a95  = A7[11];
466             a107 = A8[11];
467             a119 = A9[11];
468             a131 = A10[11];
469             a143 = A11[11];
470             }
471
472         //kolonnene fra matrise B
473         //skiftes mellom SAPenhetene.
474         par
475         {
476         //data tilordning for 1. syklus
477         if (j == 0)
478         {
479             par
480             {
481             b0  =  B0[0];
482             b1  =  B1[0];
483             b2  =  B2[0];
484             b3  =  B3[0];
485             b4  =  B4[0];
486             b5  =  B5[0];
487             b6  =  B6[0];
488             b7  =  B7[0];
489             b8  =  B8[0];
490             b9  =  B9[0];
491             b10 = B10[0];
```

```
492             b11 = B11[0];
493             }
494             par
495             {
496             b12  =  B0[1];
497             b13  =  B1[1];
498             b14  =  B2[1];
499             b15  =  B3[1];
500             b16  =  B4[1];
501             b17  =  B5[1];
502             b18  =  B6[1];
503             b19  =  B7[1];
504             b20  =  B8[1];
505             b21  =  B9[1];
506             b22  = B10[1];
507             b23  = B11[1];
508             }
509             par
510             {
511             b24  =  B0[2];
512             b25  =  B1[2];
513             b26  =  B2[2];
514             b27  =  B3[2];
515             b28  =  B4[2];
516             b29  =  B5[2];
517             b30  =  B6[2];
518             b31  =  B7[2];
519             b32  =  B8[2];
520             b33  =  B9[2];
521             b34  = B10[2];
522             b35  = B11[2];
523             }
524             par
```

```
525         {
526         b36  =  B0[3];
527         b37  =  B1[3];
528         b38  =  B2[3];
529         b39  =  B3[3];
530         b40  =  B4[3];
531         b41  =  B5[3];
532         b42  =  B6[3];
533         b43  =  B7[3];
534         b44  =  B8[3];
535         b45  =  B9[3];
536         b46  = B10[3];
537         b47  = B11[3];
538         }
539         par
540         {
541         b48  =  B0[4];
542         b49  =  B1[4];
543         b50  =  B2[4];
544         b51  =  B3[4];
545         b52  =  B4[4];
546         b53  =  B5[4];
547         b54  =  B6[4];
548         b55  =  B7[4];
549         b56  =  B8[4];
550         b57  =  B9[4];
551         b58  = B10[4];
552         b59  = B11[4];
553         }
554         par
555         {
556         b60  =  B0[5];
557         b61  =  B1[5];
```

```
558             b62  =  B2[5];
559             b63  =  B3[5];
560             b64  =  B4[5];
561             b65  =  B5[5];
562             b66  =  B6[5];
563             b67  =  B7[5];
564             b68  =  B8[5];
565             b69  =  B9[5];
566             b70  = B10[5];
567             b71  = B11[5];
568             }
569             par
570             {
571             b72  =  B0[6];
572             b73  =  B1[6];
573             b74  =  B2[6];
574             b75  =  B3[6];
575             b76  =  B4[6];
576             b77  =  B5[6];
577             b78  =  B6[6];
578             b79  =  B7[6];
579             b80  =  B8[6];
580             b81  =  B9[6];
581             b82  = B10[6];
582             b83  = B11[6];
583             }
584             par
585             {
586             b84  =  B0[7];
587             b85  =  B1[7];
588             b86  =  B2[7];
589             b87  =  B3[7];
590             b88  =  B4[7];
```

```
591             b89  =  B5[7];
592             b90  =  B6[7];
593             b91  =  B7[7];
594             b92  =  B8[7];
595             b93  =  B9[7];
596             b94  =  B10[7];
597             b95  =  B11[7];
598             }
599             par
600             {
601             b96  =  B0[8];
602             b97  =  B1[8];
603             b98  =  B2[8];
604             b99  =  B3[8];
605             b100 =  B4[8];
606             b101 =  B5[8];
607             b102 =  B6[8];
608             b103 =  B7[8];
609             b104 =  B8[8];
610             b105 =  B9[8];
611             b106 = B10[8];
612             b107 = B11[8];
613             }
614             par
615             {
616             b108  =  B0[9];
617             b109  =  B1[9];
618             b110  =  B2[9];
619             b111  =  B3[9];
620             b112  =  B4[9];
621             b113  =  B5[9];
622             b114  =  B6[9];
623             b115  =  B7[9];
```

```
624             b116  =  B8[9];
625             b117  =  B9[9];
626             b118  = B10[9];
627             b119  = B11[9];
628             }
629             par
630             {
631             b120  =  B0[10];
632             b121  =  B1[10];
633             b122  =  B2[10];
634             b123  =  B3[10];
635             b124  =  B4[10];
636             b125  =  B5[10];
637             b126  =  B6[10];
638             b127  =  B7[10];
639             b128  =  B8[10];
640             b129  =  B9[10];
641             b130  = B10[10];
642             b131  = B11[10];
643             }
644             par
645             {
646             b132  =  B0[11];
647             b133  =  B1[11];
648             b134  =  B2[11];
649             b135  =  B3[11];
650             b136  =  B4[11];
651             b137  =  B5[11];
652             b138  =  B6[11];
653             b139  =  B7[11];
654             b140  =  B8[11];
655             b141  =  B9[11];
656             b142  = B10[11];
```

```
657             b143  = B11[11];
658             }
659         }
660         //data tilordning for 2. syklus
661         if (j == 1)
662         {
663             par
664             {
665             b0  =  B1[0];
666             b1  =  B2[0];
667             b2  =  B3[0];
668             b3  =  B4[0];
669             b4  =  B5[0];
670             b5  =  B6[0];
671             b6  =  B7[0];
672             b7  =  B8[0];
673             b8  =  B9[0];
674             b9  =  B10[0];
675             b10 = B11[0];
676             b11 = B0[0];
677             }
678             par
679             {
680             b12  =  B1[1];
681             b13  =  B2[1];
682             b14  =  B3[1];
683             b15  =  B4[1];
684             b16  =  B5[1];
685             b17  =  B6[1];
686             b18  =  B7[1];
687             b19  =  B8[1];
688             b20  =  B9[1];
689             b21  =  B10[1];
```

```
690           b22  = B11[1];
691           b23  = B0[1];
692           }
693           par
694           {
695           b24  =  B1[2];
696           b25  =  B2[2];
697           b26  =  B3[2];
698           b27  =  B4[2];
699           b28  =  B5[2];
700           b29  =  B6[2];
701           b30  =  B7[2];
702           b31  =  B8[2];
703           b32  =  B9[2];
704           b33  =  B10[2];
705           b34  = B11[2];
706           b35  = B0[2];
707           }
708           par
709           {
710           b36  =  B1[3];
711           b37  =  B2[3];
712           b38  =  B3[3];
713           b39  =  B4[3];
714           b40  =  B5[3];
715           b41  =  B6[3];
716           b42  =  B7[3];
717           b43  =  B8[3];
718           b44  =  B9[3];
719           b45  =  B10[3];
720           b46  = B11[3];
721           b47  = B0[3];
722           }
```

```
723             par
724             {
725             b48  =  B1[4];
726             b49  =  B2[4];
727             b50  =  B3[4];
728             b51  =  B4[4];
729             b52  =  B5[4];
730             b53  =  B6[4];
731             b54  =  B7[4];
732             b55  =  B8[4];
733             b56  =  B9[4];
734             b57  =  B10[4];
735             b58  = B11[4];
736             b59  = B0[4];
737             }
738             par
739             {
740             b60  =  B1[5];
741             b61  =  B2[5];
742             b62  =  B3[5];
743             b63  =  B4[5];
744             b64  =  B5[5];
745             b65  =  B6[5];
746             b66  =  B7[5];
747             b67  =  B8[5];
748             b68  =  B9[5];
749             b69  =  B10[5];
750             b70  = B11[5];
751             b71  = B0[5];
752             }
753             par
754             {
755             b72  =  B1[6];
```

```
756             b73  =  B2[6];
757             b74  =  B3[6];
758             b75  =  B4[6];
759             b76  =  B5[6];
760             b77  =  B6[6];
761             b78  =  B7[6];
762             b79  =  B8[6];
763             b80  =  B9[6];
764             b81  =  B10[6];
765             b82  = B11[6];
766             b83  = B0[6];
767             }
768             par
769             {
770             b84  =  B1[7];
771             b85  =  B2[7];
772             b86  =  B3[7];
773             b87  =  B4[7];
774             b88  =  B5[7];
775             b89  =  B6[7];
776             b90  =  B7[7];
777             b91  =  B8[7];
778             b92  =  B9[7];
779             b93  =  B10[7];
780             b94  =  B11[7];
781             b95  =  B0[7];
782             }
783             par
784             {
785             b96  =  B1[8];
786             b97  =  B2[8];
787             b98  =  B3[8];
788             b99  =  B4[8];
```

```
789             b100 =  B5[8];
790             b101 =  B6[8];
791             b102 =  B7[8];
792             b103 =  B8[8];
793             b104 =  B9[8];
794             b105 =  B10[8];
795             b106 = B11[8];
796             b107 = B0[8];
797             }
798             par
799             {
800             b108  =  B1[9];
801             b109  =  B2[9];
802             b110  =  B3[9];
803             b111  =  B4[9];
804             b112  =  B5[9];
805             b113  =  B6[9];
806             b114  =  B7[9];
807             b115  =  B8[9];
808             b116  =  B9[9];
809             b117  =  B10[9];
810             b118  = B11[9];
811             b119  = B0[9];
812             }
813             par
814             {
815             b120  =  B1[10];
816             b121  =  B2[10];
817             b122  =  B3[10];
818             b123  =  B4[10];
819             b124  =  B5[10];
820             b125  =  B6[10];
821             b126  =  B7[10];
```

```
822             b127  =  B8[10];
823             b128  =  B9[10];
824             b129  =  B10[10];
825             b130  = B11[10];
826             b131  = B0[10];
827             }
828             par
829             {
830             b132  =  B1[11];
831             b133  =  B2[11];
832             b134  =  B3[11];
833             b135  =  B4[11];
834             b136  =  B5[11];
835             b137  =  B6[11];
836             b138  =  B7[11];
837             b139  =  B8[11];
838             b140  =  B9[11];
839             b141  =  B10[11];
840             b142  = B11[11];
841             b143  = B0[11];
842             }
843         }
844         //data tilordning for 3. syklus
845         if (j == 2)
846         {
847             par
848             {
849             b0  =  B2[0];
850             b1  =  B3[0];
851             b2  =  B4[0];
852             b3  =  B5[0];
853             b4  =  B6[0];
854             b5  =  B7[0];
```

```
855             b6  =  B8[0];
856             b7  =  B9[0];
857             b8  =  B10[0];
858             b9  =  B11[0];
859             b10 = B0[0];
860             b11 = B1[0];
861             }
862             par
863             {
864             b12  =  B2[1];
865             b13  =  B3[1];
866             b14  =  B4[1];
867             b15  =  B5[1];
868             b16  =  B6[1];
869             b17  =  B7[1];
870             b18  =  B8[1];
871             b19  =  B9[1];
872             b20  =  B10[1];
873             b21  =  B11[1];
874             b22  = B0[1];
875             b23  = B1[1];
876             }
877             par
878             {
879             b24  =  B2[2];
880             b25  =  B3[2];
881             b26  =  B4[2];
882             b27  =  B5[2];
883             b28  =  B6[2];
884             b29  =  B7[2];
885             b30  =  B8[2];
886             b31  =  B9[2];
887             b32  =  B10[2];
```

```
888             b33  =  B11[2];
889             b34  = B0[2];
890             b35  = B1[2];
891             }
892             par
893             {
894             b36  =  B2[3];
895             b37  =  B3[3];
896             b38  =  B4[3];
897             b39  =  B5[3];
898             b40  =  B6[3];
899             b41  =  B7[3];
900             b42  =  B8[3];
901             b43  =  B9[3];
902             b44  =  B10[3];
903             b45  =  B11[3];
904             b46  = B0[3];
905             b47  = B1[3];
906             }
907             par
908             {
909             b48  =  B2[4];
910             b49  =  B3[4];
911             b50  =  B4[4];
912             b51  =  B5[4];
913             b52  =  B6[4];
914             b53  =  B7[4];
915             b54  =  B8[4];
916             b55  =  B9[4];
917             b56  =  B10[4];
918             b57  =  B11[4];
919             b58  = B0[4];
920             b59  = B1[4];
```

```
921           }
922           par
923           {
924           b60  =  B2[5];
925           b61  =  B3[5];
926           b62  =  B4[5];
927           b63  =  B5[5];
928           b64  =  B6[5];
929           b65  =  B7[5];
930           b66  =  B8[5];
931           b67  =  B9[5];
932           b68  =  B10[5];
933           b69  =  B11[5];
934           b70  = B0[5];
935           b71  = B1[5];
936           }
937           par
938           {
939           b72  =  B2[6];
940           b73  =  B3[6];
941           b74  =  B4[6];
942           b75  =  B5[6];
943           b76  =  B6[6];
944           b77  =  B7[6];
945           b78  =  B8[6];
946           b79  =  B9[6];
947           b80  =  B10[6];
948           b81  =  B11[6];
949           b82  = B0[6];
950           b83  = B1[6];
951           }
952           par
953           {
```

```
954             b84  =  B2[7];
955             b85  =  B3[7];
956             b86  =  B4[7];
957             b87  =  B5[7];
958             b88  =  B6[7];
959             b89  =  B7[7];
960             b90  =  B8[7];
961             b91  =  B9[7];
962             b92  =  B10[7];
963             b93  =  B11[7];
964             b94  =  B0[7];
965             b95  =  B1[7];
966             }
967             par
968             {
969             b96  =  B2[8];
970             b97  =  B3[8];
971             b98  =  B4[8];
972             b99  =  B5[8];
973             b100 =  B6[8];
974             b101 =  B7[8];
975             b102 =  B8[8];
976             b103 =  B9[8];
977             b104 =  B10[8];
978             b105 =  B11[8];
979             b106 = B0[8];
980             b107 = B1[8];
981             }
982             par
983             {
984             b108  =  B2[9];
985             b109  =  B3[9];
986             b110  =  B4[9];
```

```
987             b111  =  B5[9];
988             b112  =  B6[9];
989             b113  =  B7[9];
990             b114  =  B8[9];
991             b115  =  B9[9];
992             b116  =  B10[9];
993             b117  =  B11[9];
994             b118  = B0[9];
995             b119  = B1[9];
996             }
997             par
998             {
999             b120  =  B2[10];
1000             b121  =  B3[10];
1001            b122  =  B4[10];
1002            b123  =  B5[10];
1003            b124  =  B6[10];
1004            b125  =  B7[10];
1005            b126  =  B8[10];
1006            b127  =  B9[10];
1007            b128  =  B10[10];
1008            b129  =  B11[10];
1009            b130  = B0[10];
1010            b131  = B1[10];
1011            }
1012            par
1013            {
1014            b132  =  B2[11];
1015            b133  =  B3[11];
1016            b134  =  B4[11];
1017            b135  =  B5[11];
1018            b136  =  B6[11];
1019            b137  =  B7[11];
```

```
1020            b138  =  B8[11];
1021            b139  =  B9[11];
1022            b140  =  B10[11];
1023            b141  =  B11[11];
1024            b142  = B0[11];
1025            b143  = B1[11];
1026            }
1027        }
1028        //data tilordning for 4. syklus
1029        if (j == 3)
1030        {
1031            par
1032            {
1033            b0  =  B3[0];
1034            b1  =  B4[0];
1035            b2  =  B5[0];
1036            b3  =  B6[0];
1037            b4  =  B7[0];
1038            b5  =  B8[0];
1039            b6  =  B9[0];
1040            b7  =  B10[0];
1041            b8  =  B11[0];
1042            b9  =  B0[0];
1043            b10 = B1[0];
1044            b11 = B2[0];
1045            }
1046            par
1047            {
1048            b12  =  B3[1];
1049            b13  =  B4[1];
1050            b14  =  B5[1];
1051            b15  =  B6[1];
1052            b16  =  B7[1];
```

```
1053          b17  =  B8[1];
1054          b18  =  B9[1];
1055          b19  =  B10[1];
1056          b20  =  B11[1];
1057          b21  =  B0[1];
1058          b22  = B1[1];
1059          b23  = B2[1];
1060          }
1061          par
1062          {
1063          b24  =  B3[2];
1064          b25  =  B4[2];
1065          b26  =  B5[2];
1066          b27  =  B6[2];
1067          b28  =  B7[2];
1068          b29  =  B8[2];
1069          b30  =  B9[2];
1070          b31  =  B10[2];
1071          b32  =  B11[2];
1072          b33  =  B0[2];
1073          b34  = B1[2];
1074          b35  = B2[2];
1075          }
1076          par
1077          {
1078          b36  =  B3[3];
1079          b37  =  B4[3];
1080          b38  =  B5[3];
1081          b39  =  B6[3];
1082          b40  =  B7[3];
1083          b41  =  B8[3];
1084          b42  =  B9[3];
1085          b43  =  B10[3];
```

```
1086          b44  =  B11[3];
1087          b45  =  B0[3];
1088          b46  = B1[3];
1089          b47  = B2[3];
1090          }
1091          par
1092          {
1093          b48  =  B3[4];
1094          b49  =  B4[4];
1095          b50  =  B5[4];
1096          b51  =  B6[4];
1097          b52  =  B7[4];
1098          b53  =  B8[4];
1099          b54  =  B9[4];
1100          b55  =  B10[4];
1101          b56  =  B11[4];
1102          b57  =  B0[4];
1103          b58  = B1[4];
1104          b59  = B2[4];
1105          }
1106          par
1107          {
1108          b60  =  B3[5];
1109          b61  =  B4[5];
1110          b62  =  B5[5];
1111          b63  =  B6[5];
1112          b64  =  B7[5];
1113          b65  =  B8[5];
1114          b66  =  B9[5];
1115          b67  =  B10[5];
1116          b68  =  B11[5];
1117          b69  =  B0[5];
1118          b70  = B1[5];
```

```
1119          b71  = B2[5];
1120          }
1121          par
1122          {
1123          b72  =  B3[6];
1124          b73  =  B4[6];
1125          b74  =  B5[6];
1126          b75  =  B6[6];
1127          b76  =  B7[6];
1128          b77  =  B8[6];
1129          b78  =  B9[6];
1130          b79  =  B10[6];
1131          b80  =  B11[6];
1132          b81  =  B0[6];
1133          b82  = B1[6];
1134          b83  = B2[6];
1135          }
1136          par
1137          {
1138          b84  =  B3[7];
1139          b85  =  B4[7];
1140          b86  =  B5[7];
1141          b87  =  B6[7];
1142          b88  =  B7[7];
1143          b89  =  B8[7];
1144          b90  =  B9[7];
1145          b91  =  B10[7];
1146          b92  =  B11[7];
1147          b93  =  B0[7];
1148          b94  =  B1[7];
1149          b95  =  B2[7];
1150          }
1151          par
```

```
1152          {
1153          b96  =  B3[8];
1154          b97  =  B4[8];
1155          b98  =  B5[8];
1156          b99  =  B6[8];
1157          b100 =  B7[8];
1158          b101 =  B8[8];
1159          b102 =  B9[8];
1160          b103 =  B10[8];
1161          b104 =  B11[8];
1162          b105 =  B0[8];
1163          b106 = B1[8];
1164          b107 = B2[8];
1165          }
1166          par
1167          {
1168          b108  =  B3[9];
1169          b109  =  B4[9];
1170          b110  =  B5[9];
1171          b111  =  B6[9];
1172          b112  =  B7[9];
1173          b113  =  B8[9];
1174          b114  =  B9[9];
1175          b115  =  B10[9];
1176          b116  =  B11[9];
1177          b117  =  B0[9];
1178          b118  = B1[9];
1179          b119  = B2[9];
1180          }
1181          par
1182          {
1183          b120  =  B3[10];
1184          b121  =  B4[10];
```

```
1185            b122  =  B5[10];
1186            b123  =  B6[10];
1187            b124  =  B7[10];
1188            b125  =  B8[10];
1189            b126  =  B9[10];
1190            b127  =  B10[10];
1191            b128  =  B11[10];
1192            b129  =  B0[10];
1193            b130  = B1[10];
1194            b131  = B2[10];
1195            }
1196            par
1197            {
1198            b132  =  B3[11];
1199            b133  =  B4[11];
1200            b134  =  B5[11];
1201            b135  =  B6[11];
1202            b136  =  B7[11];
1203            b137  =  B8[11];
1204            b138  =  B9[11];
1205            b139  =  B10[11];
1206            b140  =  B11[11];
1207            b141  =  B0[11];
1208            b142  = B1[11];
1209            b143  = B2[11];
1210            }
1211        }
1212        //data tilordning for 5. syklus
1213        if (j == 4)
1214        {
1215            par
1216            {
1217            b0  =  B4[0];
```

```
1218          b1  =  B5[0];
1219          b2  =  B6[0];
1220          b3  =  B7[0];
1221          b4  =  B8[0];
1222          b5  =  B9[0];
1223          b6  =  B10[0];
1224          b7  =  B11[0];
1225          b8  =  B0[0];
1226          b9  =  B1[0];
1227          b10 = B2[0];
1228          b11 = B3[0];
1229          }
1230          par
1231          {
1232          b12  =  B4[1];
1233          b13  =  B5[1];
1234          b14  =  B6[1];
1235          b15  =  B7[1];
1236          b16  =  B8[1];
1237          b17  =  B9[1];
1238          b18  =  B10[1];
1239          b19  =  B11[1];
1240          b20  =  B0[1];
1241          b21  =  B1[1];
1242          b22  = B2[1];
1243          b23  = B3[1];
1244          }
1245          par
1246          {
1247          b24  =  B4[2];
1248          b25  =  B5[2];
1249          b26  =  B6[2];
1250          b27  =  B7[2];
```

```
1251          b28  =  B8[2];
1252          b29  =  B9[2];
1253          b30  =  B10[2];
1254          b31  =  B11[2];
1255          b32  =  B0[2];
1256          b33  =  B1[2];
1257          b34  = B2[2];
1258          b35  = B3[2];
1259          }
1260          par
1261          {
1262          b36  =  B4[3];
1263          b37  =  B5[3];
1264          b38  =  B6[3];
1265          b39  =  B7[3];
1266          b40  =  B8[3];
1267          b41  =  B9[3];
1268          b42  =  B10[3];
1269          b43  =  B11[3];
1270          b44  =  B0[3];
1271          b45  =  B1[3];
1272          b46  = B2[3];
1273          b47  = B3[3];
1274          }
1275          par
1276          {
1277          b48  =  B4[4];
1278          b49  =  B5[4];
1279          b50  =  B6[4];
1280          b51  =  B7[4];
1281          b52  =  B8[4];
1282          b53  =  B9[4];
1283          b54  =  B10[4];
```

```
1284          b55  =  B11[4];
1285          b56  =  B0[4];
1286          b57  =  B1[4];
1287          b58  = B2[4];
1288          b59  = B3[4];
1289          }
1290          par
1291          {
1292          b60  =  B4[5];
1293          b61  =  B5[5];
1294          b62  =  B6[5];
1295          b63  =  B7[5];
1296          b64  =  B8[5];
1297          b65  =  B9[5];
1298          b66  =  B10[5];
1299          b67  =  B11[5];
1300          b68  =  B0[5];
1301          b69  =  B1[5];
1302          b70  = B2[5];
1303          b71  = B3[5];
1304          }
1305          par
1306          {
1307          b72  =  B4[6];
1308          b73  =  B5[6];
1309          b74  =  B6[6];
1310          b75  =  B7[6];
1311          b76  =  B8[6];
1312          b77  =  B9[6];
1313          b78  =  B10[6];
1314          b79  =  B11[6];
1315          b80  =  B0[6];
1316          b81  =  B1[6];
```

```
1317          b82  = B2[6];
1318          b83  = B3[6];
1319          }
1320          par
1321          {
1322          b84  =  B4[7];
1323          b85  =  B5[7];
1324          b86  =  B6[7];
1325          b87  =  B7[7];
1326          b88  =  B8[7];
1327          b89  =  B9[7];
1328          b90  =  B10[7];
1329          b91  =  B11[7];
1330          b92  =  B0[7];
1331          b93  =  B1[7];
1332          b94  =  B2[7];
1333          b95  =  B3[7];
1334          }
1335          par
1336          {
1337          b96  =  B4[8];
1338          b97  =  B5[8];
1339          b98  =  B6[8];
1340          b99  =  B7[8];
1341          b100 =  B8[8];
1342          b101 =  B9[8];
1343          b102 =  B10[8];
1344          b103 =  B11[8];
1345          b104 =  B0[8];
1346          b105 =  B1[8];
1347          b106 = B2[8];
1348          b107 = B3[8];
1349          }
```

```
1350          par
1351          {
1352          b108  =  B4[9];
1353          b109  =  B5[9];
1354          b110  =  B6[9];
1355          b111  =  B7[9];
1356          b112  =  B8[9];
1357          b113  =  B9[9];
1358          b114  =  B10[9];
1359          b115  =  B11[9];
1360          b116  =  B0[9];
1361          b117  =  B1[9];
1362          b118  = B2[9];
1363          b119  = B3[9];
1364          }
1365          par
1366          {
1367          b120  =  B4[10];
1368          b121  =  B5[10];
1369          b122  =  B6[10];
1370          b123  =  B7[10];
1371          b124  =  B8[10];
1372          b125  =  B9[10];
1373          b126  =  B10[10];
1374          b127  =  B11[10];
1375          b128  =  B0[10];
1376          b129  =  B1[10];
1377          b130  = B2[10];
1378          b131  = B3[10];
1379          }
1380          par
1381          {
1382          b132  =  B4[11];
```

```
1383            b133  =  B5[11];
1384            b134  =  B6[11];
1385            b135  =  B7[11];
1386            b136  =  B8[11];
1387            b137  =  B9[11];
1388            b138  =  B10[11];
1389            b139  =  B11[11];
1390            b140  =  B0[11];
1391            b141  =  B1[11];
1392            b142  = B2[11];
1393            b143  = B3[11];
1394            }
1395        }
1396        //data tilordning for 6. syklus
1397        if (j == 5)
1398        {
1399                par
1400            {
1401            b0  =  B5[0];
1402            b1  =  B6[0];
1403            b2  =  B7[0];
1404            b3  =  B8[0];
1405            b4  =  B9[0];
1406            b5  =  B10[0];
1407            b6  =  B11[0];
1408            b7  =  B0[0];
1409            b8  =  B1[0];
1410            b9  =  B2[0];
1411            b10 = B3[0];
1412            b11 = B4[0];
1413            }
1414            par
1415            {
```

```
1416          b12  =  B5[1];
1417          b13  =  B6[1];
1418          b14  =  B7[1];
1419          b15  =  B8[1];
1420          b16  =  B9[1];
1421          b17  =  B10[1];
1422          b18  =  B11[1];
1423          b19  =  B0[1];
1424          b20  =  B1[1];
1425          b21  =  B2[1];
1426          b22  = B3[1];
1427          b23  = B4[1];
1428          }
1429          par
1430          {
1431          b24  =  B5[2];
1432          b25  =  B6[2];
1433          b26  =  B7[2];
1434          b27  =  B8[2];
1435          b28  =  B9[2];
1436          b29  =  B10[2];
1437          b30  =  B11[2];
1438          b31  =  B0[2];
1439          b32  =  B1[2];
1440          b33  =  B2[2];
1441          b34  = B3[2];
1442          b35  = B4[2];
1443          }
1444          par
1445          {
1446          b36  =  B5[3];
1447          b37  =  B6[3];
1448          b38  =  B7[3];
```

```
1449          b39  =  B8[3];
1450          b40  =  B9[3];
1451          b41  =  B10[3];
1452          b42  =  B11[3];
1453          b43  =  B0[3];
1454          b44  =  B1[3];
1455          b45  =  B2[3];
1456          b46  = B3[3];
1457          b47  = B4[3];
1458          }
1459          par
1460          {
1461          b48  =  B5[4];
1462          b49  =  B6[4];
1463          b50  =  B7[4];
1464          b51  =  B8[4];
1465          b52  =  B9[4];
1466          b53  =  B10[4];
1467          b54  =  B11[4];
1468          b55  =  B0[4];
1469          b56  =  B1[4];
1470          b57  =  B2[4];
1471          b58  = B3[4];
1472          b59  = B4[4];
1473          }
1474          par
1475          {
1476          b60  =  B5[5];
1477          b61  =  B6[5];
1478          b62  =  B7[5];
1479          b63  =  B8[5];
1480          b64  =  B9[5];
1481          b65  =  B10[5];
```

```
1482          b66  =  B11[5];
1483          b67  =  B0[5];
1484          b68  =  B1[5];
1485          b69  =  B2[5];
1486          b70  = B3[5];
1487          b71  = B4[5];
1488          }
1489          par
1490          {
1491          b72  =  B5[6];
1492          b73  =  B6[6];
1493          b74  =  B7[6];
1494          b75  =  B8[6];
1495          b76  =  B9[6];
1496          b77  =  B10[6];
1497          b78  =  B11[6];
1498          b79  =  B0[6];
1499          b80  =  B1[6];
1500          b81  =  B2[6];
1501          b82  = B3[6];
1502          b83  = B4[6];
1503          }
1504          par
1505          {
1506          b84  =  B5[7];
1507          b85  =  B6[7];
1508          b86  =  B7[7];
1509          b87  =  B8[7];
1510          b88  =  B9[7];
1511          b89  =  B10[7];
1512          b90  =  B11[7];
1513          b91  =  B0[7];
1514          b92  =  B1[7];
```

```
1515          b93  =  B2[7];
1516          b94  =  B3[7];
1517          b95  =  B4[7];
1518          }
1519          par
1520          {
1521          b96  =  B5[8];
1522          b97  =  B6[8];
1523          b98  =  B7[8];
1524          b99  =  B8[8];
1525          b100 =  B9[8];
1526          b101 =  B10[8];
1527          b102 =  B11[8];
1528          b103 =  B0[8];
1529          b104 =  B1[8];
1530          b105 =  B2[8];
1531          b106 = B3[8];
1532          b107 = B4[8];
1533          }
1534          par
1535          {
1536          b108  =  B5[9];
1537          b109  =  B6[9];
1538          b110  =  B7[9];
1539          b111  =  B8[9];
1540          b112  =  B9[9];
1541          b113  =  B10[9];
1542          b114  =  B11[9];
1543          b115  =  B0[9];
1544          b116  =  B1[9];
1545          b117  =  B2[9];
1546          b118  = B3[9];
1547          b119  = B4[9];
```

```
1548          }
1549          par
1550          {
1551          b120  =  B5[10];
1552          b121  =  B6[10];
1553          b122  =  B7[10];
1554          b123  =  B8[10];
1555          b124  =  B9[10];
1556          b125  =  B10[10];
1557          b126  =  B11[10];
1558          b127  =  B0[10];
1559          b128  =  B1[10];
1560          b129  =  B2[10];
1561          b130  = B3[10];
1562          b131  = B4[10];
1563          }
1564          par
1565          {
1566          b132  =  B5[11];
1567          b133  =  B6[11];
1568          b134  =  B7[11];
1569          b135  =  B8[11];
1570          b136  =  B9[11];
1571          b137  =  B10[11];
1572          b138  =  B11[11];
1573          b139  =  B0[11];
1574          b140  =  B1[11];
1575          b141  =  B2[11];
1576          b142  = B3[11];
1577          b143  = B4[11];
1578          }
1579      }
1580      //data tilordning for 7. syklus
```

```
1581      if (j == 6)
1582      {
1583          par
1584          {
1585          b0  =  B6[0];
1586          b1  =  B7[0];
1587          b2  =  B8[0];
1588          b3  =  B9[0];
1589          b4  =  B10[0];
1590          b5  =  B11[0];
1591          b6  =  B0[0];
1592          b7  =  B1[0];
1593          b8  =  B2[0];
1594          b9  =  B3[0];
1595          b10 = B4[0];
1596          b11 = B5[0];
1597          }
1598          par
1599          {
1600          b12  =  B6[1];
1601          b13  =  B7[1];
1602          b14  =  B8[1];
1603          b15  =  B9[1];
1604          b16  =  B10[1];
1605          b17  =  B11[1];
1606          b18  =  B0[1];
1607          b19  =  B1[1];
1608          b20  =  B2[1];
1609          b21  =  B3[1];
1610          b22  = B4[1];
1611          b23  = B5[1];
1612          }
1613          par
```

```
1614          {
1615          b24  =  B6[2];
1616          b25  =  B7[2];
1617          b26  =  B8[2];
1618          b27  =  B9[2];
1619          b28  =  B10[2];
1620          b29  =  B11[2];
1621          b30  =  B0[2];
1622          b31  =  B1[2];
1623          b32  =  B2[2];
1624          b33  =  B3[2];
1625          b34  = B4[2];
1626          b35  = B5[2];
1627          }
1628          par
1629          {
1630          b36  =  B6[3];
1631          b37  =  B7[3];
1632          b38  =  B8[3];
1633          b39  =  B9[3];
1634          b40  =  B10[3];
1635          b41  =  B11[3];
1636          b42  =  B0[3];
1637          b43  =  B1[3];
1638          b44  =  B2[3];
1639          b45  =  B3[3];
1640          b46  = B4[3];
1641          b47  = B5[3];
1642          }
1643          par
1644          {
1645          b48  =  B6[4];
1646          b49  =  B7[4];
```

```
1647          b50  =  B8[4];
1648          b51  =  B9[4];
1649          b52  =  B10[4];
1650          b53  =  B11[4];
1651          b54  =  B0[4];
1652          b55  =  B1[4];
1653          b56  =  B2[4];
1654          b57  =  B3[4];
1655          b58  = B4[4];
1656          b59  = B5[4];
1657          }
1658          par
1659          {
1660          b60  =  B6[5];
1661          b61  =  B7[5];
1662          b62  =  B8[5];
1663          b63  =  B9[5];
1664          b64  =  B10[5];
1665          b65  =  B11[5];
1666          b66  =  B0[5];
1667          b67  =  B1[5];
1668          b68  =  B2[5];
1669          b69  =  B3[5];
1670          b70  = B4[5];
1671          b71  = B5[5];
1672          }
1673          par
1674          {
1675          b72  =  B6[6];
1676          b73  =  B7[6];
1677          b74  =  B8[6];
1678          b75  =  B9[6];
1679          b76  =  B10[6];
```

```
1680          b77  =  B11[6];
1681          b78  =  B0[6];
1682          b79  =  B1[6];
1683          b80  =  B2[6];
1684          b81  =  B3[6];
1685          b82  = B4[6];
1686          b83  = B5[6];
1687          }
1688          par
1689          {
1690          b84  =  B6[7];
1691          b85  =  B7[7];
1692          b86  =  B8[7];
1693          b87  =  B9[7];
1694          b88  =  B10[7];
1695          b89  =  B11[7];
1696          b90  =  B0[7];
1697          b91  =  B1[7];
1698          b92  =  B2[7];
1699          b93  =  B3[7];
1700          b94  =  B4[7];
1701          b95  =  B5[7];
1702          }
1703          par
1704          {
1705          b96  =  B6[8];
1706          b97  =  B7[8];
1707          b98  =  B8[8];
1708          b99  =  B9[8];
1709          b100 =  B10[8];
1710          b101 =  B11[8];
1711          b102 =  B0[8];
1712          b103 =  B1[8];
```

```
1713          b104 =  B2[8];
1714          b105 =  B3[8];
1715          b106 = B4[8];
1716          b107 = B5[8];
1717          }
1718          par
1719          {
1720          b108  =  B6[9];
1721          b109  =  B7[9];
1722          b110  =  B8[9];
1723          b111  =  B9[9];
1724          b112  =  B10[9];
1725          b113  =  B11[9];
1726          b114  =  B0[9];
1727          b115  =  B1[9];
1728          b116  =  B2[9];
1729          b117  =  B3[9];
1730          b118  = B4[9];
1731          b119  = B5[9];
1732          }
1733          par
1734          {
1735          b120  =  B6[10];
1736          b121  =  B7[10];
1737          b122  =  B8[10];
1738          b123  =  B9[10];
1739          b124  =  B10[10];
1740          b125  =  B11[10];
1741          b126  =  B0[10];
1742          b127  =  B1[10];
1743          b128  =  B2[10];
1744          b129  =  B3[10];
1745          b130  = B4[10];
```

```
1746            b131  = B5[10];
1747            }
1748            par
1749            {
1750            b132  =  B6[11];
1751            b133  =  B7[11];
1752            b134  =  B8[11];
1753            b135  =  B9[11];
1754            b136  =  B10[11];
1755            b137  =  B11[11];
1756            b138  =  B0[11];
1757            b139  =  B1[11];
1758            b140  =  B2[11];
1759            b141  =  B3[11];
1760            b142  = B4[11];
1761            b143  = B5[11];
1762            }
1763        }
1764        //data tilordning for 8. syklus
1765        if (j == 7)
1766        {
1767            par
1768            {
1769            b0  =  B7[0];
1770            b1  =  B8[0];
1771            b2  =  B9[0];
1772            b3  =  B10[0];
1773            b4  =  B11[0];
1774            b5  =  B0[0];
1775            b6  =  B1[0];
1776            b7  =  B2[0];
1777            b8  =  B3[0];
1778            b9  =  B4[0];
```

```
1779          b10 =  B5[0];
1780          b11 =  B6[0];
1781          }
1782          par
1783          {
1784          b12  =  B7[1];
1785          b13  =  B8[1];
1786          b14  =  B9[1];
1787          b15  =  B10[1];
1788          b16  =  B11[1];
1789          b17  =  B0[1];
1790          b18  =  B1[1];
1791          b19  =  B2[1];
1792          b20  =  B3[1];
1793          b21  =  B4[1];
1794          b22  =  B5[1];
1795          b23  =  B6[1];
1796          }
1797          par
1798          {
1799          b24  =  B7[2];
1800          b25  =  B8[2];
1801          b26  =  B9[2];
1802          b27  =  B10[2];
1803          b28  =  B11[2];
1804          b29  =  B0[2];
1805          b30  =  B1[2];
1806          b31  =  B2[2];
1807          b32  =  B3[2];
1808          b33  =  B4[2];
1809          b34  =  B5[2];
1810          b35  =  B6[2];
1811          }
```

```
1812          par
1813          {
1814          b36  =  B7[3];
1815          b37  =  B8[3];
1816          b38  =  B9[3];
1817          b39  =  B10[3];
1818          b40  =  B11[3];
1819          b41  =  B0[3];
1820          b42  =  B1[3];
1821          b43  =  B2[3];
1822          b44  =  B3[3];
1823          b45  =  B4[3];
1824          b46  =  B5[3];
1825          b47  =  B6[3];
1826          }
1827          par
1828          {
1829          b48  =  B7[4];
1830          b49  =  B8[4];
1831          b50  =  B9[4];
1832          b51  =  B10[4];
1833          b52  =  B11[4];
1834          b53  =  B0[4];
1835          b54  =  B1[4];
1836          b55  =  B2[4];
1837          b56  =  B3[4];
1838          b57  =  B4[4];
1839          b58  =  B5[4];
1840          b59  =  B6[4];
1841          }
1842          par
1843          {
1844          b60  =  B7[5];
```

```
1845          b61  =  B8[5];
1846          b62  =  B9[5];
1847          b63  =  B10[5];
1848          b64  =  B11[5];
1849          b65  =  B0[5];
1850          b66  =  B1[5];
1851          b67  =  B2[5];
1852          b68  =  B3[5];
1853          b69  =  B4[5];
1854          b70  =  B5[5];
1855          b71  =  B6[5];
1856          }
1857          par
1858          {
1859          b72  =  B7[6];
1860          b73  =  B8[6];
1861          b74  =  B9[6];
1862          b75  =  B10[6];
1863          b76  =  B11[6];
1864          b77  =  B0[6];
1865          b78  =  B1[6];
1866          b79  =  B2[6];
1867          b80  =  B3[6];
1868          b81  =  B4[6];
1869          b82  =  B5[6];
1870          b83  =  B6[6];
1871          }
1872          par
1873          {
1874          b84  =  B7[7];
1875          b85  =  B8[7];
1876          b86  =  B9[7];
1877          b87  =  B10[7];
```

```
1878          b88  =  B11[7];
1879          b89  =  B0[7];
1880          b90  =  B1[7];
1881          b91  =  B2[7];
1882          b92  =  B3[7];
1883          b93  =  B4[7];
1884          b94  =  B5[7];
1885          b95  =  B6[7];
1886          }
1887          par
1888          {
1889          b96  =  B7[8];
1890          b97  =  B8[8];
1891          b98  =  B9[8];
1892          b99  =  B10[8];
1893          b100 =  B11[8];
1894          b101 =  B0[8];
1895          b102 =  B1[8];
1896          b103 =  B2[8];
1897          b104 =  B3[8];
1898          b105 =  B4[8];
1899          b106 =  B5[8];
1900          b107 =  B6[8];
1901          }
1902          par
1903          {
1904          b108  =  B7[9];
1905          b109  =  B8[9];
1906          b110  =  B9[9];
1907          b111  =  B10[9];
1908          b112  =  B11[9];
1909          b113  =  B0[9];
1910          b114  =  B1[9];
```

```
1911          b115  =  B2[9];
1912          b116  =  B3[9];
1913          b117  =  B4[9];
1914          b118  =  B5[9];
1915          b119  =  B6[9];
1916          }
1917          par
1918          {
1919          b120  =  B7[10];
1920          b121  =  B8[10];
1921          b122  =  B9[10];
1922          b123  =  B10[10];
1923          b124  =  B11[10];
1924          b125  =  B0[10];
1925          b126  =  B1[10];
1926          b127  =  B2[10];
1927          b128  =  B3[10];
1928          b129  =  B4[10];
1929          b130  =  B5[10];
1930          b131  =  B6[10];
1931          }
1932          par
1933          {
1934          b132  =  B7[11];
1935          b133  =  B8[11];
1936          b134  =  B9[11];
1937          b135  =  B10[11];
1938          b136  =  B11[11];
1939          b137  =  B0[11];
1940          b138  =  B1[11];
1941          b139  =  B2[11];
1942          b140  =  B3[11];
1943          b141  =  B4[11];
```

```
1944            b142  =  B5[11];
1945            b143  =  B6[11];
1946            }
1947        }
1948        //data tilordning for 9. syklus
1949        if (j == 8)
1950        {
1951            par
1952            {
1953            b0  =  B8[0];
1954            b1  =  B9[0];
1955            b2  =  B10[0];
1956            b3  =  B11[0];
1957            b4  =  B0[0];
1958            b5  =  B1[0];
1959            b6  =  B2[0];
1960            b7  =  B3[0];
1961            b8  =  B4[0];
1962            b9  =  B5[0];
1963            b10 =  B6[0];
1964            b11 =  B7[0];
1965            }
1966            par
1967            {
1968            b12  =  B8[1];
1969            b13  =  B9[1];
1970            b14  =  B10[1];
1971            b15  =  B11[1];
1972            b16  =  B0[1];
1973            b17  =  B1[1];
1974            b18  =  B2[1];
1975            b19  =  B3[1];
1976            b20  =  B4[1];
```

```
1977          b21  =  B5[1];
1978          b22  =  B6[1];
1979          b23  =  B7[1];
1980          }
1981          par
1982          {
1983          b24  =  B8[2];
1984          b25  =  B9[2];
1985          b26  =  B10[2];
1986          b27  =  B11[2];
1987          b28  =  B0[2];
1988          b29  =  B1[2];
1989          b30  =  B2[2];
1990          b31  =  B3[2];
1991          b32  =  B4[2];
1992          b33  =  B5[2];
1993          b34  =  B6[2];
1994          b35  =  B7[2];
1995          }
1996          par
1997          {
1998          b36  =  B8[3];
1999          b37  =  B9[3];
2000          b38  =  B10[3];
2001          b39  =  B11[3];
2002          b40  =  B0[3];
2003          b41  =  B1[3];
2004          b42  =  B2[3];
2005          b43  =  B3[3];
2006          b44  =  B4[3];
2007          b45  =  B5[3];
2008          b46  =  B6[3];
2009          b47  =  B7[3];
```

```
2010          }
2011          par
2012          {
2013          b48  =  B8[4];
2014          b49  =  B9[4];
2015          b50  =  B10[4];
2016          b51  =  B11[4];
2017          b52  =  B0[4];
2018          b53  =  B1[4];
2019          b54  =  B2[4];
2020          b55  =  B3[4];
2021          b56  =  B4[4];
2022          b57  =  B5[4];
2023          b58  =  B6[4];
2024          b59  =  B7[4];
2025          }
2026          par
2027          {
2028          b60  =  B8[5];
2029          b61  =  B9[5];
2030          b62  =  B10[5];
2031          b63  =  B11[5];
2032          b64  =  B0[5];
2033          b65  =  B1[5];
2034          b66  =  B2[5];
2035          b67  =  B3[5];
2036          b68  =  B4[5];
2037          b69  =  B5[5];
2038          b70  =  B6[5];
2039          b71  =  B7[5];
2040          }
2041          par
2042          {
```

```
2043            b72  =  B8[6];
2044            b73  =  B9[6];
2045            b74  =  B10[6];
2046            b75  =  B11[6];
2047            b76  =  B0[6];
2048            b77  =  B1[6];
2049            b78  =  B2[6];
2050            b79  =  B3[6];
2051            b80  =  B4[6];
2052            b81  =  B5[6];
2053            b82  =  B6[6];
2054            b83  =  B7[6];
2055            }
2056            par
2057            {
2058            b84  =  B8[7];
2059            b85  =  B9[7];
2060            b86  =  B10[7];
2061            b87  =  B11[7];
2062            b88  =  B0[7];
2063            b89  =  B1[7];
2064            b90  =  B2[7];
2065            b91  =  B3[7];
2066            b92  =  B4[7];
2067            b93  =  B5[7];
2068            b94  =  B6[7];
2069            b95  =  B7[7];
2070            }
2071            par
2072            {
2073            b96  =  B8[8];
2074            b97  =  B9[8];
2075            b98  =  B10[8];
```

```
2076          b99  =  B11[8];
2077          b100 =  B0[8];
2078          b101 =  B1[8];
2079          b102 =  B2[8];
2080          b103 =  B3[8];
2081          b104 =  B4[8];
2082          b105 =  B5[8];
2083          b106 =  B6[8];
2084          b107 =  B7[8];
2085          }
2086          par
2087          {
2088          b108  =  B8[9];
2089          b109  =  B9[9];
2090          b110  =  B10[9];
2091          b111  =  B11[9];
2092          b112  =  B0[9];
2093          b113  =  B1[9];
2094          b114  =  B2[9];
2095          b115  =  B3[9];
2096          b116  =  B4[9];
2097          b117  =  B5[9];
2098          b118  =  B6[9];
2099          b119  =  B7[9];
2100          }
2101          par
2102          {
2103          b120  =  B8[10];
2104          b121  =  B9[10];
2105          b122  =  B10[10];
2106          b123  =  B11[10];
2107          b124  =  B0[10];
2108          b125  =  B1[10];
```

```
2109            b126  =  B2[10];
2110            b127  =  B3[10];
2111            b128  =  B4[10];
2112            b129  =  B5[10];
2113            b130  =  B6[10];
2114            b131  =  B7[10];
2115            }
2116            par
2117            {
2118            b132  =  B8[11];
2119            b133  =  B9[11];
2120            b134  =  B10[11];
2121            b135  =  B11[11];
2122            b136  =  B0[11];
2123            b137  =  B1[11];
2124            b138  =  B2[11];
2125            b139  =  B3[11];
2126            b140  =  B4[11];
2127            b141  =  B5[11];
2128            b142  =  B6[11];
2129            b143  =  B7[11];
2130            }
2131        }
2132        //data tilordning for 10. syklus
2133        if (j == 9)
2134        {
2135            par
2136            {
2137            b0  =  B9[0];
2138            b1  =  B10[0];
2139            b2  =  B11[0];
2140            b3  =  B0[0];
2141            b4  =  B1[0];
```

```
2142          b5  =  B2[0];
2143          b6  =  B3[0];
2144          b7  =  B4[0];
2145          b8  =  B5[0];
2146          b9  =  B6[0];
2147          b10 =  B7[0];
2148          b11 =  B8[0];
2149          }
2150          par
2151          {
2152          b12  =  B9[1];
2153          b13  =  B10[1];
2154          b14  =  B11[1];
2155          b15  =  B0[1];
2156          b16  =  B1[1];
2157          b17  =  B2[1];
2158          b18  =  B3[1];
2159          b19  =  B4[1];
2160          b20  =  B5[1];
2161          b21  =  B6[1];
2162          b22  =  B7[1];
2163          b23  =  B8[1];
2164          }
2165          par
2166          {
2167          b24  =  B9[2];
2168          b25  =  B10[2];
2169          b26  =  B11[2];
2170          b27  =  B0[2];
2171          b28  =  B1[2];
2172          b29  =  B2[2];
2173          b30  =  B3[2];
2174          b31  =  B4[2];
```

```
2175          b32  =  B5[2];
2176          b33  =  B6[2];
2177          b34  =  B7[2];
2178          b35  =  B8[2];
2179          }
2180          par
2181          {
2182          b36  =  B9[3];
2183          b37  =  B10[3];
2184          b38  =  B11[3];
2185          b39  =  B0[3];
2186          b40  =  B1[3];
2187          b41  =  B2[3];
2188          b42  =  B3[3];
2189          b43  =  B4[3];
2190          b44  =  B5[3];
2191          b45  =  B6[3];
2192          b46  =  B7[3];
2193          b47  =  B8[3];
2194          }
2195          par
2196          {
2197          b48  =  B9[4];
2198          b49  =  B10[4];
2199          b50  =  B11[4];
2200          b51  =  B0[4];
2201          b52  =  B1[4];
2202          b53  =  B2[4];
2203          b54  =  B3[4];
2204          b55  =  B4[4];
2205          b56  =  B5[4];
2206          b57  =  B6[4];
2207          b58  =  B7[4];
```

```
2208          b59  =  B8[4];
2209          }
2210          par
2211          {
2212          b60  =  B9[5];
2213          b61  =  B10[5];
2214          b62  =  B11[5];
2215          b63  =  B0[5];
2216          b64  =  B1[5];
2217          b65  =  B2[5];
2218          b66  =  B3[5];
2219          b67  =  B4[5];
2220          b68  =  B5[5];
2221          b69  =  B6[5];
2222          b70  =  B7[5];
2223          b71  =  B8[5];
2224          }
2225          par
2226          {
2227          b72  =  B9[6];
2228          b73  =  B10[6];
2229          b74  =  B11[6];
2230          b75  =  B0[6];
2231          b76  =  B1[6];
2232          b77  =  B2[6];
2233          b78  =  B3[6];
2234          b79  =  B4[6];
2235          b80  =  B5[6];
2236          b81  =  B6[6];
2237          b82  =  B7[6];
2238          b83  =  B8[6];
2239          }
2240          par
```

```
2241          {
2242          b84  =  B9[7];
2243          b85  =  B10[7];
2244          b86  =  B11[7];
2245          b87  =  B0[7];
2246          b88  =  B1[7];
2247          b89  =  B2[7];
2248          b90  =  B3[7];
2249          b91  =  B4[7];
2250          b92  =  B5[7];
2251          b93  =  B6[7];
2252          b94  =  B7[7];
2253          b95  =  B8[7];
2254          }
2255          par
2256          {
2257          b96  =  B9[8];
2258          b97  =  B10[8];
2259          b98  =  B11[8];
2260          b99  =  B0[8];
2261          b100 =  B1[8];
2262          b101 =  B2[8];
2263          b102 =  B3[8];
2264          b103 =  B4[8];
2265          b104 =  B5[8];
2266          b105 =  B6[8];
2267          b106 =  B7[8];
2268          b107 =  B8[8];
2269          }
2270          par
2271          {
2272          b108  =  B9[9];
2273          b109  =  B10[9];
```

```
2274          b110  =  B11[9];
2275          b111  =  B0[9];
2276          b112  =  B1[9];
2277          b113  =  B2[9];
2278          b114  =  B3[9];
2279          b115  =  B4[9];
2280          b116  =  B5[9];
2281          b117  =  B6[9];
2282          b118  =  B7[9];
2283          b119  =  B8[9];
2284          }
2285          par
2286          {
2287          b120  =  B9[10];
2288          b121  =  B10[10];
2289          b122  =  B11[10];
2290          b123  =  B0[10];
2291          b124  =  B1[10];
2292          b125  =  B2[10];
2293          b126  =  B3[10];
2294          b127  =  B4[10];
2295          b128  =  B5[10];
2296          b129  =  B6[10];
2297          b130  =  B7[10];
2298          b131  =  B8[10];
2299          }
2300          par
2301          {
2302          b132  =  B9[11];
2303          b133  =  B10[11];
2304          b134  =  B11[11];
2305          b135  =  B0[11];
2306          b136  =  B1[11];
```

```
2307            b137  =  B2[11];
2308            b138  =  B3[11];
2309            b139  =  B4[11];
2310            b140  =  B5[11];
2311            b141  =  B6[11];
2312            b142  =  B7[11];
2313            b143  =  B8[11];
2314            }
2315        }
2316        //data tilordning for 11. syklus
2317        if (j == 10)
2318        {
2319            par
2320            {
2321            b0  =  B10[0];
2322            b1  =  B11[0];
2323            b2  =  B0[0];
2324            b3  =  B1[0];
2325            b4  =  B2[0];
2326            b5  =  B3[0];
2327            b6  =  B4[0];
2328            b7  =  B5[0];
2329            b8  =  B6[0];
2330            b9  =  B7[0];
2331            b10 =  B8[0];
2332            b11 =  B9[0];
2333            }
2334            par
2335            {
2336            b12  =  B10[1];
2337            b13  =  B11[1];
2338            b14  =  B0[1];
2339            b15  =  B1[1];
```

```
2340            b16  =  B2[1];
2341            b17  =  B3[1];
2342            b18  =  B4[1];
2343            b19  =  B5[1];
2344            b20  =  B6[1];
2345            b21  =  B7[1];
2346            b22  =  B8[1];
2347            b23  =  B9[1];
2348            }
2349            par
2350            {
2351            b24  =  B10[2];
2352            b25  =  B11[2];
2353            b26  =  B0[2];
2354            b27  =  B1[2];
2355            b28  =  B2[2];
2356            b29  =  B3[2];
2357            b30  =  B4[2];
2358            b31  =  B5[2];
2359            b32  =  B6[2];
2360            b33  =  B7[2];
2361            b34  =  B8[2];
2362            b35  =  B9[2];
2363            }
2364            par
2365            {
2366            b36  =  B10[3];
2367            b37  =  B11[3];
2368            b38  =  B0[3];
2369            b39  =  B1[3];
2370            b40  =  B2[3];
2371            b41  =  B3[3];
2372            b42  =  B4[3];
```

```
2373            b43  =  B5[3];
2374            b44  =  B6[3];
2375            b45  =  B7[3];
2376            b46  =  B8[3];
2377            b47  =  B9[3];
2378            }
2379            par
2380            {
2381            b48  =  B10[4];
2382            b49  =  B11[4];
2383            b50  =  B0[4];
2384            b51  =  B1[4];
2385            b52  =  B2[4];
2386            b53  =  B3[4];
2387            b54  =  B4[4];
2388            b55  =  B5[4];
2389            b56  =  B6[4];
2390            b57  =  B7[4];
2391            b58  =  B8[4];
2392            b59  =  B9[4];
2393            }
2394            par
2395            {
2396            b60  =  B10[5];
2397            b61  =  B11[5];
2398            b62  =  B0[5];
2399            b63  =  B1[5];
2400            b64  =  B2[5];
2401            b65  =  B3[5];
2402            b66  =  B4[5];
2403            b67  =  B5[5];
2404            b68  =  B6[5];
2405            b69  =  B7[5];
```

```
2406          b70  =  B8[5];
2407          b71  =  B9[5];
2408          }
2409          par
2410          {
2411          b72  =  B10[6];
2412          b73  =  B11[6];
2413          b74  =  B0[6];
2414          b75  =  B1[6];
2415          b76  =  B2[6];
2416          b77  =  B3[6];
2417          b78  =  B4[6];
2418          b79  =  B5[6];
2419          b80  =  B6[6];
2420          b81  =  B7[6];
2421          b82  =  B8[6];
2422          b83  =  B9[6];
2423          }
2424          par
2425          {
2426          b84  =  B10[7];
2427          b85  =  B11[7];
2428          b86  =  B0[7];
2429          b87  =  B1[7];
2430          b88  =  B2[7];
2431          b89  =  B3[7];
2432          b90  =  B4[7];
2433          b91  =  B5[7];
2434          b92  =  B6[7];
2435          b93  =  B7[7];
2436          b94  =  B8[7];
2437          b95  =  B9[7];
2438          }
```

```
2439          par
2440          {
2441          b96  =  B10[8];
2442          b97  =  B11[8];
2443          b98  =  B0[8];
2444          b99  =  B1[8];
2445          b100 =  B2[8];
2446          b101 =  B3[8];
2447          b102 =  B4[8];
2448          b103 =  B5[8];
2449          b104 =  B6[8];
2450          b105 =  B7[8];
2451          b106 =  B8[8];
2452          b107 =  B9[8];
2453          }
2454          par
2455          {
2456          b108  =  B10[9];
2457          b109  =  B11[9];
2458          b110  =  B0[9];
2459          b111  =  B1[9];
2460          b112  =  B2[9];
2461          b113  =  B3[9];
2462          b114  =  B4[9];
2463          b115  =  B5[9];
2464          b116  =  B6[9];
2465          b117  =  B7[9];
2466          b118  =  B8[9];
2467          b119  =  B9[9];
2468          }
2469          par
2470          {
2471          b120  =  B10[10];
```

```
2472            b121  =  B11[10];
2473            b122  =  B0[10];
2474            b123  =  B1[10];
2475            b124  =  B2[10];
2476            b125  =  B3[10];
2477            b126  =  B4[10];
2478            b127  =  B5[10];
2479            b128  =  B6[10];
2480            b129  =  B7[10];
2481            b130  =  B8[10];
2482            b131  =  B9[10];
2483            }
2484            par
2485            {
2486            b132  =  B10[11];
2487            b133  =  B11[11];
2488            b134  =  B0[11];
2489            b135  =  B1[11];
2490            b136  =  B2[11];
2491            b137  =  B3[11];
2492            b138  =  B4[11];
2493            b139  =  B5[11];
2494            b140  =  B6[11];
2495            b141  =  B7[11];
2496            b142  =  B8[11];
2497            b143  =  B9[11];
2498            }
2499        }
2500        //data tilordning for 12. syklus
2501        if (j == 11)
2502        {
2503            par
2504            {
```

```
2505          b0  =  B11[0];
2506          b1  =  B0[0];
2507          b2  =  B1[0];
2508          b3  =  B2[0];
2509          b4  =  B3[0];
2510          b5  =  B4[0];
2511          b6  =  B5[0];
2512          b7  =  B6[0];
2513          b8  =  B7[0];
2514          b9  =  B8[0];
2515          b10 =  B9[0];
2516          b11 =  B10[0];
2517          }
2518          par
2519          {
2520          b12  =  B11[1];
2521          b13  =  B0[1];
2522          b14  =  B1[1];
2523          b15  =  B2[1];
2524          b16  =  B3[1];
2525          b17  =  B4[1];
2526          b18  =  B5[1];
2527          b19  =  B6[1];
2528          b20  =  B7[1];
2529          b21  =  B8[1];
2530          b22  =  B9[1];
2531          b23  =  B10[1];
2532          }
2533          par
2534          {
2535          b24  =  B11[2];
2536          b25  =  B0[2];
2537          b26  =  B1[2];
```

```
2538            b27  =  B2[2];
2539            b28  =  B3[2];
2540            b29  =  B4[2];
2541            b30  =  B5[2];
2542            b31  =  B6[2];
2543            b32  =  B7[2];
2544            b33  =  B8[2];
2545            b34  =  B9[2];
2546            b35  =  B10[2];
2547            }
2548            par
2549            {
2550            b36  =  B11[3];
2551            b37  =  B0[3];
2552            b38  =  B1[3];
2553            b39  =  B2[3];
2554            b40  =  B3[3];
2555            b41  =  B4[3];
2556            b42  =  B5[3];
2557            b43  =  B6[3];
2558            b44  =  B7[3];
2559            b45  =  B8[3];
2560            b46  =  B9[3];
2561            b47  =  B10[3];
2562            }
2563            par
2564            {
2565            b48  =  B11[4];
2566            b49  =  B0[4];
2567            b50  =  B1[4];
2568            b51  =  B2[4];
2569            b52  =  B3[4];
2570            b53  =  B4[4];
```

```
2571            b54  =  B5[4];
2572            b55  =  B6[4];
2573            b56  =  B7[4];
2574            b57  =  B8[4];
2575            b58  =  B9[4];
2576            b59  =  B10[4];
2577            }
2578            par
2579            {
2580            b60  =  B11[5];
2581            b61  =  B0[5];
2582            b62  =  B1[5];
2583            b63  =  B2[5];
2584            b64  =  B3[5];
2585            b65  =  B4[5];
2586            b66  =  B5[5];
2587            b67  =  B6[5];
2588            b68  =  B7[5];
2589            b69  =  B8[5];
2590            b70  =  B9[5];
2591            b71  =  B10[5];
2592            }
2593            par
2594            {
2595            b72  =  B11[6];
2596            b73  =  B0[6];
2597            b74  =  B1[6];
2598            b75  =  B2[6];
2599            b76  =  B3[6];
2600            b77  =  B4[6];
2601            b78  =  B5[6];
2602            b79  =  B6[6];
2603            b80  =  B7[6];
```

```
2604          b81  =  B8[6];
2605          b82  =  B9[6];
2606          b83  =  B10[6];
2607          }
2608          par
2609          {
2610          b84  =  B11[7];
2611          b85  =  B0[7];
2612          b86  =  B1[7];
2613          b87  =  B2[7];
2614          b88  =  B3[7];
2615          b89  =  B4[7];
2616          b90  =  B5[7];
2617          b91  =  B6[7];
2618          b92  =  B7[7];
2619          b93  =  B8[7];
2620          b94  =  B9[7];
2621          b95  =  B10[7];
2622          }
2623          par
2624          {
2625          b96  =  B11[8];
2626          b97  =  B0[8];
2627          b98  =  B1[8];
2628          b99  =  B2[8];
2629          b100 =  B3[8];
2630          b101 =  B4[8];
2631          b102 =  B5[8];
2632          b103 =  B6[8];
2633          b104 =  B7[8];
2634          b105 =  B8[8];
2635          b106 =  B9[8];
2636          b107 =  B10[8];
```

```
2637          }
2638          par
2639          {
2640          b108  =  B11[9];
2641          b109  =  B0[9];
2642          b110  =  B1[9];
2643          b111  =  B2[9];
2644          b112  =  B3[9];
2645          b113  =  B4[9];
2646          b114  =  B5[9];
2647          b115  =  B6[9];
2648          b116  =  B7[9];
2649          b117  =  B8[9];
2650          b118  =  B9[9];
2651          b119  =  B10[9];
2652          }
2653          par
2654          {
2655          b120  =  B11[10];
2656          b121  =  B0[10];
2657          b122  =  B1[10];
2658          b123  =  B2[10];
2659          b124  =  B3[10];
2660          b125  =  B4[10];
2661          b126  =  B5[10];
2662          b127  =  B6[10];
2663          b128  =  B7[10];
2664          b129  =  B8[10];
2665          b130  =  B9[10];
2666          b131  =  B10[10];
2667          }
2668          par
2669          {
```

```
2670            b132  =  B11[11];
2671            b133  =  B0[11];
2672            b134  =  B1[11];
2673            b135  =  B2[11];
2674            b136  =  B3[11];
2675            b137  =  B4[11];
2676            b138  =  B5[11];
2677            b139  =  B6[11];
2678            b140  =  B7[11];
2679            b141  =  B8[11];
2680            b142  =  B9[11];
2681            b143  =  B10[11];
2682            }
2683        }
2684        }
2685        do
2686        {
2687        par//elementene multipliseres i parallell
2688        {
2689        xilinxmult0(q0,a0,b0);
2690        xilinxmult1(q1,a1,b1);
2691        xilinxmult2(q2,a2,b2);
2692        xilinxmult3(q3,a3,b3);
2693        xilinxmult4(q4,a4,b4);
2694        xilinxmult5(q5,a5,b5);
2695        xilinxmult6(q6,a6,b6);
2696        xilinxmult7(q7,a7,b7);
2697        xilinxmult8(q8,a8,b8);
2698        xilinxmult9(q9,a9,b9);
2699        xilinxmult10(q10,a10,b10);
2700        xilinxmult11(q11,a11,b11);
2701        xilinxmult12(q12,a12,b12);
2702        xilinxmult13(q13,a13,b13);
```

```
2703        xilinxmult14(q14,a14,b14);
2704        xilinxmult15(q15,a15,b15);
2705        xilinxmult16(q16,a16,b16);
2706        xilinxmult17(q17,a17,b17);
2707        xilinxmult18(q18,a18,b18);
2708        xilinxmult19(q19,a19,b19);
2709        xilinxmult20(q20,a20,b20);
2710        xilinxmult21(q21,a21,b21);
2711        xilinxmult22(q22,a22,b22);
2712        xilinxmult23(q23,a23,b23);
2713        xilinxmult24(q24,a24,b24);
2714        xilinxmult25(q25,a25,b25);
2715        xilinxmult26(q26,a26,b26);
2716        xilinxmult27(q27,a27,b27);
2717        xilinxmult28(q28,a28,b28);
2718        xilinxmult29(q29,a29,b29);
2719        xilinxmult30(q30,a30,b30);
2720        xilinxmult31(q31,a31,b31);
2721        xilinxmult32(q32,a32,b32);
2722        xilinxmult33(q33,a33,b33);
2723        xilinxmult34(q34,a34,b34);
2724        xilinxmult35(q35,a35,b35);
2725        xilinxmult36(q36,a36,b36);
2726        xilinxmult37(q37,a37,b37);
2727        xilinxmult38(q38,a38,b38);
2728        xilinxmult39(q39,a39,b39);
2729        xilinxmult40(q40,a40,b40);
2730        xilinxmult41(q41,a41,b41);
2731        xilinxmult42(q42,a42,b42);
2732        xilinxmult43(q43,a43,b43);
2733        xilinxmult44(q44,a44,b44);
2734        xilinxmult45(q45,a45,b45);
2735        xilinxmult46(q46,a46,b46);
```

```
2736        xilinxmult47(q47,a47,b47);
2737        xilinxmult48(q48,a48,b48);
2738        xilinxmult49(q49,a49,b49);
2739        xilinxmult50(q50,a50,b50);
2740        xilinxmult51(q51,a51,b51);
2741        xilinxmult52(q52,a52,b52);
2742        xilinxmult53(q53,a53,b53);
2743        xilinxmult54(q54,a54,b54);
2744        xilinxmult55(q55,a55,b55);
2745        xilinxmult56(q56,a56,b56);
2746        xilinxmult57(q57,a57,b57);
2747        xilinxmult58(q58,a58,b58);
2748        xilinxmult59(q59,a59,b59);
2749        xilinxmult60(q60,a60,b60);
2750        xilinxmult61(q61,a61,b61);
2751        xilinxmult62(q62,a62,b62);
2752        xilinxmult63(q63,a63,b63);
2753        xilinxmult64(q64,a64,b64);
2754        xilinxmult65(q65,a65,b65);
2755        xilinxmult66(q66,a66,b66);
2756        xilinxmult67(q67,a67,b67);
2757        xilinxmult68(q68,a68,b68);
2758        xilinxmult69(q69,a69,b69);
2759        xilinxmult70(q70,a70,b70);
2760        xilinxmult71(q71,a71,b71);
2761        xilinxmult72(q72,a72,b72);
2762        xilinxmult73(q73,a73,b73);
2763        xilinxmult74(q74,a74,b74);
2764        xilinxmult75(q75,a75,b75);
2765        xilinxmult76(q76,a76,b76);
2766        xilinxmult77(q77,a77,b77);
2767        xilinxmult78(q78,a78,b78);
2768        xilinxmult79(q79,a79,b79);
```

```
2769        xilinxmult80(q80,a80,b80);
2770        xilinxmult81(q81,a81,b81);
2771        xilinxmult82(q82,a82,b82);
2772        xilinxmult83(q83,a83,b83);
2773        xilinxmult84(q84,a84,b84);
2774        xilinxmult85(q85,a85,b85);
2775        xilinxmult86(q86,a86,b86);
2776        xilinxmult87(q87,a87,b87);
2777        xilinxmult88(q88,a88,b88);
2778        xilinxmult89(q89,a89,b89);
2779        xilinxmult90(q90,a90,b90);
2780        xilinxmult91(q91,a91,b91);
2781        xilinxmult92(q92,a92,b92);
2782        xilinxmult93(q93,a93,b93);
2783        xilinxmult94(q94,a94,b94);
2784        xilinxmult95(q95,a95,b95);
2785        xilinxmult96(q96,a96,b96);
2786        xilinxmult97(q97,a97,b97);
2787        xilinxmult98(q98,a98,b98);
2788        xilinxmult99(q99,a99,b99);
2789        xilinxmult100(q100,a100,b100);
2790        xilinxmult101(q101,a101,b101);
2791        xilinxmult102(q102,a102,b102);
2792        xilinxmult103(q103,a103,b103);
2793        xilinxmult104(q104,a104,b104);
2794        xilinxmult105(q105,a105,b105);
2795        xilinxmult106(q106,a106,b106);
2796        xilinxmult107(q107,a107,b107);
2797        xilinxmult108(q108,a108,b108);
2798        xilinxmult109(q109,a109,b109);
2799        xilinxmult110(q110,a110,b110);
2800        xilinxmult111(q111,a111,b111);
2801        xilinxmult112(q112,a112,b112);
```

```
2802        xilinxmult113(q113,a113,b113);
2803        xilinxmult114(q114,a114,b114);
2804        xilinxmult115(q115,a115,b115);
2805        xilinxmult116(q116,a116,b116);
2806        xilinxmult117(q117,a117,b117);
2807        xilinxmult118(q118,a118,b118);
2808        xilinxmult119(q119,a119,b119);
2809        xilinxmult120(q120,a120,b120);
2810        xilinxmult121(q121,a121,b121);
2811        xilinxmult122(q122,a122,b122);
2812        xilinxmult123(q123,a123,b123);
2813        xilinxmult124(q124,a124,b124);
2814        xilinxmult125(q125,a125,b125);
2815        xilinxmult126(q126,a126,b126);
2816        xilinxmult127(q127,a127,b127);
2817        xilinxmult128(q128,a128,b128);
2818        xilinxmult129(q129,a129,b129);
2819        xilinxmult130(q130,a130,b130);
2820        xilinxmult131(q131,a131,b131);
2821        xilinxmult132(q132,a132,b132);
2822        xilinxmult133(q133,a133,b133);
2823        xilinxmult134(q134,a134,b134);
2824        xilinxmult135(q135,a135,b135);
2825        xilinxmult136(q136,a136,b136);
2826        xilinxmult137(q137,a137,b137);
2827        xilinxmult138(q138,a138,b138);
2828        xilinxmult139(q139,a139,b139);
2829        xilinxmult140(q140,a140,b140);
2830        xilinxmult141(q141,a141,b141);
2831        xilinxmult142(q142,a142,b142);
2832        xilinxmult143(q143,a143,b143);
2833        }
2834        //delproduktene adderes i fire omganger
```

```
2835      par
2836      {
2837      s0  = q0+q1;
2838      s1  = q2+q3;
2839      s2  = q4+q5;
2840      s3  = q6+q7;
2841      s4  = q8+q9;
2842      s5  = q10+q11;
2843      s6  = q12+q13;
2844      s7  = q14+q15;
2845      s8  = q16+q17;
2846      s9  = q18+q19;
2847      s10 = q20+q21;
2848      s11  = q22+q23;
2849      s12  = q24+q25;
2850      s13  = q26+q27;
2851      s14  = q28+q29;
2852      s15  = q30+q31;
2853      s16  = q32+q33;
2854      s17  = q34+q35;
2855      s18  = q36+q37;
2856      s19  = q38+q39;
2857      s20  = q40+q41;
2858      s21  = q42+q43;
2859      s22  = q44+q45;
2860      s23  = q46+q47;
2861      s24  = q48+q49;
2862      s25  = q50+q51;
2863      s26  = q52+q53;
2864      s27  = q54+q55;
2865      s28  = q56+q57;
2866      s29  = q58+q59;
2867      s30  = q60+q61;
```

```
2868        s31  = q62+q63;
2869        s32  = q64+q65;
2870        s33  = q66+q67;
2871        s34  = q68+q69;
2872        s35  = q70+q71;
2873        s36  = q72+q73;
2874        s37  = q74+q75;
2875        s38  = q76+q77;
2876        s39  = q78+q79;
2877        s40  = q80+q81;
2878        s41  = q82+q83;
2879        s42  = q84+q85;
2880        s43  = q86+q87;
2881        s44  = q88+q89;
2882        s45  = q90+q91;
2883        s46  = q92+q93;
2884        s47  = q94+q95;
2885        s48  = q96+q97;
2886        s49  = q98+q99;
2887        s50  = q100+q101;
2888        s51  = q102+q103;
2889        s52  = q104+q105;
2890        s53  = q106+q107;
2891        s54  = q108+q109;
2892        s55  = q110+q111;
2893        s56  = q112+q113;
2894        s57  = q114+q115;
2895        s58  = q116+q117;
2896        s59  = q118+q119;
2897        s60  = q120+q121;
2898        s61  = q122+q123;
2899        s62  = q124+q125;
2900        s63  = q126+q127;
```

```
2901      s64  = q128+q129;
2902      s65  = q130+q131;
2903      s66  = q132+q133;
2904      s67  = q134+q135;
2905      s68  = q136+q137;
2906      s69  = q138+q139;
2907      s70  = q140+q141;
2908      s71  = q142+q143;
2909      }
2910      par
2911      {
2912      t0 = s0+s1;
2913      t1 = s2+s3;
2914      t2 = s4+s5;
2915      t3 = s6+s7;
2916      t4 = s8+s9;
2917      t5 = s10+s11;
2918      t6 = s12+s13;
2919      t7 = s14+s15;
2920      t8 = s16+s17;
2921      t9 = s18+s19;
2922      t10 = s20+s11;
2923      t11 = s22+s23;
2924      t12 = s24+s25;
2925      t13 = s26+s27;
2926      t14 = s28+s29;
2927      t15 = s30+s31;
2928      t16 = s32+s33;
2929      t17 = s34+s35;
2930      t18 = s36+s37;
2931      t19 = s38+s39;
2932      t20 = s40+s41;
2933      t21 = s42+s43;
```

```
2934      t22 = s44+s45;
2935      t23 = s46+s47;
2936      t24 = s48+s49;
2937      t25 = s50+s51;
2938      t26 = s52+s53;
2939      t27 = s54+s55;
2940      t28 = s56+s57;
2941      t29 = s58+s59;
2942      t30 = s60+s61;
2943      t31 = s62+s63;
2944      t32 = s64+s65;
2945      t33 = s66+s67;
2946      t34 = s68+s69;
2947      t35 = s70+s71;
2948      }
2949      par
2950      {
2951      u0 = t0+t1;
2952      u1 = t3+t4;
2953      u2 = t6+t7;
2954      u3 = t9+t10;
2955      u4 = t12+t13;
2956      u5 = t15+t16;
2957      u6 = t18+t19;
2958      u7 = t21+t22;
2959      u8 = t24+t25;
2960      u9 = t27+t28;
2961      u10 = t30+t31;
2962      u11 = t33+t34;
2963      }
2964      par
2965      {
2966      p0 = u0+t2;
```

```
2967      p1 = u1+t5;
2968      p2 = u2+t8;
2969      p3 = u3+t11;
2970      p4 = u4+t14;
2971      p5 = u5+t17;
2972      p6 = u6+t20;
2973      p7 = u7+t23;
2974      p8 = u8+t26;
2975      p9 = u9+t29;
2976      p10 = u10+t32;
2977      p11 = u11+t35;
2978      }
2979      //til elementer i produktmatrisen R
2980      //som lagres
2981      par
2982      {
2983      R0[j]  = p0;
2984      R1[j]  = p1;
2985      R2[j]  = p2;
2986      R3[j]  = p3;
2987      R4[j]  = p4;
2988      R5[j]  = p5;
2989      R6[j]  = p6;
2990      R7[j]  = p7;
2991      R8[j]  = p8;
2992      R9[j]  = p9;
2993      R10[j] = p10;
2994      R11[j] = p11;
2995      }
2996      //når j=11 er R beregnet
2997      if (j==11)
2998      {
2999      ferdig = 1;
```

```
3000        }
3001        j=j+1;
3002        //reset av hjelpevariabler
3003        par
3004        {
3005        q0=0;
3006        q1=0;
3007        q2=0;
3008        q3=0;
3009        q4=0;
3010        q5=0;
3011        q6=0;
3012        q7=0;
3013        q8=0;
3014        q9=0;
3015        q10=0;
3016        q11=0;
3017        q12=0;
3018        q13=0;
3019        q14=0;
3020        q15=0;
3021        q16=0;
3022        q17=0;
3023        q18=0;
3024        q19=0;
3025        q20=0;
3026        q21=0;
3027        q22=0;
3028        q23=0;
3029        q24=0;
3030        q25=0;
3031        q26=0;
3032        q27=0;
```

```
3033        q28=0;
3034        q29=0;
3035        q30=0;
3036        q31=0;
3037        q32=0;
3038        q33=0;
3039        q34=0;
3040        q35=0;
3041        q36=0;
3042        q37=0;
3043        q38=0;
3044        q39=0;
3045        q40=0;
3046        q41=0;
3047        q42=0;
3048        q43=0;
3049        q44=0;
3050        q45=0;
3051        q46=0;
3052        q47=0;
3053        q48=0;
3054        q49=0;
3055        q50=0;
3056        q51=0;
3057        q52=0;
3058        q53=0;
3059        q54=0;
3060        q55=0;
3061        q56=0;
3062        q57=0;
3063        q58=0;
3064        q59=0;
3065        q60=0;
```

```
3066      q61=0;
3067      q62=0;
3068      q63=0;
3069      q64=0;
3070      q65=0;
3071      q66=0;
3072      q67=0;
3073      q68=0;
3074      q69=0;
3075      q70=0;
3076      q71=0;
3077      q72=0;
3078      q73=0;
3079      q74=0;
3080      q75=0;
3081      q76=0;
3082      q77=0;
3083      q78=0;
3084      q79=0;
3085      q80=0;
3086      q81=0;
3087      q82=0;
3088      q83=0;
3089      q84=0;
3090      q85=0;
3091      q86=0;
3092      q87=0;
3093      q88=0;
3094      q89=0;
3095      q90=0;
3096      q91=0;
3097      q92=0;
3098      q93=0;
```

```
3099        q94=0;
3100        q95=0;
3101        q96=0;
3102        q97=0;
3103        q98=0;
3104        q99=0;
3105        q100=0;
3106        q101=0;
3107        q102=0;
3108        q103=0;
3109        q104=0;
3110        q105=0;
3111        q106=0;
3112        q107=0;
3113        q108=0;
3114        q109=0;
3115        q110=0;
3116        q111=0;
3117        q112=0;
3118        q113=0;
3119        q114=0;
3120        q115=0;
3121        q116=0;
3122        q117=0;
3123        q118=0;
3124        q119=0;
3125        q120=0;
3126        q121=0;
3127        q122=0;
3128        q123=0;
3129        q124=0;
3130        q125=0;
3131        q126=0;
```

```
3132        q127=0;
3133        q128=0;
3134        q129=0;
3135        q130=0;
3136        q131=0;
3137        q132=0;
3138        q133=0;
3139        q134=0;
3140        q135=0;
3141        q136=0;
3142        q137=0;
3143        q138=0;
3144        q139=0;
3145        q140=0;
3146        q141=0;
3147        q142=0;
3148        q143=0;
3149        p0=0;
3150        p1=0;
3151        p2=0;
3152        p3=0;
3153        p4=0;
3154        p5=0;
3155        p6=0;
3156        p7=0;
3157        p8=0;
3158        p9=0;
3159        p10=0;
3160        p11=0;
3161        }
3162        }
3163        while (j<=11);
3164
```

```
3165  }
```