

Prosjekt i
SIF8064 Datamaskinarkitektur
-
Computational RAM

Rune M. Andersen

3. april 2003

Forord

Denne rapporten er skrevet som del av obligatorisk øvingsopplegg i faget SIF8064 - Datamaskinarkitektur ved NTNU, våren 2003. Rapporten beskriver 'Computational RAM'/CRAM (minne med integrert beregningskraft), som er et forsøk på å øke stormaskiners ytelse ved å utnytte prinsipper om parallell beregning nært lageret.

Rapporten beskriver oppbygning og virkemåte for CRAM, dets egenskaper og begrensninger, og hvordan det skiller seg fra konvensjonell RAM. Tanker rundt prosjektets gjennomførelse og relevans blir drøftet i slutten av rapporten.

Rune M. Andersen

Innhold

1	Sammendrag	1
2	Gjennomføring	3
2.1	Forundersøkelser	3
2.2	Språklige konvensjoner	3
2.3	Egenskaper ved CRAM	4
2.3.1	Overordnet arkitektur	4
2.3.2	Båndbredde	6
2.3.3	Strømforbruk	6
2.4	Applikasjonsområder	7
2.5	CRAM versus konvensjonell RAM	7
2.6	Ulemper	7
2.7	Annen info rundt CRAM	8
2.7.1	Bruk i dag	8
2.7.2	Oppmerksomhet	8
3	Læring	9
4	Kobling til pensum	11
5	Forslag til øvingsoppgave	13
6	Prosjektevaluering	15
7	Konklusjon	17
8	Kildehenvisninger	19

Kapittel 1

Sammendrag

Mannen bak CRAM heter Duncan Elliott og jobber som professor ved University of Alberta, Canada. CRAM er egentlig vanlig RAM (DRAM) hvor man har plassert enbitsprosessorer på modulen sammen med selve minnet. På denne måten kan man utføre parallelle beregninger i svært høye hastigheter internt på modulen, uten å flytte data over bussen. Foruten ytelsesforbedringen sparer man også strøm.

En fungerende prototyp er utviklet og teknologien er rettet spesielt mot vektoroperasjoner og bildebehandling. Elliott har også publisert en artikkel på hvordan man kan oppnå 1 petaOp/s ved hjelp av CRAM.

Kapittel 2

Gjennomføring

2.1 Forundersøkelser

Foruten Elliotts artikkel [Elliott1992] har jeg gjort søk på internett og funnet en del interessante artikler, forelesninger og nettsider. Et søk etter frasen 'computational ram' gav meg i underkant av 300 treff, men mange av disse (rundt 80%) var enten lenker til Elliotts hjemmeside, artikkelen hans, eller ordbokforklaringer av begrepet 'computational ram'.

Et søk på BibSys gav ingen relevante treff, så informasjonene måtte for det meste hentes fra nettet. Hovedvekten er derfor lagt på artiklene som lå på Elliotts hjemmeside, sammen med de mest interessante søketreffene.

Jeg har sendt en e-post til Elliott i tilfelle han hadde artikler eller liknende som kunne være av interesse for meg, men jeg har så langt ikke mottatt noe svar.

2.2 Språklige konvensjoner

På grunn av forskjellig bruk av uttrykk og begreper i de engelske kildene, er det i denne rapporten forsøkt å benytte et så konsistent språk som mulig. Følgende uttrykk er brukt:

- *Prosessorene* henviser til SIMD-prosessorene finnes på CRAM-modulen.
- *Modul (minnemodul)* benyttes om det man i dagligtalen litt feilaktig kaller *minnebrikke*, men som egentlig betyr printkortet med tilhørende brikker og kontakter (pinner).
- *Minnebrikke* benyttes om selve brikkekretsene som minnet består av, altså den pakningen transistorene er kapslet inn i.

- *Parallelinstruksjoner* er de 8-bits instruksjonene man sender til SIMD-prosessorene.
- *Buss/databuss* er bussen som knytter minnet og CPU på vertsmaskinen sammen. Hvor andre busser (for eksempel resultatbussen på CRAM-modulen) er nevnt, vil det fremgå av sammenhengen.
- *Naturlig parallellitet* benyttes om oppgaver som egner seg godt for parallellisering.

2.3 Egenskaper ved CRAM

2.3.1 Overordnet arkitektur

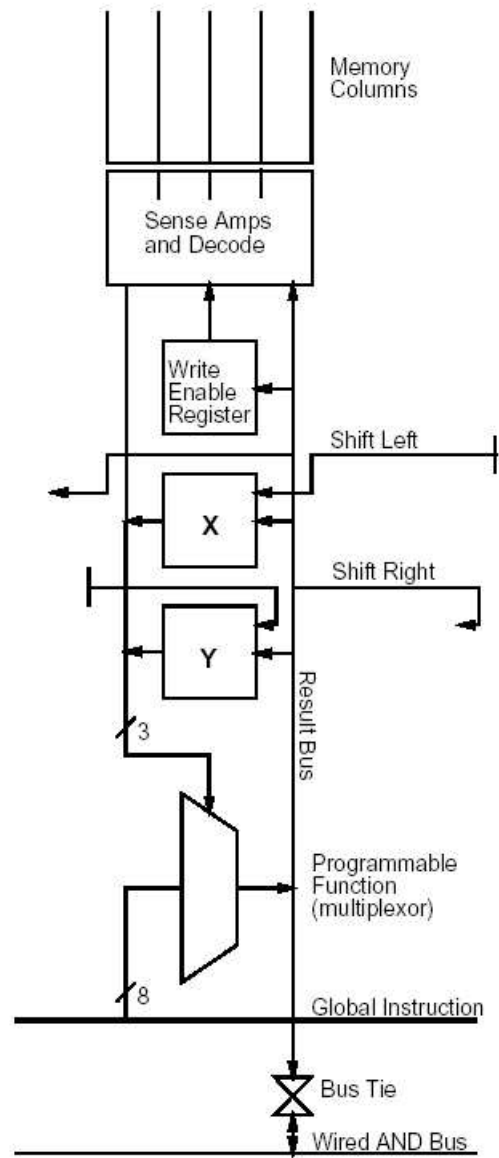
CRAM bygger på konvensjonell RAM, hvor man har plassert SIMD (single instruction, multiple data) prosessorer på modulen sammen med minnekretsene. SIMD betyr at prosessorene utfører like operasjoner på mange datasett samtidig, for eksempel pikselvise bildeoperasjoner. Elliott [ElliottPres] setter opp en del punkter for å begrunne valget av SIMD-prosessorer. Under er noen av dem gjengitt.

- *Ytelse*
Høy ytelse i forhold til størrelse og pris.
- *Lange ord*
Man utnytter båndbredden bedre fordi man ikke bruker bare deler av ordene.
- *Arkitektur*
Man ønsker å beholde eksisterende minnearkitektur ikke legge til overhead.

Slik konvensjonelt minne er oppbygd lagres dataene i rader og kolonner av transistorer. Ladningene er svært små og ville forsvunnet uten ekstra forsterkning ved utlesning. Forsterkningen skjer i såkalte 'sense amplifiers' og sendes deretter ut på modulens pinner. [YehWWW]. I CRAM er 'sense amplifler'-ene satt sammen med hver sin lille prosessor. Prosessorene er bit-serielle fordi man ønsket å bruke så lite areal som mulig på minnemodulene, og består i hovedsak av to registre (X og Y) og en multiplexer som fungerer som ALU (arithmetic logic unit).

For illustrasjon se figur 2.1 (side 5). Figuren er hentet fra [Elliott1992].

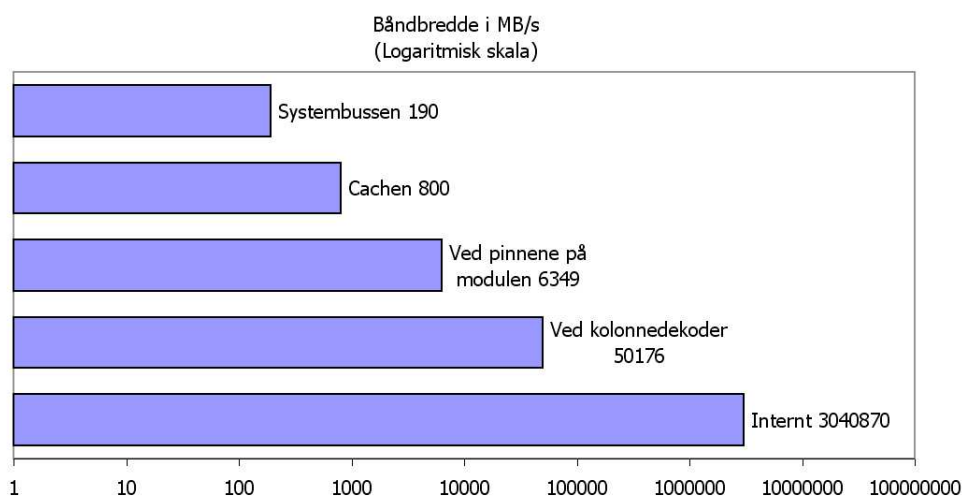
Man leser inn en bit fra minnet (fra 'sense amplifler'-en) og en fra hvert av registrene X og Y. Instruksjonene er 8-bit og blir multiplexet over adresselinjene. Etter at instruksjonen er utført legges resultatet på en resultatbuss og kan skrives tilbake til minnet og/eller X- og Y-registrene. Kommunikasjon mellom forskjellige prosessorer skjer ved å skifte data mellom X- og Y-registrene, til høyre på det ene og til venstre på det andre.



Figur 2.1: Prinsippskisse av arkitektur

2.3.2 Båndbredde

En hovedmotivasjon for flytte beregningskraften over i minnet og ikke motsatt, er den høye båndbredden internt i minnet. Mens man er begrenset til i underkant av 1 GB/s på overføringer mellom cache og prosessor, kan man på minnemodulen utnytte båndbredde i underkant av 3 TB/s, det vil si ca. 3000 ganger raskere. For en sammenlikning av båndbredde se figur 2.2 (side 6). Merk at søylene er fremstilt i logaritmisk skala. Figuren er en nytegning av figuren fra [ElliottWWW].



Figur 2.2: Sammenlikning av båndbredde

2.3.3 Strømforbruk

Elliott beskriver i sin artikkel [Elliott1995] hvordan man kan oppnå hastigheter rundt en petaop/s i CRAM. Han beskriver også strømforbruket man ville fått ved å bruke konvensjonelt minne på disse hastighetene. Fordi man med CRAM kan begrense mengden data man sender vekk fra modulen, kan man redusere strømforbruket samtidig som man øker hastigheten. De korte linjene internt på modulen krever mindre strøm for å drives enn man trenger for å drive signaler fra modulen og over på databussen.

Dersom denne type minne ble vanlig i vanlige konsumentprodukter, som for eksempel bærbare PCer, kunne man økt batteritiden vesentlig. Normalt sett ville virkningen vær motsatt; høyere hastigheter, høyere strømforbruk.

2.4 Applikasjonsområder

For å utnytte den CRAM best mulig er det viktig å finne problemer som har en tilstrekkelig grad av innebygd parallellitet, det vil si problemer som består av mange delproblemer som er mulig å løse uavhengig av hverandre. Matriseoperasjoner og bildebehandling er eksempler på slike problemer. Operasjoner på bilder gjøres ofte som pikseloperasjoner, der et resultat først skal beregnes for hvert enkelt piksel i bildet før man kan gjøre den samme runden på nytt. Å gjøre disse operasjonene parallellt ville kunne redusert tiden med en faktor lik antall piksler i bildet (opp til en grense hvor man begynner å gå i metning i forhold til antall prosessorer på minnemodulen). Gitt et bilde på 256x256 piksler, ville man kunne oppnå en maksimal hastighetsøkning av størrelse 65536 ganger. Dette tallet er naturligvis ikke realistisk på grunn av overhead ved å administrere operasjonene, men selv om man bare kunne høste en liten del av forbedringen ville man ha oppnådd svært mye.

Matriseoperasjoner, for eksempel å løse likningssett eller gjøre numeriske operasjoner, er også godt egnet for CRAM. Numeriske metoder fungerer på samme måte som med pikseloperasjoner. Man løser alle likningene for et gitt sett med verdier, før man bruker løsningen som inndata til neste likningssett.

2.5 CRAM versus konvensjonell RAM

CRAM er i utgangspunktet konvensjonell RAM med prosessorer i tillegg til minnebrikkene, og CRAM-moduler ville derfor kunne brukes som helt vanlige minnemoduler. Prosessorene krever ekstra plass på modulen, men i forhold til ytelsesforbedringen er dette noe man har råd til å leve med. I prototypen Elliott har utviklet bruker prosessorene 9% av det totale arealet på kretsen, men han påpeker at i moduler produsert på samlebånd ville selve minnet kreve mindre av plassen [Elliott1992].

Bruk av pinner på minnemodulen begrenses ved å multiplexe parallellinstruksjonene på adresselinjene i gitte klokkesykler. De resterende syklene fungerer modulen som vanlig.

2.6 Ulemper

Selv om fordelene med å implementere prosessorer i minnet er store, bør ulepene nevnes. Under følger noen punkter.

- *Areal*
Beregningsdelen vil gi ekstra komponenter på minnemodulen og vil ta opp noe av plassen. Mens prototypen bruker 9%, kan designet forbedres til ikke å bruke mer enn 3%.

- *Kostnader*
Kostnadene er anslått å øke med omtrent samme faktor som arealet, hvilket betyr at prisen vil være overkommelig.
- *Hurtighet*
Vanlige minneoperasjonene vil ta lengre tid fordi man multiplexer minneadresser og parallellinstruksjoner på samme buss. (Paralleloperasjonene vil imidlertid gå svært mye raskere).
- *Kompleksitet*
Samtidig som man øker antallet komponenter på minnemodulen, øker man også kompleksiteten, og introduserer nye feilkilder i et allerede godt utprøvd design.

2.7 Annen info rundt CRAM

2.7.1 Bruk i dag

The Varsity Online [VarsityWWW] meldte i 1997 at brikkeprodusenten Accelerix hadde lisensiert Elliotts CRAM-teknologi. En grafikkontroller kalt PhantASM med CRAM skulle være i produksjon og man ventet at den ville komme på markedet i 1998. Så vidt jeg kan se av søk på nettet er denne kontrolleren ikke spesielt utbredet, så jeg antar at den ikke ble den helt store suksessen, kanskje i skyggen av nVidias svært populære GeForce-brikker. Rundt årskiftet 1999/2000 ble forøvrig Accelerix kjøpt opp av Mosaid, som nå også eier patentene på CRAM.

2.7.2 Oppmerksomhet

I forbindelse med årets nominasjon til Douglas R. Colton Medal for Research Excellence (utdeles av Canadian Microelectronics Corporation, CMC), ble tidligere vinnere presentert. Der kunne man lese at prisen for året 2001 ble tildelt Douglas Elliott for hans arbeid rundt CRAM. Prispengene lød på 3.500 CAD, det vil si ca 17.000 NOK med dagens kurs.

Kapittel 3

Læring

Jeg har lært at man kan gå nye veier for å øke ytelsen i parallelle systemer. Metoden har vært prøvd tidligere, men man kan spørre seg hvorfor det ikke har blitt en suksess. Istedenfor å bringe minnet nærmere prosessoren har man altså flyttet prosessoren over på minnet. En enkel idé som kan øke ytelsen dramatisk.

Har også lært litt mer hva SIMD er og hvordan denne typen prosessorer kan utnyttes, og fått mer kunnskap om hvordan man kan bruke enkle prosessorer til å løse store oppgaver.

Minnearkitektur har også fått oppmerksomhet, og jeg vet mer om emnet nå enn jeg visste før jeg begynte med dette prosjektet.

Og sist men ikke minst har jeg lært hvordan CDRAM er bygget opp og fungerer.

Kapittel 4

Kobling til pensum

- *Parallellitet*
CRAM kan være en svært kraftig måte å gjøre parallelle beregninger, noe som er en rød tråd gjennom faget.
- *SIMD*
Hvordan SIMD-prosessorer virker og kan brukes i praksis. Spesielt hvilke fordeler disse har i forhold til å løse oppgaver med stor grad av innebygd parallellitet.
- *Minne*
Kommunikasjonsnett er en del av pensum, og i løpet av prosjektet har jeg sett hvordan man kan flytte problemet vekk fra bussen. I stedet for å flytte store mengder data mellom minne og CPU, kan man gjøre en stor del av beregningene inne på minnet og la bussen være fri til annet bruk.

Kapittel 5

Forslag til øvingsoppgave

Forslag til øvingsoppgave kunne være å forsøke å simulere programkjøringer i CRAM ved å lage en programparser. Man lager et program som inneholder helt enkle egendefinert instruksjoner. Deretter kjøres disse i parseren og tiden beregnes i diskret steg. Ved å bruke tråder får man en tilstrekkelig grad av parallellitet. For hver tråd samler man informasjon om hvor mye arbeid som er utført og hvor lang tid (hvor mange steg) arbeidet tok. Dette kan fremstilles grafisk for å enkelt se hvordan belastningen fordeler seg i systemet.

Å leke seg med hvordan forskjellige program fordeler seg ville illustrere at parallell-programmering ikke er trivielt selv med helt enkel kode, samtidig som man kunne forsøke å finne smarte løsninger for å fordele lasten bedre.

Kapittel 6

Prosjektevaluering

Å jobbe med et prosjekt istedenfor å ha de vanlige øvingene kan innebære både fordeler og ulemper. Man har større ansvar for sitt eget arbeid, men må mer enn før på egen hånd sørge for overholdelse av tidsfrister og progresjon i arbeidet. Slik studiet er i fjerde klasse, har man fire store fag, selv om hvert av dem bare er 2,5 vektall. De fleste av fagene (for min del alle) har obligatoriske prosjekter. Mange av disse spiller også inn på sluttkarakteren. Spesielt prosjektfaget 'Ekspertes i Team' krever sin del av semesteret, ettersom man her skal levere to rapporter, samt ferdigstille den faktiske oppgaven. Ønsker man å levere et godt resultat i noen fag må kanskje andre fag lide på grunn av tidsnød.

Den største fordelen er også den største ulempen: Stor grad av frihet. Uten faste milepæler underveis vil arbeidet utsettes til tidsfristen nærmer seg, noe de fleste studenter får føle når eksamen nærmer seg og man ikke har lest pensum enda.

Jeg kan ikke annet enn å gi meg selv kritikk for ikke å ha kommet skikkelig i gang med prosjektet tidligere, men noen fag måtte prioriteres før andre. I forbindelse med 'Ekspertes i Team' er jeg med på en kyb-landsby hvor vårt team skal være med på å lage en robot som skal delta i 'Eurobot 2003'-konkurransen i Frankrike i slutten av mai. Prosjektet tilhører Halvard Angelvik som skriver hovedoppgaven sin rundt roboten, og både Halvard og andre deltakerene på prosjektet forventer at alle gjør sitt beste for å få roboten ferdig til konkurransen starter. Dette prosjektet har derfor fått høyere prioritet enn mine andre fag ettersom vi skal forsvare NTNUs (og Norges..?) ære i Frankrike.

Kapittel 7

Konklusjon

CRAM virker som en svært lovende teknologi. Ettersom grensesnittet er det samme som med vanlig minne, skulle man tro at CRAM raskt kunne gjøre sitt inntog på grafikkort i de høyere prisklassene, hvor ytelsen ville vært suverent bedre enn det man finner i butikkene i dag.

Selv om teknologien kanskje først og fremst er ment for stormaskiner, mener jeg den har stort potensiale innen blant annet spillbransjen. Mange unge mennesker er villig til å betale like mye for et enkelt grafikkort som en komplett PC koster i andre butikker.

Så gjenstår det bare å la tiden vise om CRAM-teknologien er konkurransedyktig eller ikke.

Kapittel 8

Kildehenvisninger

- [Elliott1992] Duncan G. Elliott, W. Martin Snelgrove and Michael Stumm. *Computational Ram: A Memory-SIMD Hybrid and its Applicatoin to DSP*, 1992
- [ElliottWWW] <http://www.ee.ualberta.ca/~elliott/cram/> *Computational RAM*
- [Elliott1995] Duncan G. Elliott, W. Martin Snelgrove, Christian Cojocar, and Michael Stumm. *A PetaOp/s is Currently Feasible by Computing in RAM*, 1995
- [ElliottPres] Duncan G. Elliott, W. Martin Snelgrove, Christian Cojocar, and Michael Stumm. *A PetaOp/s is Currently Feasible by Computing in RAM, Presentation*, 1995
- [VarsityWWW] <http://www.varsity.utoronto.ca/archives/118/oct09/scitech/computers.html> *Varsity Science & Technology – Computer memory CRAMmed with power*
- [CMCWWW] http://www.cmc.ca/news/awards/colton_medal.html *Douglas R. Colton Medal for Research Excellence*
- [YehWWW] <http://prizedwriting.ucdavis.edu/past/1998-1999/yeh.html> *Scientific/Technical, Sinclair Yeh*
- [Elliott1997] Duncan Elliott, Michael Stumm and Martin Snelgrove. *Computational RAM: The case for SIMD computing in memory*
- [SMD099] Håkan Jonsson and Daniel Olsson *Computational RAM, Project in SMD099 Digital Structures on Silicon*