Gruppeopgave 2

Henrik Bendt (191191)
Jens Fredskov (240191)
Naja Wulff Mottelson (300988)

1. december 2011

1 Indledning

2 Den transistorbaserede computer

2.1 Situationen før i tiden

Den transistorbaserede computer kendes bedre blot som en computer, da almindelige computere i dag alle er transistorbaserede. Hvis man ser på fremtidsudsigterne for denne er flere faktorer afgørende.

Hvis man først ser på Gordon E. Moores artikel *Cramming more components onto integrated circuits* fra 1965, er synet på fremtiden rimelig positiv. Om forventningerne skrives bl.a.:

"The complexity for minimum component costs has increased at a rate of roughly a factor of two per year (see graph on next page). Certainly over the short term this rate can be expected to continue, if not to increase. Over the longer term, the rate of increase is a bit more uncertain, although there is no reason to believe it will not remain nearly constant for at least 10 years. That means by 1975, the number of components per integrated circuit for minimum cost will be 65,000. I believe that such a large circuit can be built on a single wafer." [1]

Her beskrives netop en forventning om, at man vil kunne bygge billigere kredsløb med en større densitet af transistorer, år for år, minimum de næste 10 år. Ser vi på historien har Moores forudsigelse holdt stik, frem til inden for de sidste 10 år ca. hvor man grundet varmeudvikling (hvilket Moore nævner, men ikke mener vil være et problem da han mener man nemt kan komme til at køle overfladen af komponenterne, og da han mener man kan bruge samme mængde strøm til flere transistorer, når afstanden bliver mindre, og derved undgå en større varmeudvikling) har været nødt til at se på andre løsninger end blot at øge antallet af transistorer.

[1] *Cramming more components onto integrated circuits*, s. 2

2.2 Situationen i dag

For at se hvordan det ser ud for den transistorbaserede computer i dag kan man efterfølgende se på artiklen *Supercomputere – Mange bække små...* af Brian Vinter, fra 2010. Denne beskriver hvordan vi i dag netop har ramt en mur for hastigheden af en enkelt processor, og hvordan den udvikling som Moore altså snakkede om i 1965 altså ikke længere gælder for hastighed, men kun for antallet af transistorer per chip.

Dette skyldes, at det at øge frekvensen yderligere resultere i så høje temperaturer, at man med den tidligere udvikling, i 2007 skulle have en 16 GHz processor med en temperatur på godt og vel 6000 grader celsius. Derfor er denne udvikling stoppet, og i stedet må man kigge nye veje for at fortsætte med at presse ydelse ud af den transistorbaserede computer. Specfikt nævner Vinter at Moores lov (en tommelfinger regel der siger at en transistors størrelse halveres hver 18. måned, og tidligere forudsagde at en CPU's hastighed blev fordoblet hver 18. måned) på sin vis stadig er gyldig, da computere stadig for mere regnekraft i form af flere kerner (læs: regnekraft i form af multitrådning). Om dette siger Vinter dog:

"Det lugter af et simpelt regnestykke fra folkeskolen: Hvis én mand kan grave en grøft på en time, hvor hurtigt kan to mænd så gøre det? En halv time ville vi nok hurtigt gætte på, men hvad med 1.000 mænd? Kan de gøre det på 3,6 sekunder? Det er klart for enhver, at man ikke kan grave en grøft på 3,6 sekunder – de mange mennesker ville simpelthen stå i vejen for hinanden. Sådan er det også, når vi skal parallelisere computerprogrammer; det kan være meget vanskeligt, selv for de letteste problemer." [2]

Med andre ord ligger der altså en ny udfordring i forbindelse med at lave algoritmer og programmer der kan udnytte de mange kerner som computere vil have i fremtiden. Endvidere kommentere Vinter også på det faktum at hukommelsen i en computer i dag ikke kan følge med processoren:

"Det helt store problem ligger i selve von Neumann-modellen, nemlig i, at CPU'en hele tiden skal læse fra hukommelsen – og desværre er hukommelsen meget langsommere end CPU'en. Hukommelsen i en computer er opbygget i en langsommere teknologi for at kunne presse så meget hukommelse ind som muligt." [3]

Dette løses i dag vha. af en cache-hukommelse, der ligger som lag imellem processoren og hukommelsen, men endvidere taler Vinter om tre alternativer som der i øjeblikket kigges på.

Det første er at gøre hukommelsen hurtigere, ved at benytte sig af en lille men meget hurtig hukommelse til hver processorkerne, og så lade hver af disse have et lille hjælpekomponent som de bruger til at flytte hukommelsen mellem den hurtige og den langsomme hukommelse. Dette er en teknik IBM i øjeblikket kigger på og benyttes bl.a. i Sonys Playstation 3.

Det andet er at gøre hukommelsen hurtigere i gennemsnit (en teknik firmaet NVidia arbejder på). Dette gør de ved at hente mange flere bytes fra hukommelsen af gangen (specifikt 448), og derefter deles disse ud til et tilsvarende antal kerner. Således gøres det op til programmører at skrive programmer der kan benytte dette.

[2] *Supercomputere – Mange bække små...*, s. 25
[3] Ibid., s. 26

Det trejde er at gøre processoren relativt langsommere. Idéen er at når man alligevel skal programmere til flere processorkerner, kan lige så godt skrive programmet til flere end det faktiske antal kerner. Dette kan udnyttes af systemet da processorene gøres langsommere ved at de deles op i virtuelle processorer. Således kan hukommelsen bedre følge med processoren. Dette betyder dog igen når man programmere skal skrive det lige så mange kerner som der er virtuelle kerner.

3 Kvantecomputeren vs. den fysiske Turingmaskine

Til at beskrive computeren som beregningsenhed og matematisk fænomen benyttes traditionelt en model defineret i 1936 af matematikeren Alan Turing. Denne maskine (Turingmaskinen) vil ikke blive beskrevet i detaljer her[4] - det essentielle ved Turingmaskinen er for os de implikationer modellen har for selve begrebet beregnelighed. Hagar udtrykker det:

"That the Turing machine model (what we now call "an algorithm") captures the concept of computability in its entirety is the essence of the Church-Turing thesis, according to which any effectively calculable function can be computed using a Turing machine." [5]

Turing-modellen kan også tænkes som en fysisk model, såfremt man omsætter fysikkens centrale parametre (tid og rum) til deres abstrakte beregningsmæssige sidestykker (hukommelse og antal af beregningstrin)[6]. De problemer der kan løses af en klassisk Turingmaskine falder i to klasser [7] afhængigt af hvor effektivt de kan løses:*Medgørlige* problemer kan beregnes i polynomiel tid af en Turingmaskine, hvor *umedgørlige* problemer kun kan løses i exponentiel tid.

Det har hidtil vist sig umuligt at finde en klassisk fysisk simulation af Turingmaskinen som er i stand til at løse umedgørlige problemer mere effektivt end denne nedre grænse. Med dette for øje forsøger man nu at etablere en model som bygger på et andet fysisk fundament, navnligt kvantemekanikken.

3.1 Hvad er en kvantecomputer?

Kvantecomputeren er grundlæggende set en en teoretisk maskinemodel[8] som benytter elementer fra kvantemekanikken (superpositioner, *entanglements*) til at repræsentere og operere på data. Groft forenklet adskiller kvantecomputeren sig fra den klassiske transistorbaserede maskine ved ikke at være klart deterministisk: Hvor den traditionelle maskines grundlæggende enhed er det binære tal (hvis eneste mulige tilstande er 1 og 0) opererer kvantecomputeren på qubits - en enheden som kan repræsentere et 1, et nul eller en kvantesuperposition af begge tilstande. Dette har bl.a. signifikante implikationer for lagring, da én qubit kan lagre sit data i flere tilstande simultant:

[4] For en nærmere beskrivelse se artiklen *Quantum Computing*.
[5] *Quantum Computing*, s. 3
[6] Se Ibid. s. 5
[7] Betegnet hhv. P og NP - en nærmere diskussion af disse udelades her men behandles i Ibid. s. 4
[8] Introduceret første gang i 1982 af fysikeren Richard Feynman

"[...] the amount of information that can be stored in a system of n unmeasured qubits grows exponentially in n" [9]

Som teksten pointerer, bl.a. i udlægningen af de forskellige konkrete kvantealgoritmer, kommer denne forsøgning i mulig lagring ikke uden udgifter - her hovedsageligt relateret til hvordan qubitdataet hentes effektivt fra lageret.

3.2 Udviklingsperspektiver og begrænsninger

Det mest essentielle perspektiv i kvantecomputerens udvikling må siges at være dens potentielle omvæltning af de etablerede grænser for beregnelighed af umedgørlige problemer. En kvantecomputer kan, teoretisk set, beregne uhyre beregningskrævende operationer med eksponentielt speedup for visse kendte algoritmer [10]. Det er dog på nuværende tidspunkt stadigvæk et åbent spørgsmål hvorvidt kvantecomputeren kan siges at være mere effektiv end den klassiske Turingmaskine i alle tilfælde, og ikke blot i isolerede tilfælde med bestemte algoritmer.

Af begrænsninger ifbm. udviklingen af kvantecomputere kan én fremhæves som særligt central, navnligt det faktum at kvantecomputeren er meget kompleks at realisere som en fysisk maskine: skønt de grundlæggende elementer i en sådan maskine er fremstillet, er det pt. ikke lykkedes forskningen at kombinere disse til en samlet universel maskine[11].

Derudover er en potentiel hindring ved kvantecomputerens dens drastisk mere teoretisk involverede fundament ift. den klassiske maskine: Som fænomen er kvantecomputeren ganske simpelt betragteligt sværere at forstå, og dermed at programmere til. Som Hagar pointerer:

"One of the embarrassments of quantum computing is the fact that, so far, only one algorithm has been discovered [...] that is significantly faster than any known classical one. It is almost certain that one of the reasons for this scarcity of quantum algorithms is related to the lack of our understanding of what makes a quantum computer quantum." [12]

4 DNA-computeren

DNA-computeren går kort sagt ud på, at bruge DNA-strenge som datamodeller/objekter og bruge deres biologiske egenskaber til dataprocesser/beregninger. Det banebrydende eksperiment gik ud på at beregne et lille eksempel på Hamilton-sti-problemet ved hjælp af DNA-strenge og biologiske metoder og processer; bl.a. "Watson-Crick pairing", "polymerases" og "microscopic iron balls" (alle metoder er med engelsk navn). Han fulgte algoritmen, hvor man fremstiller en stor mængde tilfældige veje mellem de mulige knuder, frasortere dem der er for lange og for korte, ikke starter eller slutter i de givne start- og slutknuder, gennemløber samme knude flere gange og dem der ikke gennemløber hver knude. Til sidst, hvis der er nogen veje tilbage i mængden,

[9] Ibid. s.
[10] Oplagte eksempler på sådanne er Shors faktoriseringsalgoritme og Grovers kvantesøgningsalgoritme.
[11] Se Ibid. s. 19
[12] Ibid. s. 20

er svaret at der findes en sti fra start- til slutknuden der gennemløber alle knuder præcist én gang. Hver sti mellem to knuder og hver knude(i artiklen er stier beskrevet som flyruter og knuder som byer) blev beskrevet ved en bestemt kodning af en DNA-streng. Strengen kan symbolsk deles på midten, så knuden har et fornavn, hvor en sti kan gå til, og et efternavn, hvor en sti kan gå fra. Ligeledes for en sti, beskriver den første ende af strengen knuden, som stien går fra, og den anden ende beskriver knuden, som stien går til. Hver DNA-streng har en komplementær DNA-streng, dens "Watson-Crick complement", hvor f.eks. strengen TCGG er komplimentær med strengen AGCC. For hver knude blev den komplimentære streng fundet, og sammen med strengene for stierne blev disse produceret (omkring 10^{14} molekyler pr. streng) og ført til et glas vand, med et par yderligere få ingredienser som salt. Inden for ca. et sekund var løsningen "beregnet" i glasset, idet alle stier fandt sammen med deres ene halvdels komplimentære streng, der var halvdelen af en knudes streng, og deres anden halvdels komplimentære, halve knudes streng. Derved blev alle knuder forbundet på de måder, der var muligt mellem dem, hvilket gav en stor mængde tilfældige veje. Herpå blev der, ud fra algoritmen og over en uge, sorteret i bl.a. længder og start- og slutknuder, f.eks. ved at tilføre mikroskopiske jernkugler til de muligt korrekte DNA-strenge og bruge magnetisme til at trække dem fra resten af mængden. Til sidst var svaret fundet, hvis der var nogen DNA-strenge tilbage. Og det var der.

De stærke sider ved DNA-computeren er dels den utroligt hurtige udregning pga. sin stærkt paralleliserende metode, og dels pga. den store datalagring på meget begrænset plads. Her nævnes i artiklen at et gram DNA, svarende til en kubikcentimeter i tørret tilstand, kan lagre data svarende til omkring 1 billion cd'er (hvis en cd svare til ca. 700mb, svare det altså til 700 billioner mb data eller 667572021 tb data) hvilket er utroligt sammenlignet med dagens teknologiske muligheder - og prisklasser. Derudover giver følgende citat en god idé om potentialet i molekulær-computerens beregninshastighed ift. energien, som også især er i fokus for tiden (med energibesparende computere og servere og desuden at begrænse varmeudledningen):

"Molecular computers also have the potential for extraordinary energy efficiency. In principle, one joule is sufficient for approximately $2 * 10^{19}$ *ligation operations. This is remarkable considering that the second law of thermodynamics dictates a theoretical maximum of* $34 * 10^{19}$ *(irreversible) operations per joule (at room temperature). Existing supercomputers are far less efficient, executing at most* 10^9 *operations per joule."* [13]

Altså har den molekulære computere et meget større potentiale for at opnå maksimal energieffektivitet end transistor-computeren.

Man kan forestille sig, at molekulære computere, pga. deres umiddelbart langsomme udprintstid (tiden det tager for at modtage resultatet) kan bruges, i hvert fald til at begynde med, som en ny slags super-computer. Den kan potentielt udregne problemer, der for eksisterende supercomputere tager markant længere tid (fra måneder til årtier, og mere) at udregne, end tiden det tager for at udprinte resultatet fra molekulær-computer. Derved kan man udnytte dens hurtige regnehastighed, selvom resultatet er besværligt at udprinte. Derimod vil man i den kommercielle computerverden få svært ved at bruge molekulær-computeren, idet beregningerne er små ift. beregninger man bruger super-computere til, og samtidig skal bruges umiddelbart efter aktiveringen af beregningen. Det går f.eks. ikke, at det tager en uge at få opdateret beløbet på sin bankkonto, efter man har trukket penge fra den, mens det for f.eks.

[13] *Computing with DNA*, s. 61

beregninger til kortlægning af hjernen eller udregning af logistiske problemer (som eksemplet i artiklen, blot større) i forvejen tager meget lang tids udregning, hvorfor det kun er positivt man kan gøre det så markant hurtigere.

5 Diskussion og konklusion

bubber was here <3 bubber was not here.. no, that was somebody else...