Norges teknisk-naturvitenskapelige universitet

INSTITUTT FOR DATATEKNIKK OG INFORMASJONSVITENSKAP

LF SEMESTERPRØVE
TDT4258 MIKROKONTROLLER SYSTEMDESIGN

Onsdag 6.mai 2009

Tid: 1025 - 1155

Tillatte hjelpemidler: D: Ingen trykte eller håndskrevne hjelpemidler tillatt.
Bestemt, enkel kalkulator tillatt

Bruk svarskjemaet på siste ark til å angi svarene dine.

Riktig svar: 2 poeng
Galt svar: -0,5 poeng
Ubesvart: 0 poeng
Flere avkryssinger pr. deloppgave: -0.5 poeng!

Semesterprøven gir maksimalt 20 poeng, som så regnes om til å utgjøre 30% av sluttkarakteren i faget.

SVARENE SKAL ANGIS PÅ EGET SVARARK BAKERST I OPPGAVESETTET

a) Hva er ”wait states”?

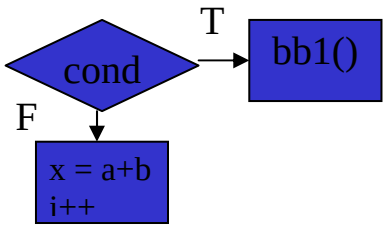
A1. En av tre tilstander en prosess kan settes til av operativsystemet.

kommentar: Tull og tøys.

A2. En nodetype i en kontroll-dataflytgraf (CDFG)

Feil, CDFG har nodar av type valgnoder og dataflytnoder, start, stop etc

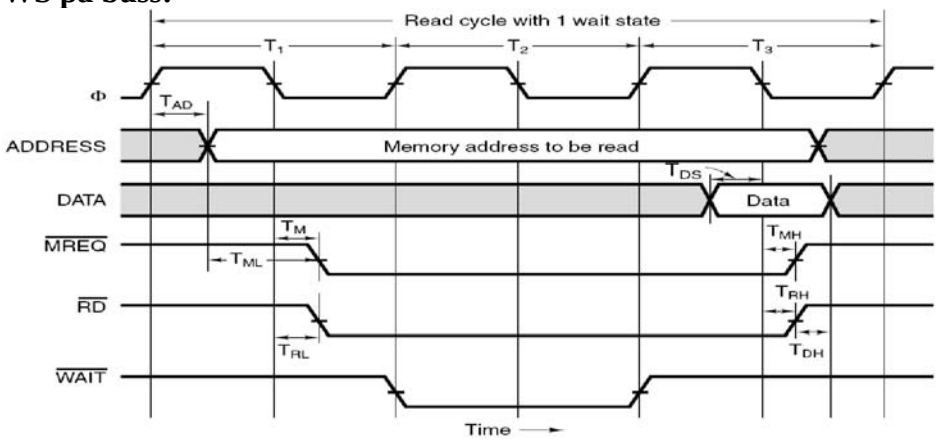
CDFG:



A3. Introduksjon av ekstra klokkeperioder ved kommunikasjon med eksterneenheter som har for lang responstid.

Dette er rett, for å kunne handtere einingar med forskjelig responstid kan ein legje inn ekstra klokkeperiodar som gir einingen meir tid til å svare.

WS på buss:



A4. Angir antall klokkeperioder mellom hver gang CPU sjekker tilstanden til en enhet når en ”Busy-wait”-programvareimplementasjon er brukt.

Dette er også berre tull og tøys, kunn "wait" i namnet.

A5. Tilstander i en tilstandsmaskin som kun kan returnere til starttilstanden (idle/wait).

Dette er også berre tull og tøys, kunn brukt "wait" i i spørsmålet.

b) Innen design av innvevde systemer nyttes begrepet ”design patterns”. Hva ligger i dette begrepet?

B1. Generell beskrivelse av løsningsmetode for en problemklasse. Dette innbefatter en beskrivelse som er uavhengig av en spesifikk implementasjon. Tilstandsmaskiner og sekvensdiagram er eksempel på en slik implementasjonsuavhengig beskrivelse

Dette er rett: Rett frå foil i forelesing og bok.

B2. En hierarkisk designmetode som angir de forskjellige nivåene i en designprosess. ”Top-down” og ”bottom-up”er eksempel på ”design patterns”.

”Top-down” og ”bottom-up” er "design processes" eller design prosesar basert på å innføre abstraksjonsnivå.

B3. ”Design patterns” beskriver hvordan partisjoneringen mellom maskinvare og programvare gjøres (såkalt HW/SW designflyt).

HW/SW designflyt er ein metode (eller eit forsøk) på å gjere programmvare og maskinvare design som ein prosess, der ein gjennom heile designet prøver å optimalisere ved å legge oppgåver i maskinvare eller programmvare. HW/SW designflyt kan nyttast innan abstraksjonsnivåa i ein design prosess.

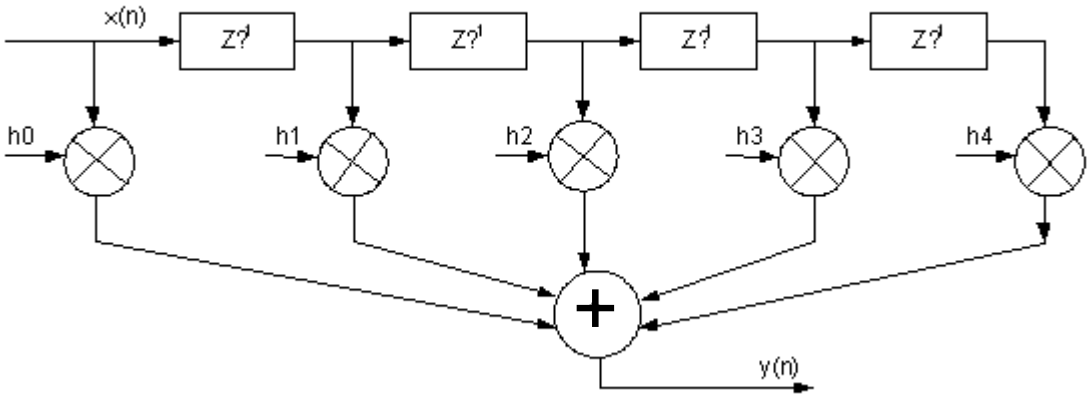
B4. Den prosessen der en gruppe går gjennom et design. En slik prosess inkluderer gjerne designere, ledere, kunde og eventuelt publikum.

Dette er ein beskrivelse av "Design review".

B5. Angir den felles designflyten som maskinvare og programvare går gjennom etter at det er gjort en maskinvare/programvare partisjonering.

Dette er ein beskrivelse av "Hierarkisk HW/SW-flyt".

c) Et ”Finite Impulse Response” (FIR) filter skal implementeres. Valget står mellom en typisk Digital Signal Processor (DSP) og en Field Programable Gate Array (FPGA) implementasjon. Hvilken påstand er **ikke** korrekt for dette valget av design. Blokkdiagram av et typisk FIR-filter er vist i figuren under.



x(n): sample in, Z: delay, h: coefficient, y(n): sample out

C1. Klokkefrekvensen til DSP-en begrenser størrelsen på filteret.

Som alle prosessorar (basert på von Neumann arkitektur) utfører DSP-ar programmet sekvensielt. I figuren er det fleire operasjonar som må gjennomførast (f.eks. mange multiplikasjonar). DVS at ved over ein gitt størelse på filteret (antal operasjonar som må gjerast for å køyre ein iterasjon (gjere ferdig utdata)) vil tiden det tar å prosessere bli for stor. Ved ein gitt klokkefrekvens vil då antall ferdig proseserte utdata gå ned når filter størelsen aukar (ekstremt: mot null når filter størelsen går mot uendeleg). **Påstanden er rett**

C2. Antall gates i FPGA-en begrenser størrelsen på filteret.

Dette er også relatert til resursar. I ein FPGA kan ein implementere alle del-operasjonane i filteret som eigne HW-modular. DVS at antal gates avgrensar antal delmodular som kan implementerast. **Påstanden er rett**

C3. Ytelsen til filterrutinen i en DSP er avhengig av eventuelle andre oppgaver som DSP-en utfører.

Som i C1, DSP-ar utfører programmet sekvensielt. DVS at viss det er andre oppgåver som krever tid (klokkeperiodar) så går tiden (klokkeperiodar) som er tilgjengeleg til filterrutina ned. Dette påvirkar då mulig størelsen (og sample frekvensen), dvs ytelsen. **Påstanden er rett**

C4. Ytelsen til en FPGA-implementasjon er tilnærmet uavhengig av andre oppgaver implementert i FPGA-en

Som i C2, i ein FPGA kan modular implementerast som seperate einingar som kan køyre i paralell. DVS ytelsen vert ikkje påvirka av prosessering i andre modular. **Påstanden er rett**

C5. Ved lik klokkefrekvens vil maksimalt antall samples som kan behandles i sekundet være likt.

Sidan ein DSP utfører programmet sekvensielt må han bruke meir enn ein klokke periode for kvart sample. Antal avhengig av størelsen på filteret (f.eks. antal MultiplyAccumulate-instruksjonar). DVS antal sample i sekundet er > f. I ein FPGA kan heile filteret paraleliserast. DVS at det er mulig å få gjennom antal sample i sekundet = f. **Påstanden er feil.**

d) En mikrokontroller har følgende avbruddskilder:

1. Synkron serieport
2. Asynkron serieport
3. Generelt eksternt avbrudd 1
4. Generelt eksternt avbrudd 2
5. Generelt eksternt avbrudd 3
6. Timer avbrudd 1
7. Timer avbrudd 2
8. NMI

Hvilken påstand om den interne registerstrukturen er korrekt?

D1. Interrupt mask-registeret er på 7 bit. Interrupt priority-registeret er på 21 bit

Detter er rett. Det trengs 7 mask bit, NonMaskableInterrupt kan ikkje maskerast. Med 3 bit kan interrupts prioriterast i max 8 nivå. Kan altså prioritere dei 7 aktuelle interrupta i 7 nivå. Nivå 0 vert ikkje brukt sidan det er NMI. **Dette er rett**

D2. Interrupt mask-registeret er på 8 bit. Interrupt priority-registeret er på 24 bit

For mange maskbit, NMI skal ikkje maskerast. **Feil**

D3. Interrupt mask-registeret er på 8 bit. Interrupt priority-registeret er på 8 bit

For mange maskbit, NMI skal ikkje maskerast. **Feil**

D4. Interrupt mask-registeret er på 7 bit. Interrupt priority-registeret er på 7 bit

Kan berre skilje mellom to nivå. DVS ingen prioritering sidan nivå 0 er NMI. **Feil**

D5. Interrupt mask-registeret er på 3 bit. Interrupt priority-registeret er på 8 bit

For få maskbit. **Feil**

e) Hvilken påstand om effektforbruk er **ikke** riktig, (alternativ at påstand E5 er riktig)?

E1. Dersom vi halverer forsyningsspenningen vil effektforbruket bli redusert til ¼.

$P = (U^2)/R$. Påstanden er korrekt

E2. Reduksjon av klokkefrekvens reduserer det totale energiforbruket for å utføre et gitt program.

Sjølv om energiforbruke for ein gitt tidseining går ned så vil antal klokkeperiodar for å utføre programmet være likt. DVS det er ingen reduksjon i energiforbruk. **På standen er FEIL**

E3. Statisk strømforbruk er uavhengig av klokkefrekvens.

Statisk strømforbruk for ein krets er det strømforbruke kretsen har utanom endring i signal nivå (f.eks. klokke). **Påstanden er korrekt**

E4. Statisk strømforbruk for en enhet kan kun elimineres hvis maskinvarekomponenten kobles fra forsyningsspenningen.

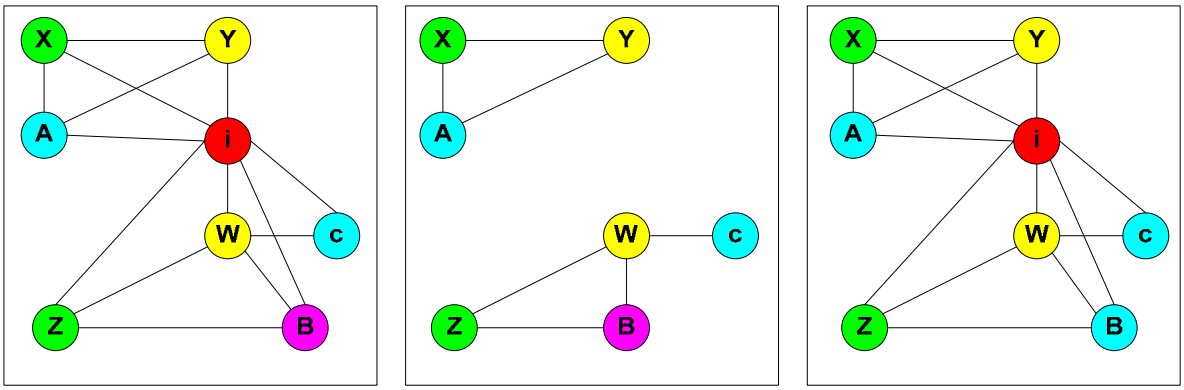
Statisk strømforbruk er for det meste lekasjestraumar. ein må då kople vekk forsyningsspenningen for å eliminere. **Påstanden er korrekt**

E5. Ingen av påstandene over er korrekte.

Sjå over.

f) I figuren under er en kodesnutt forsøkt overført til en konfliktgraf med graffarging. Hvilken påstand er korrekt for figuren?

```
for (i = 0; i < size; i++) {
        a = x * y[i];
        b = w + z[i];
        c = w + c;
}
```



F1. Konfliktgraf 1 og 2 representerer koden korrekt som en graf uten konflikter.
F2. Konfliktgraf 1 og 3 representerer koden korrekt som en graf uten konflikter.
F3. Konfliktgraf 2 og 3 representerer koden korrekt som en graf uten konflikter.
F4. Konfliktgraf 2 representerer koden korrekt som en graf uten konflikter.
F5. Ingen av konfliktgrafene er en korrekt representasjon av koden.

size er ikkje med i nokon av grafane. size er aktiv saman med andre variablar og krever ein node så **F5 er korrekt svar**.

g) Hvilke(n) optimaliserings grep kan gjøres i koden?

```
for (i = 0; i < n*m; i++) {
        c[i] = a[i] * b[i];
}
```

G1. ”Code motion”

Kan flytte ut n*m

G2. ”*Loop distribution*”

Ikkje nokon optimalisering..

G3. ”*Loop fusion*”

Kunn ein loop
.
G4. ”Code motion” og “*loop unrolling*”

*kan flytte ut n*n og rulle ut løkka*

G5. ”*strength reduction”* og “*loop unrolling”*

Nei

Kommentar til oppgåva:
Huks at det står **kan** og at det er ikkje nokon anna informasjon enn koden. MEN, ved enkelte tilfeller er “*loop unrolling”* ein langt frå rettframm prosess så me gir rett på G1 og G4.

h) AVR32 relatert:

Her følger fem påstander om AVR32 prosessoren og AVR32-basert mikrokontrolleren dere har brukt i labøvingene 1, 2 og 3.
Hvilke av dem er **IKKE** riktig?

H1. AVR32 arkitektur støtter 4 prioritetsnivåer av regulære, "maskable" avbrudder.

Påstanden er korrekt

H2. Audio Bitstream Digital-Analog Converter, ABDAC, som ble brukt i labøving2 får klokkesignal for D/A konversjon fra "Power Manager".

Påstanden er korrekt

H3. Klokkesignalen som bestemmer med hvilken frekvens ABDAC sender ut audio data kan bare komme fra de to "phase-locked loops" (PLLs).

Påstanden er ikkje korrekt

H4. For bruk av PIO avbrudd kreves det at avbruddskontroller programmeres først, dvs avbruddskontroller må programmeres før PIO-kontroller.

Sjå databla eventuelt øvingshefte for detaljar. Datablad 19,. Parallel I/O controller (PIO). 19.4.4. side 251 (AT32AP7000 Preliminary)

H5. Det følger en linje av AVR32 assembly:
csrf 16
Dette representerer instruksjonen "clear status register flag 16", og betyr "enable global interrupts".

Dette stemmer sjå databla. **Påstanden er korrekt**

i) "Device Drivers" relatert:

Hvilke av de følgende påstander er **IKKE** riktig:

I1. Kjernemoduler kan lastes og fjernes dynamisk.

I2. Kjernemoduler kan bruke funksjoner fra C standardbiblioteket <stdio.h>.

Dette stemmer ikkje. **Påstanden er ikkje korrekt.** Sjå øvingshefte side 56.

I3. Kjernemoduler er eventbaserte.

I4. En driver må be om tilgang til en PIO port og få tillatelse til å bruke den før PIO porten brukes av driveren.

I5. ioctl(...) funksjon kalles fra brukerområdet, men den utføres i kjerneområdet.

j) VHDL relatert:
Hvilke påstand er **IKKE** riktig?

J1. Både "component instances" og "process statements" er "concurrent statements" i VHDL.

J2. Generelt kan testbenkentiteter instansieres.

Dette stemmer ikkje testbenker er ein VHDL-modul som instansierer HW-designet. **Påstanden er ikkje korrekt.** Sjå øvingshefte side 69.

J3. En prosess i VHDL er sensitiv bare til "events" som skjer på signalene som står i sensitivitetslista til prosessen.

J4. Syntesen av design som er implementert i VHDL avhenger av synteseverktøyet som brukes.

J5. Tilstandsmaskiner er en måte å lage kontroll-logikk på i VHDL.