Danmarks Tekniske Universitet

# 02321 Hardware/Sofware Programmering

## 3 ugers projekt


**Udarbejdet af:**

**Gruppe nr. 1**

**s093485 - Christensen, Anders Jan**

**s072657 - Tang Khoa Nguyen, Nikolai**

**s093478 - Hansen, Mathias**

**s010164 - Mitri Emil Kjær Rebeiz, Sami**

**s000536 - Vandall Zimsen, Jakob**

DTU Informatik
Institut for Informatik og Matematisk Modellering


**02321 Hardware/Sofware Programmering**

**Vejledere: Edward Alexandru Todirica**
**Afleveringsfrist: 21. juni 2011 kl. 13:00**
**Rapporten indeholder 19 sider inkl. forside**

# Timeregnskab

| Dato | Deltager | Design | Impl. | Test | Dok. | Andet | I alt |
|---|---|---|---|---|---|---|---|
| 2011-01-03 | Mathias | | | | 1 | 4 | 5 |
| | Total | 24 | 116 | 78 | 121 | 57 | 396 |

## Akkumuleret timeregnskab

I det akkumulerede timeregnskab er kun vist de aktive medlemmer i gruppen. Ved 3-ugers periodens begyndelsen blev gruppen reduceret med ét medlem.

| Sami | Mathias | Anders | Nikolai | Jakob |
|---|---|---|---|---|
| 100 | 200 | 300 | 100 | 200 |

## Rollefordeling

### Analyse/Design

**Lorem ipsum:** ?

### LC3 Implementering

**IO:** Primært udarbejdet af Mathias

**VGA samt ROMs til VGA:** Primært udarbejdet af Jakob & Nikolai

**Hukommelse:** Primært udarbejdet af Anders & Sami

### Spil implementering

**Lorem ipsum:** ?

### Test

**Lorem ipsum:** ?

**Dokumentation**

**Lorem ipsum:** ?

# Indholdsfortegnelse



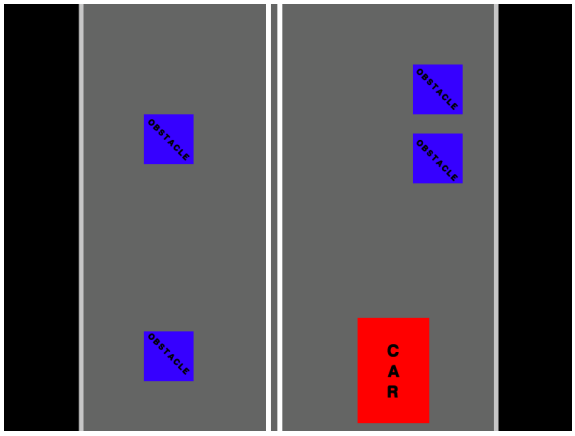
# Figurer

I dette 3 ugers projekt skal der sammensættes en komplet computer, baseret på `von Neumann arkitekturen` udfra en udleveret implementeret `LC3 CPU`, computeren implementeres på et `Virtex-II` Pro FPGA board.

LC3 Cpu'en er leveret til os som en black box, hvortil vi skal tilslutte de andre nødvendige komponenter for at have en komplet kørende computer; memory, vga samt et rat påsat en A/D-converter som I/O til LC3'en. Et eksternt stykke hardware i form af en skærm er også nødvendig.

Såfremt det bliver nødvendigt, kan der være tale om at dele af spillet implementeres på en almindelig PC og med data sendt over seriel forbindelse til `LC3` computeren. Derefter udvikles et spil baseret på den implementerede LC3 computer. Det er valgt at udvikle et bil spil, hvor bilen skal styres af et fysisk rat eller tastatur. Målet med spillet er at undgå forhindringer i form af andre biler, kasser og lignende som indsættes på kørebanen tilfældigt til forskellige tilfældigt tidpunkt.

På figur 1 ses et simpelt mock-up over hvordan spillet kunne se ud, hvor der er indtegnet en simpel baggrund, en bil man styrer samt forhindringer man skal undgå.

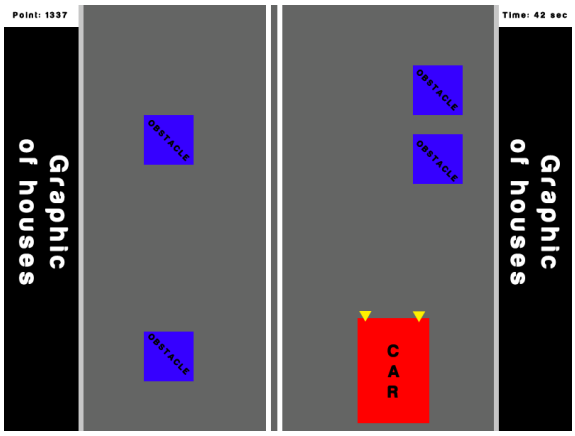



Figur 1: Simpelt mock-up over systemet

Afhængigt af tiden op til aflevering, påtænkes der at udvide projektet med en eller flere ekstra tilføjelser. Disse tilføjelser kunne f.eks. være at indsætte flere *forskellige* forhindringer på vejen, benytte *forskellige* baggrunde, have flere *forskellige* baner, lave bedre grafik eller at implementere lyd.

På figur 2 på næste side ses et tænkt videreudviklet mock-up over spillet, hvor points og tid også er vist.

Vores mål og krav er, at implementere et minimum af funktionalitet, som beskrevet i den opgave case som ligger til grund for projektet, inden for den givne tidshorisont. Ikraft af vores roller som udviklere, stiler vi naturligvis efter at lave det bedst kørende simulator spil på den bedst kørende LC3 computer. Tiden er dog en afgørende faktor for, hvor meget vi når at implementere ud over vores minimum krav. Vi påtænker at kode hardwaren i `VHDL` og applikationen i `C`.




Figur 2: Videreudviklet mock-up over systemet

Desuden er det ønskeligt fra kundens side, at bruge viden fra andre af semesterets kurser til projektet, herunder f.eks. implementering af en database til at gemme spiller navne og highscores i spillet, med henvisning til kurset `02344 OOAD og databaser`.

# Kapitel 1

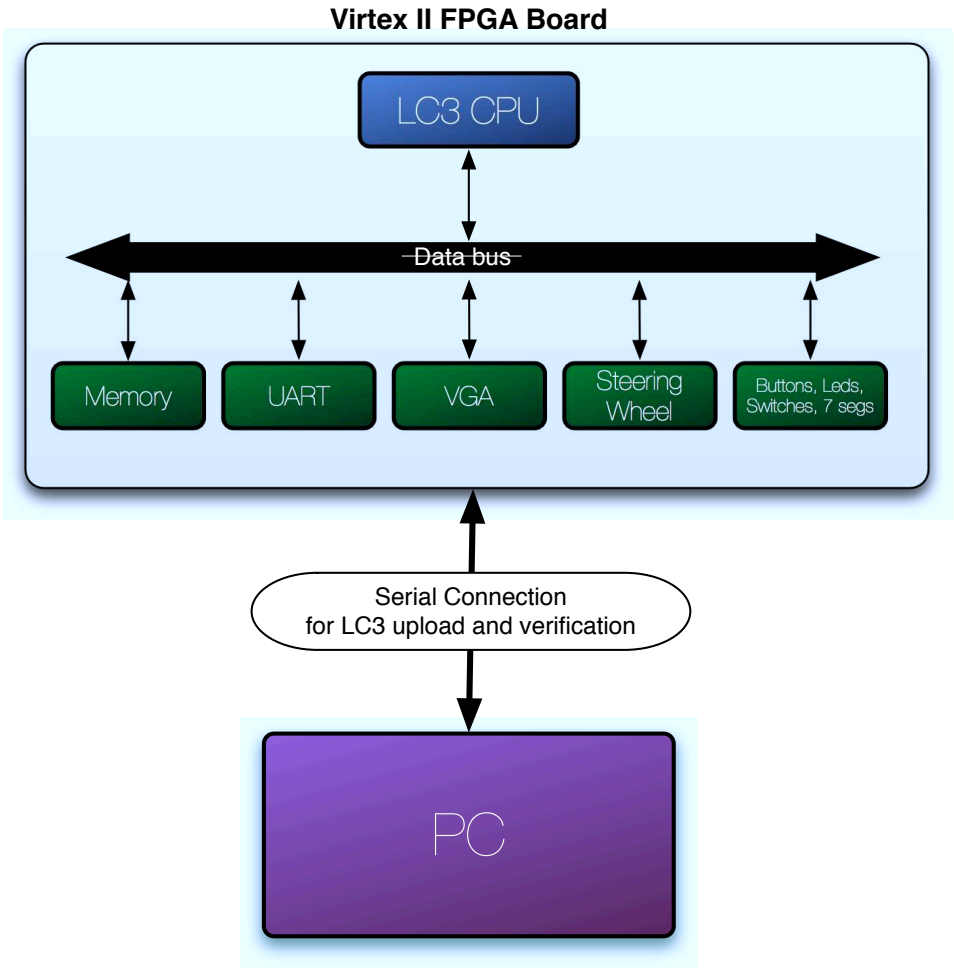
# Design



Vores system som overordnet er beskrevet på figur 1.1 på den følgende side, er opbygget af et `FPGA board` der implementerer en computer kørt af en `LC3 CPU`. Til computeren er der af eksterne I/O komponenter tilknyttet en skærm, et I/O board[1] samt et rat som essentielt er en variabel modstand. Desuden benyttes der to-vejs seriel kommunikation til en PC til upload af bruger programmer til `LC3` computeren samt verificering af disse.

Spillet er først implementeret på PC i C kode, derefter skal indholdet overføres til FPGA board vha. en serial forbindelse. Indholdet af spillet gemmes i block ram på FPGA. En VGA skærm viser "output"altså baggrund,forgrund( bil, forhindringer), og er opdateret løbende, for at man kan se og følge med i hvad det sker og dermed kan spille. Spillet kan styres direkte fra FPGA board vha. en styrrings enhed, i dette tilfælde har vi valgt at bruge et rat i stedet for et keyboard.

[1] Digilent DIO4™I/O board




Figur 1.1: Design system

LC-3 system efter vores design består af forskellige komponenter som er sat sammen, dette kan tælles: CPU, rat, MEM, UART og forbindende ”busser”.

Udover VGA skærm kan man koble flere hardware enheder til når der er behov for det. De kan være lydkort,... o.s.v. CPU har til formål at udføre de beregninger som ligger til grund for hvilke instruktioner som skal udføres, dvs. hvad applikationen ”siger” der skal ske hvornår.... MEM bruges generelt til at gemme div. Data, bl.a. de respektive instruktioner som skal udføres ... UART bruges til at ”holde” styr på at sende/modtage data instruktioner til/fra komponenterne i computeren .... rattet bruges til at styre bilen, som kører i forgrunden. Figur xx viser vores overordnede system. (OBS! Vi skal selv tegne LC-3 system diagram som ligner denne figur)

## Design valg 1.1

### 1.1.1 Wrapper

Til LC3 systemet valgtes det at bruge wrappers til de elementer der både har et input og et output, dette gjorte det nemmere at samle padding og tristate buffer. Ved at lave en wrapper kunne der sendes signaler videre ned i LC3 systemet fra bussen.

### 1.1.2 Padding

Padding er en metode hvorved længden af en bitstreng ændres. Da det var nødvendigt at ændre bus adresse længden i f.eks. UART’en fra en længde på 16 bit ned til 8 bit, da det er den længde UART forventer. Dette gøres ved at når UART wrapperen modtager et signal sender den kun de 8 mindst betydende bit videre ned til UART’en at arbejde med, og når et signal modtages fra UART’en paddes der med 0’er foran, så der bliver sendt et signal tilbage til bussen der er konstrueret således ’00000000’ og UART signal. Derved ender vi med et signal på 16 bit igen, som er den længde bussen arbejder med.

### 1.1.3 Tri State Buffer

En tri state buffer fungerer således at der bliver sendt et enable signal der et andet signale angiver om der må skrives eller læses fra et element, og derudover sørger for at hvis det ikke er det element der skal bruge signalet, kun vil modtage Z’er der i et signal er en tom værdi. På denne måde er det muligt at styre hvilke elementer der bliver skrevet og læst fra på en given clock cycle.

### 1.1.4 Chip Selector

Chip selector gør det muligt at lave en kontrol af hvilke adresse der bliver skrevet til, og igennem et I/O adresse register, er det blevet bestem hvilke værdi der har med hvilke signaler at gøre. Ved at tjekke på denne adresse kan man derved bestemme om et signal til tri state bufferen skal være 1 eller 0.

## Ram 1.2

LC3 complete system RAM implementering. Vi har valgte at bruge single-port synchronous RAM. Vi valgte single-port RAM fordi vi i første omgang ikke har brug for at skrive til eller læse fra 2 adresser samtidig, som duel-port RAM ville have givet os lov til. Synchronous tillader at vi bruge block rammen på FPGA boardet, og ikke CLBs rammen, der er forbeholdt til logic og asynchronous RAM. Til det vi skulle lave var der ingen grund til at bruge asynchronous, da alle vores udregninger ikke behøver at blive behandlet sammen clock cycle. Vi implementerede vores RAM, ved at følge bogen (indsæt eksempel/reference) og derefter justere længden af adressen så det passede til den størrelse vores program der blev kørt på FPGA'en havde, ved at reducere størrelsen på rammen, sørgede vi for at syntetiseringen af vores LC3 system ikke tog unødvendig langt tid. Vi blev dog løbende nød til at ændre på adressen længden af hukommelsen, da vores C kode blev længere. Hvad bruger vi rammen til ud over vores c kode?

Vi valgte til vores Ram at lave en Wrapper, der indeholder en tristatebuffer og oversætter signalet til den rigtige længde. Tristate Bufferen sørger for at signal kun kan gå den ene vej, så der enten kan blive læste fra Rammen eller skrevet til den på en given clock cycle. Dette indeholder alle vores elementer, og dette gør sammen med kontrol på bus adressen at vi kun kan skrive eller læse fra det vi ønsker. Padding er den måde vi løser at vi har en bus adresse på 16 bit, men i vores hukommelse kun bruger eksempelvis 14 bit, så vælger vi at bruge de 14 mindst betydende bit når hukommelsen modtager en bus adresse. Hvis hukommelsen sender en adresse til bussen, er denne for kort, dette løses ved at sætte 0'er foran, så vi ender med '00' og adressen fra hukommelsen.

## VGA 1.3

Lorem ipsum dolor...

## Rat 1.4

Lorem ipsum dolor...

## DIO4 I/O 1.5

Lorem ipsum dolor...

## UART 1.6

UART er en forkortelse for Universal asynchronous reciever and transmitter. Vi gik ud fra eksemplet i bogen, og valgte at implementere UART'en da det ville gøre vores debugging meget nemmere. Vi valgte at lave en wrapper til UART'en da vores bus signal er 16 bit langt og det signal vi bruger i UART'en kun er 8 bit. Dette blev gjort ved at når der sendes til UART'en bruges kun (7 downto 0), altså de 8 mindst betydende bits. Og hvis i signal sendes fra UART'en paddes der med 0'er foran de 8 sendte bit. Vi brugte UART'en til at teste om Rammen og CPU kommunikerede korret sammen, via et test program givet til os, dette var et echo program der skrev de input tastaturet fik ud på skærmen.

## Grafik 1.7

Lorem ipsum dolor...

## LC3 Instruktioner 1.8

Lorem ipsum dolor...

## Highscore 1.9

Lorem ipsum dolor...

# Kapitel 2

# Diskussion

Under 3 uger arbejdet med projekt, har vi lært og oplevet mange nye tinge. Det tales om både gode og svære tilgang.

De gode: • Vi får implementeret vores advandced/udvidelse version af spillet, og det virker som forventning. Bilens form og farver kan tegnes efter vores vilje, den kan styres fra raten. • Point statistik for hver spiller kan vises på skærm vha. en lokale database. Dette er en ekstra tinges som vi har implementeret ved brug af viden fra kurset OOAD Database. •

De svære: • Det tager meget længere tid at implementere hele computeren system og få dem op at køre end vi regner med i starten. Hardware/Vhld delen tog os 2 uger at blive færdig med. • Især UART og MEM var en udfordring for os at implementere, der ligger masse ”tricks” bag i. • Efter implementering af hver komponent, har vi det svært at få dem til at ”snakke” sammen med LC-3 CPU, der opstod fejl i forskellige steder i vores program, som tog os ekstrem lang tid at finde rundt og rette dem. • For at nå vores tidsplanlægning om deadline for arbejde er vi nødt til at bruge weekenderne på laboratoriet.

# Kapitel 3

# Videreudvikling



## Hvad vi har nået.. 3.1

Vi har nået .... baseret på start vanskeligheder men føler os tilfredse med det produkt vi har udviklet. Vi har ..... komponenter der virker og et kørende/næsten kørende produkt og hvis vi havde haft mere tid ville vi have bygget videre på .... (grafikken, spillet generelt etc). Vi tænker at en version 2 af dette spil kunne være med .... (bedre grafik, flere baner, større DB til yderligere data der skal gemems, bedre statistik over f.eks. hændelser i spillet så som antal "sejre"kontra antal spillede spil.. etc.) En videre udvikling kunne også være .... (bedre boards, bedre komponenter, måske noget med spil over netværk, etc)

# Kapitel 4

# Konklusion



## ET UDKAST!!! 4.1

Vi har med dette projekt fået en bedre forståelse for sammenspillet mellem hardware og software, hvordan de nødvendige komponenter i LC3'en bruges i samspil med et stykke software - her vores simulator spil. Vi har udviklet os fremadrettet mht. at forstå hvad der egentligt sker på komponent niveau når man bruger en computer, der kører noget software. Vi føler os bedre til at kunne abstrahere på forskellige niveauer mht. at "dykke ned"på/i komponent niveau og få et sådant til at fungere, i sammenspil med et stykke software (her vores simulator spil) og bevæge os op og ned mellem de forskellige abstraktions lag iht. low-level/high-level udvikling. Det har været meget interessant at lave/kode de respektive komponenter(så som memory, VGA.. etc) vi har brugt og derefter få dem til at "snakke"sammen med vores spil. Det har været et meget interessant projekt at gå i krig med, det har givet en meget bedre forståelse for hvad der egentligt sker "inden i"en computer, når man som bruger "bare"sidder foran skærmen og bruger den til div. ting. Vores viden er blevet bredere efter dette projekt. Dog er tidsfaktoren en afgørende faktor for udviklingen af produktet kontra ønskede opnåede mål, vi forestiller os at man ude i erhvervslivet har mere tid til udviklingen, men vi er generelt tilfredse med det vi har nået på den givne tid.. :)

# Bilag

# I/O Registers .1

| Address | Description |
|---|---|
| **xFE00** | Stdin Status Register |
| **xFE02** | Stdin Data Register |
| **xFE04** | Stdout Status Register |
| **xFE06** | Stdout Data Register |
| **xFE0A** | Switches Data Register |
| **xFE0E** | Buttons Data Register |
| **xFE12** | 7SegDisplay Data Register |
| **xFE16** | Leds Data Register |
| **xFE18** | Steering Wheel Status Register |
| **xFE1A** | Steering Wheel Data Register |
| **xFE1C** | Car VGA X position |
| **xFE1E** | Obstacle 1 VGA X position |
| **xFE20** | Obstacle 1 VGA Y position |
| **xFE22** | Obstacle 2 VGA X position |
| **xFE24** | Obstacle 2 VGA Y position |
| **xFE26** | Obstacle 3 VGA X position |
| **xFE28** | Obstacle 3 VGA Y position |
| **xFE2A** | VGA Refresh Tick register |
| **xFE2C** | Vibrator register |
| **xFE30** | VGA Road movement data register |

# Kildetekst .2

# Indholdsfortegnelse

# Figurer