Levdatum: 20150219
Utvecklare: Benny Tall
Epost: benny.r.tall@gmail.com

Slutrapport ingenjörsjobb C

# "Masken Starstruck"

## Sammanfattning

Denna rapport innehåller en beskrivning av det ingenjörsjobb som genomförts i kursen C. Ingenjörsjobbet bestod av att konstruera det kända spelet "Masken" som skapades under 1970-talet.

# Innehållsförteckning

# 1. Inledning

Denna rapport innehåller en beskrivning av det ingenjörsjobb som genomförts i VHDL. Ingenjörsjobbet bestod av att konstruera det kända spelet "Masken" som skapades under 1970-talet. Slutresultatet blev en egen version med namnet "Masken Starstruck", se avsnitt 3 för avsteg från kraven och slutgiltig lösning. Steg 1 i ingenjörsjobbet var att leverera en startrapport och steg 2 var att leverera denna rapport tillsammans med konstruktionen.

# 2. Urspunglig & genomförd tidsplan

Nedan i tabell 1 visas den ursprungliga tidsplanen med aktiviteter.
Ingenjörsjobbet för C-kursen utfördes efter VHDL-kursens ingenjörsjobb och inte parallellt, vilket var den ursprungliga planen. Arbetet utfördes från vecka 6 till 7.

Tabell 1. Ursprunglig tidsplan med aktiviteter.

| Nr | Aktivitet | v.1 | v.2 | v.3 | v.4 | v.5 | v.6 | v.7 | v.8 | v.9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1. | Skriva och leverera startrapport. | | | | | | | | | |
| 2. | Dokumentera krav, funktionalitet och resultat i slutrapport. Aktiviteten pågår under hela utvecklingscykeln. | | | | | | | | | |
| 3. | Utveckling med nedanstående delaktiviteter. | | | | | | | | | |
| 3.1 | Modellera en arkitektur över vilka olika komponenter som ska ingå i konstruktionen och hur de är interagerar.. | | | | | | | | | |
| 3.2 | Undersöka och testa befintliga komponenter som kan användas i projektet. Både egna och från tredje part. | | | | | | | | | |
| 3.3 | Utveckling och programmering av konstruktionen. | | | | | | | | | |
| 4. | Test/Verifikation. ModelSim används vid behov i VHDL-kursen. | | | | | | | | | |
| 5. | Test/Validering på DE2-115-kortet. SignalTap används vid behov. | | | | | | | | | |
| 6. | Slutföra slutrapport och leverera projektet. | | | | | | | | | |

# 3. Krav & specifikation

## 3.1 Ursprungliga krav

För kraven hänvisas till den startrapport (rapport A) som finns med i denna leverans finns en specifikation över hur spelet ska fungera och dess krav.

## 3.2 Ändrade krav & nya önskemål

Under arbetets gång förändrades kravbilden av kunden och nedan är de förändringar som implementerades och som inte fanns med i de ursprungliga kraven.
- De "väggar" som nämns i kraven, och som spelaren ska undvika att krocka med, har istället blivit "stjärnor".
- Masken växer ju längre tid som fortgår i spelet, men den har en maxlängd som inte passeras.

- Masken ökar sin hastighet ju längre tid som fortgår, men den har även en maxhastighet då den inte ökar mer i hastighet.
- Poängen beräknas inte efter hur lång tid spelet spelas, utan istället på hur många svängar spelaren gör med masken.

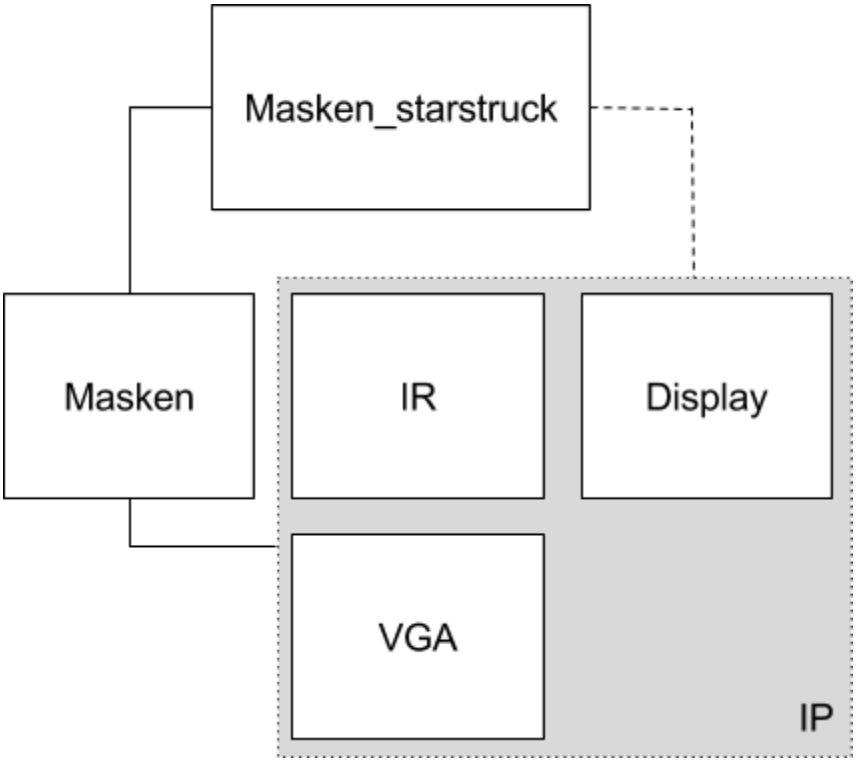
# 4. Systemarkitektur

## 4.1 Sammanfattning

Spelet är utvecklat på ett Altera FPGA-kort och körs på en NIOS II-processor. Den tillsammans med de drivrutiner och komponenter som BSP innehåller erbjuder möjligheter att hantera hårdvaran och de In- och ut-gångar som finns.
Systemkonstruktionen består av en huvudkomponent som hanterar det huvudsakliga flödet i spelet, från start till slut. Denna komponent innehåller i stort sett enbart anrop till andra funktioner som är strukturerade i enskilda och avgränsade filer inom respektive användningsområde, t ex för IR-mottagare och LCD-display.

## 4.2 Komponentöversikt

Nedan i figur 1 visas en bild över mjukvarukomponenterna i systemet.
Den huvudsakliga gången och logiken bygger på att huvudkomponenten Masken_starstruck anropar funktioner i komponenten Masken som i sin tur anropar funktioner i komponenterna IR, Display och VGA. Något enstaka funktionsanrop sker direkt från Masken_starstruck till någon av de övriga komponenterna.




Figur 1. Översiktsbild över komponenterna i systemet.

#### 4.2.1 Masken_starstruck.c

Detta är huvudkomponenten som är den drivande i systemet och består av anrop till funktioner i övriga komponenter enligt ordningen nedan.

initAndResetMasken (Masken.c)
Reset:at och initierar variabler.

initScore display (Display.h)
Tömmer och reset:at LCD-displayen.

resetSnakePositionsArray (Masken.c)
Tömmer array med maskens positioner.

resetWallsPositionsArray (Masken.c)
Skapar nya slumpartade positioner för stjärnorna i spelet och spar dem i array.

clearVGA (VGA.c)
Rensar skärmen från allt innehåll.

showStarsOnVGA (Masken.c)
Ritar ut alla stjärnor på skärmen, från array med positioner.

waitForPlayer (IR.c)
Väntar på att spelaren ska trycka knapp 5 på IR-kontrollen.

execGame (Masken.c)
Hanterar logiken från dess att spelet startats med knapp 5 till dess att spelet avslutas med "Game Over".

displayGameOver (Display.c)
Visar "Game Over" på LCD-displayen och den totala poängen.

waitForPlayer (IR.c)
Väntar på att spelaren ska trycka knapp 5 på IR-kontrollen och på så sätt återstarta spelet.

#### 4.2.2 Masken.c

Komponenten hanterar logiken från det att spelet startas med knapp 5 på IR-kontrollen till dess att spelet avslutas med "Game Over".

#### 4.2.3 Display.c

Komponenten innehåller funktionalitet för att hantera det som behöver visas på LCD-displayen.

#### 4.2.4 VGA.c

Komponenten innehåller funktionalitet för att hantera pixlarna på VGA-skärmen.

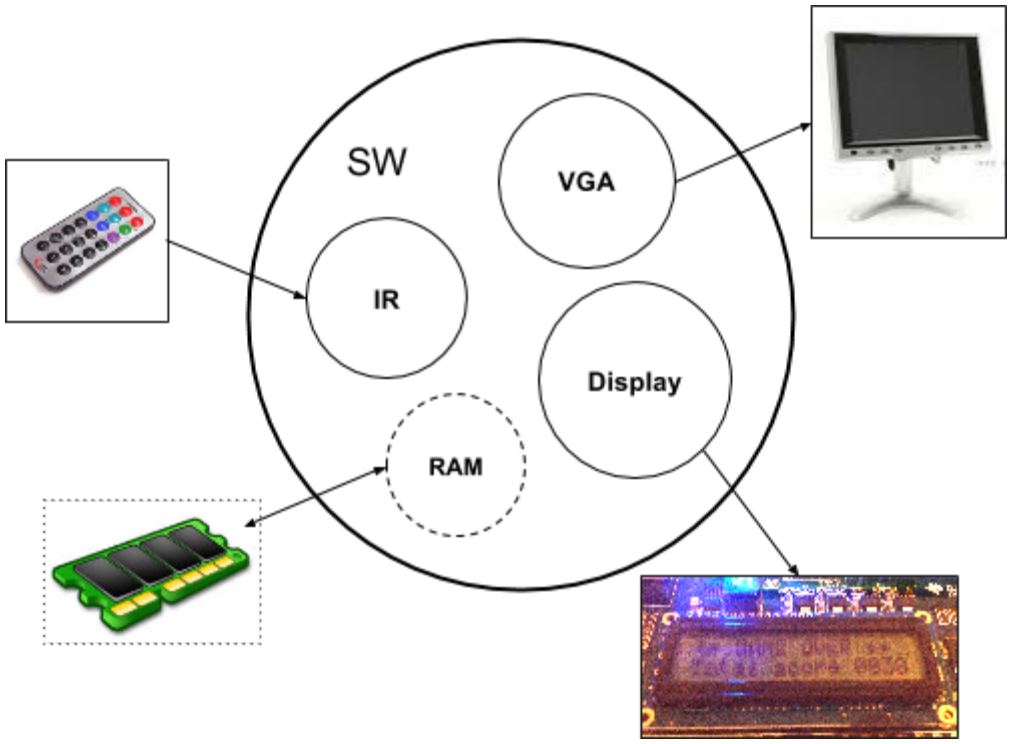
#### 4.2.5 IR.c

Komponenten innehåller funktionalitet för att hantera IR-kontrollen.

### 4.3 In- & utgångar (I/O)

Systemet använder sig av flera in- och ut-enheter för att kommunicera. En VGA-skärm används för presentation av spelet, en IR-kontroll för att styra spelet och en LCD-display för att visa poängen.

Internt i NIOS II processorn finns fler i/o som används internt på Altera-kortet, den mest uppenbara är RAM-minnet som vi använder. Nedan i figur 2 visas en översiktbild av in- och ut-gångarna tillsammans med förklaringar i avsnitten under figur 2.




Figur 2. Översiktsbild över de in- och ut-enheter som finns i kostruktionen..

#### 4.3.1 IR - ingång/input

IR-kontrollen och mottagaren på Altera-kortet används för att användaren ska kunna styra spelet.
För att ta emot IR-signaler i systemet utförs en kontroll och utläsning av data som ligger på en specifik adress i minnet. Detta sker kontinuerligt för att inte missa några knapp-tryck från spelaren.
Denna komponent har lagts som en IP-komponent/SW-drivrutin i BSP.

#### 4.3.2 VGA - utgång/output

VGA-skärmen används för att presentera spelet för spelaren.
Denna komponent har lagts som en IP-komponent/SW-drivrutin i BSP.

#### 4.3.3 Display - utgång/output

LCD-displayen på Altera-kortet används för att presentera poängen och meddelanden för spelaren.
Den uppdateras löpande med aktuell poäng och när spelet avslutas; vid "Game Over".
Denna komponent har lagts som en IP-komponent/SW-drivrutin i BSP.

#### 4.3.4 RAM - utgång/output & ingång/input

Tillsammans med RAM-minnet använder NIOS II processorn flera andra resurser på Altera-kortet som utförs utan direkt vetskap av systemet/spelet. Men den huvidsakliga och mest uppenbara resursen är RAM-minnet.

### 4.4 Minneskarta

Nedan visas en minneskarta från BSP:n. Ingen förändring är genomförd i inställningarna för hur olika delar i systemet sparas i minnet.

| Slave Descriptor | Address Range | Size | Attributes |
|---|---|---|---|
| TERASIC_SRAM_0 | 0x00200000 - 0x002FFFFF | 1048576 | memory |
| character_lcd | 0x00101060 - 0x00101061 | 2 | |
| jtag_uart | 0x00101058 - 0x0010105F | 8 | printable |
| sysid | 0x00101050 - 0x00101057 | 8 | |
| pio_in_key | 0x00101040 - 0x0010104F | 16 | |
| pio_out_led | 0x00101030 - 0x0010103F | 16 | |
| MY_TIMER | 0x00101020 - 0x0010102F | 16 | |
| pio_0 | 0x00101010 - 0x0010101F | 16 | |
| pio_IR_8_bit | 0x00101000 - 0x0010100F | 16 | |
| onchip_ram | 0x000C0000 - 0x000EFFFF | 196608 | memory |
| IP_VGA_0 | 0x00000000 - 0x0007FFFF | 524288 | |

Tabell 2. Minneskarta över de olika delarna i BSP:n.

## 4.5 Inställningar & setup

Inställningarna för NIOS II BSP har ändrats enligt nedan:

- ingen support finns för C++
- reducerade enhetsdrivrutiner
- litet C bibliotek används

I övrigt har ingen ändring gjorts av inställningarna i Quartus, Eclipse eller andra verktyg.

# 5. Test

## 5.1 Testprotokoll

Nedan i tabell 3 finns testfallen och resultatet av validering.

Tabell 3. Testprotokoll med testfall för validering.

| Testfall | Beskriving | Förväntat resultat | Resultat |
|---|---|---|---|
| 1 | Kör igång/exekvera spelet genom att välja "run as Nios II Hardware" för projektet i Eclipse. | Stjärnor visas utspridda över skärmen. | OK |
| 2 | Starta spelet genom att trycka knapp 5 på IR-kontrollen. | Masken visas och börjar röra sig uppåt på skärmen. | OK |
| 3 | Tryck på 6:an för att svänga masken till höger (innan den når övre "väggen"). | Masken svänger till höger. | OK |
| 4 | Kontrollera att poängen visas i nummerdisplayen på kortet. | Poängen visas i nummerdisplayen på kortet. | OK |
| 5 | Tryck på 8:an för att svänga masken nedåt (innan den når den högra "väggen"). | Masken svänger nedåt. | OK |
| 6 | Sväng åt valfritt håll och kontrollera att poängen ökar direkt efter att masken svängt. | Poängen ökar. | OK |
| 7 | Kontrollera att masken ökar i längd och hastighet ju längre spelet fortskrider. | Masken blir längre och rör sig fortare. | OK |
| 8 | Styr masken så den kraschar in i en "vägg" eller stjärna. | Spelet stannar och "Game Over" visas i LCD-displayen tillsammans med den totala poängen. | OK |
| 9 | Starta om spelet genom att trycka på 5:an på IR-kontrollen. | Skärmen rensas, nya stjärnor ritas ut och masken börjar röra sig uppåt på skärmen. | OK |
| 10 | Spela spelet så länge du kan. | Spelet pågår till du kraschar in i en vägg eller strjärna. | OK |

## 5.2 Validering

Resultatet av valideringen redovisas i testprotokollet i tabell 3 ovan.

### 5.2.1 Video

En video med en genomgång av spelet finns även på YouTube [1].

## 5.3 Tidsanalys

För att kontrollera vissa tider i konstruktionen har en tidsanalys genomförts av funktionen som rensar skärmen mellan spelomgångarna och funktionen som hanterar logiken under spelets gång då spelaren aktivt spelar spelet.

### 5.3.1 Rensning av skärm

Funktionen som rensar skärmen kontrollerades då den upplevdes långsam i förhållande till andra delar i konstruktionen. Den tar ca 6 sekunder att exekvera. Möjliga förbättringsåtgärder kommer vidtas i version 2 av spelet.

### 5.3.2 Spelmotor

Funktionen som hanterar spelet när det aktivt spelas kontrollerades i avseendet att mäta hur mycket "ledig" tid vi har i systemet vid det akuella tillfället. Det finns när spelet startar 16 millisekunder mellan varje förflyttning av masken. Denna tid minskar ju längre spelet fortgår eftersom masken ökar i hastighet. Tiden kan utnyttjas till t ex ny funktionalitet i nästkommande version av spelet.

# 6. Slutsatser

Som slutsats kan konstateras att projektet kunde levereras enligt plan med vissa smärre ändringar av krav och funktionalitet vilket redovisats i avsnitt 3.2.

## 6.1 Förbättringsförslag

Tester kommer att genomföras för att undersöka om rensningen av skärmen kan ske på ett mer effektivt och snabbt sätt för att minska väntetiden för spelaren.

# 7. Referenser

[1] https://www.youtube.com/watch?v=qJmL2-HwCAU

# 8. Bilagor

- SW-filer
- IP-filer
- Spelmanual: masken_starstruck_manual.pdf
- SW manual för VGA-funktioner
- Startrapporten, rapport A.
- Kort film på Youtube: https://www.youtube.com/watch?v=qJmL2-HwCAU