5 i rad

Konstruktör: Lars Högberg, TEIS AB/C-kurs, larhog@gmail.com, 2015-05-03

Sammanfattning: Detta är en teknisk rapport som beskriver mjukvaruimplementeringen av spelet "5 i rad" på experiementkortet Altera DE2-115. Spelet körs av på den mjuka processorn NIOS economy och styrs med hjälp av fyra tryckknappar på DE2-kortet. Utgångsenheter är en VGA-skärm, en LCD-skärm samt två gröna lysdioder.

Innehåll

1 Specifikation

1.1 Generella kundkrav

Kundens krav listas i Tabell 1.

Tabell 1: Kundkrav

Kundens krav	Kommentar
1. Krav på konstruktionskoden: a. ”filhuvud” i alla C-filer, b. Namn på variabler m.m. ska vara tydliga, c. Kontroll av gränsvärden ska ske, t.ex. att värdet inte ligger utanför skärmen, då ska ett felmeddelande skrivas ut, d. Koden ska vara rikligt kommenterad. e. Alla drivrutiner ska finnas i BSP f. Strukturerad kod, inte bara en applikationsfil för hela spelet	a. ✔ b. ✔ c. ✔ VGA-drivrutinen ger en felutskrift i NIOS-konsollen om något hamnar utanför. Appikationskoden tillåter ej att markören går utanför skärmen. d. ✔ e. ✔ f. ✔ se Figur 2.
2. a. Spelspecifikation. b. Testprotokoll. c. Mjukvaruarkitekturen. d. Hierarkin. e. Minneskarta. f. Alla I/O enheter och de viktiga funktionerna tydligt i ett kapitel (mycket från start rapporten). g. Testprotokollet ska minst testa ett fall från varje krav.	a. ✔ se 1.2. b. ✔ se 4.1 c. ✔ se 3. d. ✔ se Figur 2. e. ✔ se Figur 6. f. ✔ se 2. g. ✔ se 4.1
3. Tidsanalys av den färdiga koden. Svara på frågan om hur mycket ledig tid som finns i systemet. Frekvensen för NIOS är 50 MHz.	✔ se 4.2
4. Alla drivrutinerna ska finnas i komponenterna under HAL/src och inc. ”tcl” filerna ska definiera alla drivrutinerna för respektive komponent, så att de kommer med i BSP foldern (inte under applikations folder).	✔
5. En kort SW manual för VGA drivrutinerna (exempel se Sierra SW manual). Även exekveringstiden ska vara defilerad för varje funktion. Koden ska vara robust skriven.	✔
6. Beskriv hur koden har optimerats för hastighet och storlek med inställningar i verktygen och i koden. Beskriv minnets arkitektur och motivera varför vissa delar av koden/data har länkats till olika minnen.	✔ se 3.

7. Spela in en kort film på Youtube, som valideringsbevis att spelet fungerar och lägg in ”AGSTUTEIS” som ett av sökorden. För inspiration titta på äldre projekt (TEIS, AGSTU och FPGA). Filmen ska börja med en kort förklaring av spelet, därefter en översiktsbild av SW konstruktionen och till sist ett smakprov på spelet. Gör filmen lik en presentation av hela projektet och det ska hållas kort (max 3 minuter).	✔ se [1].
8. Leveransen ska innehålla tre foldrar: a. \Konstruktionsbeskrivning i. Konstruktionsrapport på ett standardiserat sätt, (Krav 3, 6) ii. En kort SW manual för VGA drivrutinerna (se krav 5) iii. SW filerna i en mapp, (Krav 1) iv. Alla IP komponenter, Timer, VGA IP med dess drivrutiner. (Krav 4) b. \Spel i. En enkel spelmanual (beskriv hur användaren ska spela spelet), (Krav 2) c. Diverse i. Startrapporten	a. ✔ i. ✔ ii. ✔ iii. ✔ iv. ✔ b. ✔ c. ✔
9. Leveransen ska ske till plattformen Itslearning. Namnet på filen ska vara ”förnamn_efternamn_C_ingenjorsjobb_b.zip”.	✔

1.2 Spelspecifikation

Spelet “fem i rad” för två spelare har skapats enligt följande kravlista. Krav 3 har reviderats något med nya färger.

1. VGA-skärmen visar spelplanen som i princip består av lagda brickor.
2. Det finns 32x24 möjliga ställen att lägga brickorna på.
3. Aktuell ruta indikeras en cirkel något mindre än brickorna. Färgen är magenta för spelare 1 och cyan för spelare 2.
4. Nya brickor får ej läggas där brickor lagts tidigare.
5. Spelet styrs av 4 tryckknapparna på DE2-kortet (upp, ner, vänster, höger).
6. 5 i rad vinner¨, (vågrätt, lodrätt eller diagonalt).
7. 2 spelare, en röd och en grön. Aktuell spelare visas på lysdioderna (LEDG0= Spelare 1, LEDG1 = Spelare 2). Vinnare indikeras genom att lysdioden för den spelaren börjar att blinka.
8. LCD visar nedräkning, varje spelare har bara en viss tid på sig att göra sitt drag.

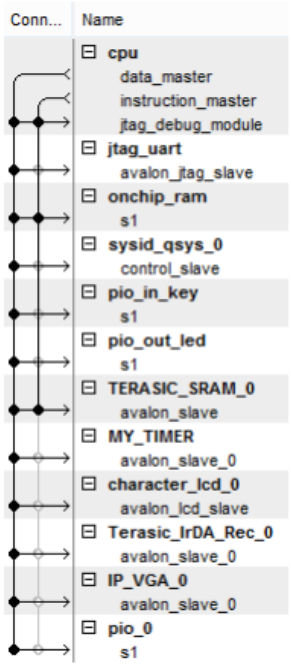
1.3 Tidplan

Genom intensitvt övertidsarbete slutfördes projektet på halva tiden. Nedan återges den ursprungliga tidplanen:

Vecka 1.	Sätt upp ett fungerande system där alla komponenter fungerar som de ska.
Vecka 2.	Programmera själva spelet.
Vecka 3.	Test och dokumenation.

2 Hårdvara

Projektet använder en harvard-arkitektur, d.v.s. en databuss och en instruktionsbuss. CPU:n är NIOS economy som körs i 50 MHz. Se Figur 1. Komponenterna beskrivs i Tabell 2.



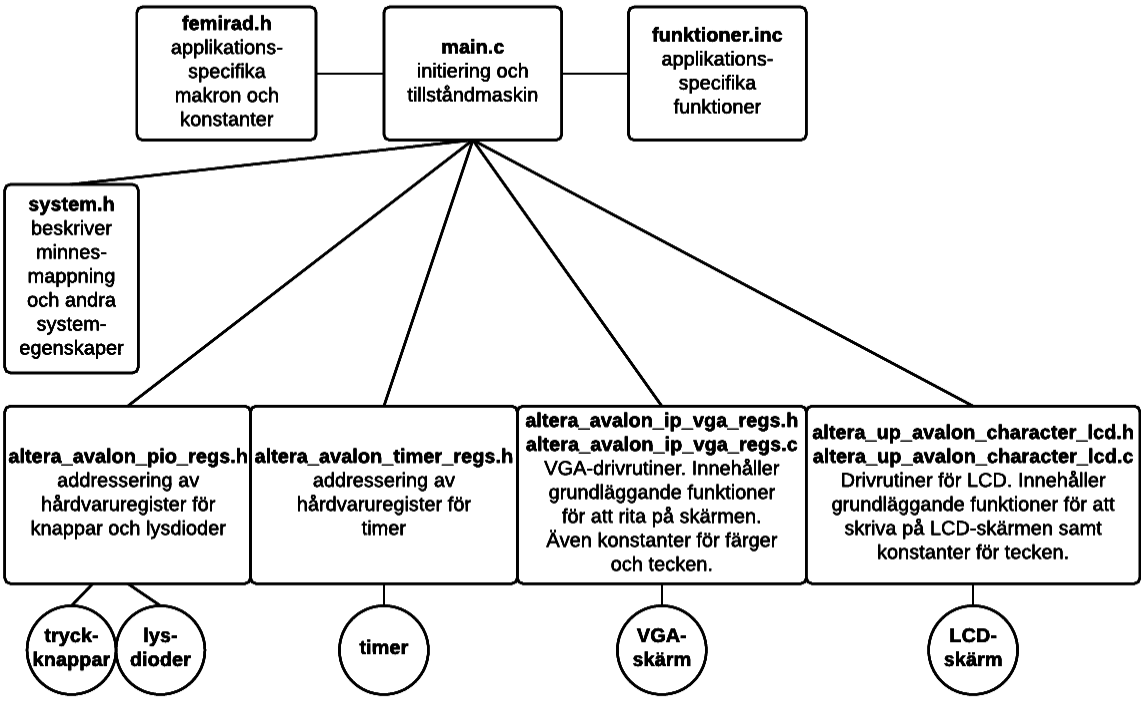
Figur 1: Arkitektur med bussar. [3]

Tabell 2: Beskrivning av mikrodatorarkitekturen, baserat på Nios II Memory Map från summary.html. Komponenter med *kursiv stil* används ej i detta projekt.

Slavbeskrivning	Adressområde	Beskrivning
TERASIC_SRAM_0	*0x00200000 - 0x002FFFFF*	*1MB externt SRAM. Används ej i detta projekt.*
character_ lcd	0x00101060 - 0x00101061	Alteras IP-kärna som är gjord för att kommunicera med 16x2-raders LCD-displayen på DE2-kortet.
jtag_uart	0x00101058 - 0x0010105F	För kommunikation med DE2-kortet via USB.
sysid	0x00101050 - 0x00101057	Hårdvarans ID-nummer
pio_in_key	0x00101040 - 0x0010104F	De fyra tryck-knappar på DE2-kortet
pio_out_led	0x00101030 - 0x0010103F	Fyra gröna lysdioder på DE2-kortet
MY_TIMER	0x00101020 - 0x0010102F	En enkel 32-bitars räknare. Räknar klocktik, har ingångar för start, stop och reset.
pio_0	0x00101010 - 0x0010101F	Pio_0 är en bit kopplad till VGA_SYNC_N för att kunna skriva till VGA-minnet när VGA-kontrollenheten inte läser från minnet, då undviks blinkande på skärmen.
pio_IR_8_bit	*0x00101000 - 0x0010100F*	*IP komponent för att kommunicera med IR-komponenten på DE2. Används ej i detta projekt.*
onchip_ram	0x000C0000 - 0x000EFFFF	191 kB CPU-nära minne.
IP_VGA_0	0x00000000 - 0x0007FFFF	En enkel 640x480 VGA grafikkontroller. Den stöder endast upplösningen 320x240 i 8 färger (superpixlar).

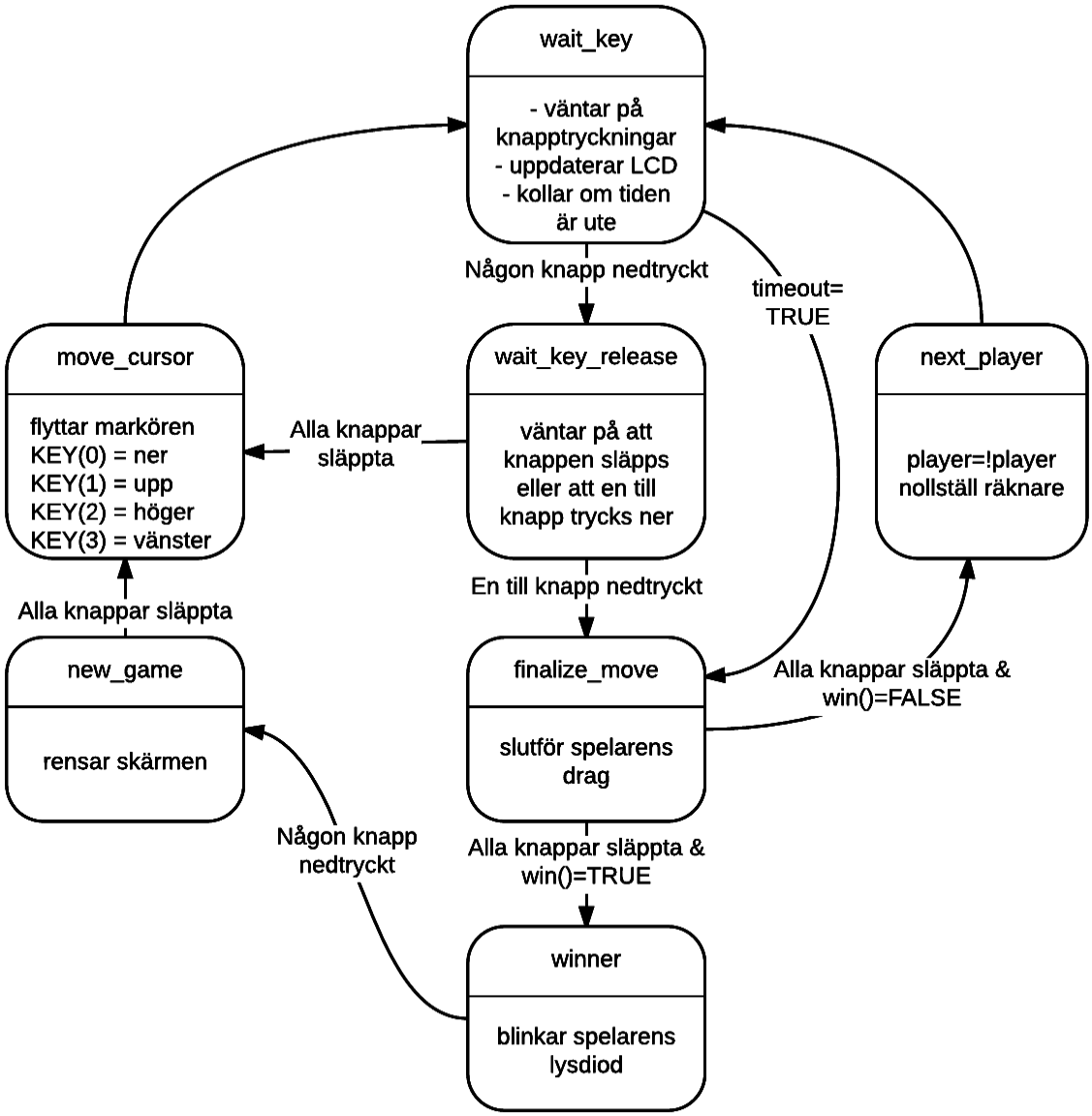
3 Mjukvara

Applikationen finns i *main.c* som använder header-filen *femirad.h*. Alla funktioner är lagrade i en separat fil *funktioner.inc*. I Figur 2 beskrivs applikationsfilerna samt de filer i board support package som ligger närmast applikationsmjukvaran.



Figur 2: Mjukvaruhierarkin. Fyrkanter och cirklar indikerar mjukvara respektive hårdvara.

main.c börjar med att initiera LCD-kommunikation och att rita upp startskärmen. Programmet väntar sedan på att spelaren trycker ned någon av tryckknapparna innan tillståndsmaskinen startas, se Figur 3.

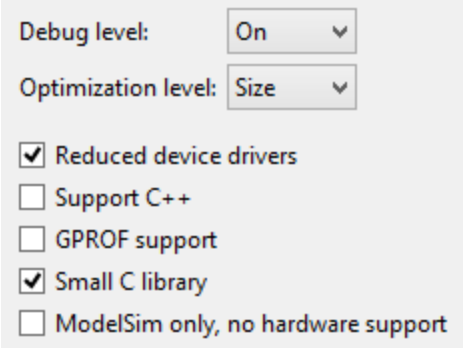


Figur 3: Tillståndsmaskinen i spelet.

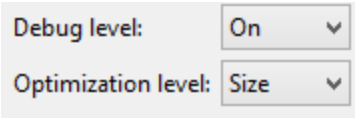
För minimera exekveringstiden har multiplikationer och divisioner använts sparsamt. I de få fall det behövts har operationen lagts på ställen som exekveras sällan. Division av timer-värde för att beräkna tiden i sekunder görs t.ex. enbart innan utskrift på LCD, en gång i sekunden. Variabeltyper är valda för att undvika “type casting” så mycket det går.

För att minimera minnesanvändningen lagras information om spelarnas markeringar i ett 32-bitars ord per rad. Varje bit representerar en kolumn och eftersom spelplanen har 32 kolumner går inget minne till spillo.

För att få ner minnesanvändningen ytterligare justerades kompileringsinställningarna. Med inaktivering av *Support C++* samt aktivering av *Optimization level (Size), Reduced device drivers* och *Small C library* blev storleken på koden endast 13 kB se Figur 4, Figur 5 och Tabell 3.



Figur 4: Kompileringsinställningar för board support package.



Figur 5: Kompileringsinställningar för applikationen.

Tabell 3: Storlek på laddfilen från kommandot *nios2-elf-size*.

text	data	bss	dec	hex	filename
12260	768	16	13044	32f4	femirad.elf

Den låga minnesanvändningen gjorde att all kod och data kunde allokeras till det snabbaste minnet; *onchip_ram*. Se Figur 6.

Linker Section Name	Linker Region Name
.bss	onchip_ram
.entry	reset
.exceptions	onchip_ram
.heap	onchip_ram
.rodata	onchip_ram
.rwdata	onchip_ram
.stack	onchip_ram
.text	onchip_ram

Figur 6: Minnesallokering.

4 Resultat

4.1 Validering

Alla testfall i valideringen slutfördes med godkänt resultat, se Tabell 4.

Tabell 4: Testprotokoll. Varje test relaterar till samma nummer i spelspecifikationen.

Test	Beskrivning	Förväntat utfall	Godkänt, ev. anm
1	Kontrollera spelplanen	Spelplanen visas korrekt med godtagbart eller inget flimmer	✔
2	Placering av brickor	Det finns 32x24 möjliga ställen att lägga brickorna på	✔
3	Aktuell ruta	Aktuell ruta indikeras en cirkel något mindre än brickorna. Färgen är magenta för spelare 1 och cyan för spelare 2.	✔
4	Försök lägga en bricka på en upptagen plats	Ej tillåtet -> inget händer	✔, spelet går över till nästa spelare
5	Tryckknappar, testa samtliga	Markören kan flyttas i riktningarna upp, ner, vänster och höger med separata tryckknappar.	✔
6	Vinst	Första spelare med 5 i rad vinner och spelet avslutas	✔ , nytt spel startas vid knapptryckning
7	Aktuell spelare	Aktuell spelare visas på lysdioderna (LEDG0= Spelare 1, LEDG1 = Spelare 2). Vinnare indikeras genom att lysdioden för den spelaren börjar att blinka.	✔
8	LCD	LCD visar nedräkning, när tiden är ute läggs brickan där markören råkar befinna sig.	✔ , om det redan finns en bricka, går spelet bara över till nästa spelare

4.2 Tidsanalays

Spelet är av sin natur inte tidskritiskt. Men det finns några frågor som kan vara intressant att studera:

- Hur lång tid tar uppstart av spelet?
- Hur lång tid tar det att flytta markören?
- Hur mycket ledig tid finns i systemet?

För att svara på dessa frågor har exekveringstiderna för olika delar av koden uppmätts med hjälp av den hårdvarutimer som finns i systemet. En temporär version av applikationen kompilerades med start och stop av denna timer på utvalda ställen. Eftersom spelet använder timern för att räkna tiden är metoden inte ideal, men med ett genomtänkt utförande fungerade den i praktiken utan problem.

Hur lång tid tar uppstart av spelet?
Att rita upp startskärmen tar 57669080 klockcykler = 1,15 s. Att från startskärmen starta spelet, d.v.s. tiden från att användaren trycker på en knapp till att spelplanen är uppritad uppmättes till 66836007 klockcykler = 1,34 s. Om användaren trycker ned en knapp redan innan startskärmen är uppritad tar alltså uppstarten ungefär 2.5 s.

Hur lång tid tar det att flytta markören?
Om det tar för lång tid att flytta markören kommer spelet att upplevas som långsamt. En människa börjar uppleva responstiden som störande runt 200 ms [2]. I Tabell 5 beräknas den totala tidsåtgången till 58 ms + *x*, där *x* är tiden det tar för spelaren att släppa knappen. *x* är sannolikt mycket större än 58 ms, vilket visar att spelet rent tekniskt är tillräckligt snabbt men eftersom spelet inte reagerar förrän spelaren har släppt knappen kan det ändå upplevas långsamt. Lösningen på detta kan vara att använda flera tryckknappar, t.ex. med hjälp av IR-fjärrkontrollen till DE2-kortet. Då skulle inte spelet behöva använda samma knappar för att flytta markören och att slutföra ett drag.

Tabell 5: Beräkning av tidsåtgång för att flytta markören

Händelse	**Tidsåtgång**
usleep(10000) (antaget worst case, d.v.s. att spelaren trycker ner knappen precis efter inläsning av tryckknappen)	704235 = 14.1 ms
wait_key ➔ *wait_key_release*	306 cykler = 6 µs
usleep(10000)	704235 = 14.1 ms
Tidsåtgång för spelaren att släppa knappen	*x*
usleep(10000) (antaget worst case, d.v.s. att spelaren släpper knappen precis efter inläsning av tryckknappen)	704235 = 14.1 ms
wait_key_release ➔ *move_cursor*	198 cykler = 4 µs
usleep(10000)	704235 cykler = 14.1 ms
move_cursor	91482 cykler = 1.8 ms
SUMMA	**58 ms + *x***

Hur mycket ledig tid finns i systemet?
Större delen av tiden befinner sig spelet i tillståndet *wait_key* där spelet väntar på att spelaren ska trycka på någon av knapparna. *wait_key* exekveras på 283 klockcykler = 5,7 µs. Efter detta sätts spelet i vänteläge genom funktionsanropet *usleep(10000)* som tar 704235 klockcykler = 14,1 ms att exekvera. Varje sekund uppdateras LCD-displayen, detta tar 316614 = 6,3 ms. Antalet gånger som *wait_key* exekveras under en sekund kan då beräknas till:

(1000 ms-6.3 ms)/(14.1 ms +0.0057 ms)=70.

d.v.s på en sekund exekveras nyttig kod: 70*5,7 µs + 6,3 ms=6,7 ms.

I normalfallet befinner sig spelet alltså i vänteläge 99,4% av tiden.

5 Slutsats

Spelet fungerar enligt specifikation, det finns dock utrymme för förbättringar. Här listas några förbättringsförslag:

- Använd IR-fjärrkontrollen för att styra spelet.
- Använd interruptbaserad inläsning av tryckknapparna.
- Räkna antal vinster för varje spelare och visa detta på den andra raden av LCD-displayen.

6 Referenser

[1] 5 i rad, https://youtu.be/jS14sQdPkz8 , uppladdad 2015-05-03.
[2] Console Gaming: The Lag Factor, http://www.eurogamer.net/articles/digitalfoundry-lag-factor-article?page=2, hämtad 2015-05-02.