Digital konstruktion
Grupp 8
Digitalt vattenpass

av

Michael Bodin 820830-1930 bodin@student.chalmers.se

Johan Böhlin 840319-7117 johboh@student.chalmers.se

Patrik Opacic 821215-5637 opacic@student.chalmers.se

Peiman Khorramshahi 810503-0756 e0peiman@student.chalmers.se

Innehållsförteckning

2. Sammanfattning

Vi har konstruerat ett digitalt vattenpass som mäter lutningsvinkeln i två axlar och visar det på en display. Konstruktionen består av fem integrerade kretsar plus en programmerbar krets. Vattenpasset visar korrekta värden mellan -10 till +10 grader, med en decimal noggranhet.

3. Inledning

Vår uppgift var att konstruera ett digitalt vattenpass med hjälp av en CPLD (programmerbar krets) och andra integrerade kretsar. Syftet med uppgiften var att ge praktiska kunskaper och färdigheter i att använda digitala integrerade kretsar, att arbeta med begränsade resurser och att gå från en specifikation till en fungerande krets. Det var fritt för projektgruppen lösa uppgiften som de ville såvida följande krav uppfylldes:

- 4-12 IC-kretsar skulle användas
- Det skulle finnas en kontrollenhet och dataväg
- Kontrollenheten skulle köras i minst 1 MHz
- Programmerbar logik skulle användas

Arbetet utfördes i en grupp om fyra teknologer och löpte under sex veckor. Projektmedlemmarna hade tillgång till ett labbrum där en dator, kopplingsdäck, oscilloskop, komponenter, logikanalysatorer och annan utrustning fanns tillgänglig. På datorn fanns också utvecklingsverktyget Xilinx installerat. I Xilinx kunde man skapa tillståndsgrafer och kopplingsscheman som programmet sedan använde för att programmera vår CPLD. Vidare fanns det möjlighet att simulera hela CPLD:n eller enskilda delar, dock så fungerade denna funktion inte helt felfritt.

Denna rapport beskriver projektgruppens arbete. Rapporten fungerar som systemspecifikation och kan också användas som en manual. Tanken är att den ska kunna läsas av både användare och utvecklare. Rapporten börjar med att ge en översiktlig beskrivning av det digitala vattenpassets konstruktion. Denna del beskriver i stora drag hur systemet fungerar, vilka komponenter som används samt hur konstruktionen designats. Vidare i rapporten beskrivs de olika delarna i konstruktionen mer ingående. Först och främst beskrivs CPLD:n och dess kopplingsschema. Vidare beskrivs i detalj funktionen hos de andra komponenter vi använt oss av i konstruktionen. Slutligen presenteras en felanalys av systemet samt vilka slutsatser som dragits.

4. Systemspecifikation

Gradmätning: -10 till 10 grader (i X- och Y-led).

16 teckensdisplay (med kontrastinställningsmöjligheter) där båda lutningsvinklarna skrivs ut med en decimal.

Resetknapp som nollställer systemet.

5. Manual

Anslut en stabiliserad 5V strömkälla till kopplingsplinten på kretskortet. Se till att ansluta 5V och jord rätt så du inte polvänder. När strömmen är ansluten tryck på den röda reset knappen för att starta vattenpasset. Displayen bör nu visa tecken. Ser du inga tecken eller om det är väldigt mörkt bör du justera konstrasten på displayen. Kontrasten justeras genom att med hjälp av en liten skruvmejsel vrida på den blåa potentiometern som är till höger om displayen. Du kan behöva justera om kontrasten om temperaturen där vattenpasset används varierar mycket, och även då betraktningsvinkeln ändras. Luta på vattenpasset för att se utslag på displayen. Om du vill låsa aktuellt värde på displayen håll inne resetknappen under tiden du vill låsa, och släpp den om du vill återgå till normalt läge.

Felsökning

- *Varför går kontrasten inte att justera in?*
- Strömförsörjningen kan vara för dålig. Byt batteri eller se över strömkällan och försök igen.

- *Varför visar displayen bara grader upp till 38 grader?*
- Det är på grund av en hårdvarubegränsning hos accelerometern.

- *Varför hänger min hylla snett efter jag har använt vattenpasset för att bygga den?*
- Det kan vara för att du har försökt mäta vinklar större än 10 grader där exakthet inte kan garanteras.

6. Delblock

Här följer en övergripande beskrivning av systemet:

- CPLD:n tar emot en puls från accelerometern och aktiverar räknarna.
- Räknarna räknar pulsens längd och lägger ”längden” som adress till minnet.
- I minnet finns en tabell med gradtal motsvarande varje ”längd”.
- CPLD:n skriver gradtalet på displayen.

Se Appendix A.1 sida 14 för blockschema.

6.1 CPLD

Översikt

CPLD:n innehåller de mest avancerade funktionerna. Den består av tre huvudkomponenter. En styrenhet, som styr vattenpasset, en logik del som vi kallar för XY-logiken som hjälper till att slå fram rätt adress i minnet och en LCD-enhet som styr displayen. I CPLD:n har vi även en tvåvägsmux som vi tar emot signalerna från accelerometern. Se CPLD:ns kopplingsschema i appendix A.3 sidan 16.

Tvåvägsmuxen
Denna komponent har två ingångar från accelerometern, en ingång som pulserar efter X-lutningen och en som pulserar efter Y - lutningen. Den har även en till insignal som bestämmer om den ska välja x eller y som insignal. SIG-signal som är en utsignal, är antingen densamma som x eller som y. Valsignalen XY, kommer från styrenheten som bestämmer vilken av X eller Y signalen som skall läsas. SIG, går i sin tur in i xy-logiken.

Styrenheten
Styrenheten styr kretsen, den har tre ingångar (exkl. CLK), den börjar med att lägga ut en XY signal som är hög, och nollställer vår räknare, genom skicka ut en resetc signal till xy-logiken.
xy-logiken styr vår externa räknare, som genererar en DONEC=1 till styrenheten då SIG (som är hög vid detta tillfälle) går låg igen. När styrenheten får DONEC, skickar den ut en LCDW signal till LCDtest som aktiverar LCD för skrivning och skrivet det som skall stå på displayen. När LCDtest är klar skickar den enheten en klar-signal, LCDDONE, till styrenheten som då inverterar värdet på sin xy-utgång (så att andra axeln väljes för avläsning).

XY-logiken
XY-logiken är en logikdel som ger en intern klockpuls, ut till två externa räknare, som i sin tur pekar på vilken minnesadress som skall läsas av LCDtest. Vad denna enhet gör i kort är att pulsa så länge SIG-signalen från tvåvägsmuxen är hög, så man får ett värde i räknaren på hur lång SIG-pulsen var och därmed kommer pekaren så snart den har värdet på hur lång SIG-pulsen var, att peka på det vinkelvärde i minnet som SIG-pulsens längd motsvarar.

I detalj fungerar det så att vid reset kommer XY-logiken att sätta sig i vänteläge för att vänta på att en SIG-puls ska bli låg (den ska inte pulsa om SIG råkar vara hög då den startar, utan den väntar på en ny SIG-puls). När den väl blir låg sätts enheten i ett annat vänteläge där den väntar på att SIG-pulsen ska bli hög igen (alltså början på en ny SIG-puls). När SIG väl blir hög igen börjar XY-logiken pulsera ut ur sin PULSE_OUT utgång, pulserna som kommer ut där så länge SIG innehar halva klockfrekvensen 0.5 MHz. PULSE_OUT går till externa räknaren som börjar räkna pulserna från XY-logiken, När SIG-pulsen går låg igen, slutar enheten att pulsa, och skickar iväg en signal till styrenheten att räknaren är klar och pekar på rätt adress i minnet. När styrenheten fått signal (DONEC) från xy-logiken att räknarna har räknat klart, skickar styrenheten ut LCDW till lcdstyrenheten för att tala om att den ska skriva till displayen. När displayutskriften är klar (LCDDONE) skickas en resetsignal (RESETC) som nollställer räknarna och börjar om med att räkna ytterligare en SIG-puls.

LCDtest – Styrenhet för LCD
Denna enhet driver displayen i konstruktionen. Den består av 30 olika tillstånd och har två ingångar och en databuss (DATA_IN) på åtta ingångar. Den har även DATA_UT på åtta utgångar. DATA_UT och DATA_IN bildar tillsammans DATA_BI som är en kombinatorisk ut- och ingång. Enheten har två utgångar (ENABLE och RS) som går till resp. ingångar på displayen. Den har två utgångar för att styra minnet B2 som bestämmer om vi ska hämta tecken och första siffran eller andra och tredje siffran från minnet (se mer information i avsnitt 6.5 om hur datan lagras i minnet). Vi har också OEM som aktiverar minnesdressen vi vill läsa från och lägger ut innehållet på bussen.

I kort fungerar LCDtest som så att vid RESET kommer enheten börja med att initiera displayen. Interface, antal tecken som kan visas samtidigt på displayen, markör och markörblink ställs in. Efter det kommer den att vänta på klarsignal, för skrivning, från styrenheten.

När den har fått klartecken kommer den undersöka vilket värde som ligger på XY ingången och antingen skriva "X:" på vänstra delen av displayen eller "Y:" på högra delen av displayen. Efter det kommer enheten att läsa av första värdet från minnet (dvs. tecknet och första gradtalet). Den skriver ut "+" eller "-", beroende av vad fjärde biten är i datavärdet. Sen omvandlar den första talet

(som kommer i binärform från minnet) till ascii-tecken, och skriver ut det. När den har skrivit klart första tecknet aktiveras B2 och adressen där andra och tredje talet pekas ut. Andra värdet hämtas, konverteras och skrivs ut på displayen. Efter det skrivs ”,”. Sedan läses andra byten in från minnet, tredje talet plockas ut, konverteras och skrivs ut på displayen. Nu har enheten gått ett varv i sin interna loop och hoppar tillbaka till det tillstånd där den väntar på klartecken från styrenheten.

Se förenklad LCDTest i appendix A.6.

Ytan på CPLD

CPLD:n har sammanlagt fyra stycken insignaler (inkluderat CLK och RESET). Utöver CLK och RESET är det X och Y signalerna från accelerometern som går in i den. Ut kommer sju signaler. E (Enable) som klockar in data i displayen, RS (Register select) som styr om data som skickas är en instruktion eller tecken. Vi har också PULSE_OUT som pulserar till räknaren (hur, står under XY-logiken). Sen har vi en CLEAR_COUNTER och en CLEAR_COUNTER2 som nollställer fjorton-respektive fyrabitarsräknarna efter varje gång LCDtest är klar. Vi har också B2 som bestämmer på vilken halva av minnet vi ska läsa på och OEM som aktiverar minnet. Sedan finns det åtta i/o signaler, som är databussen mellan CPLD:n och själva displayen. Hur signalerna är placerade på CPLD:n syns på bilden under.

I/O Namn	***I/O funktion***	***Pinne***
Y	Input	P4
X	Input	P3
RS	Output	P2
REAL_RESET	Input	P39
PULSE_OUT	Output	P7
OEM	Output	P44
E	Output	P43
DATA_bi<7>	Input/Output	P28
DATA_bi<6>	Input/Output	P29
DATA_bi<5>	Input/Output	P33
DATA_bi<4>	Input/Output	P34
DATA_bi<3>	Input/Output	P35
DATA_bi<2>	Input/Output	P36
DATA_bi<1>	Input/Output	P37
DATA_bi<0>	Input/Output	P38
CLK	Input	P5
CLEAR_COUNTER2	Output	P6
CLEAR_COUNTER	Output	P9
B2	Output	P42

6.1.1 Datavägen
Vi har en åttabitars dataväg i vår konstruktion. CPLD:n använder bussen till för att läsa från minnet och för att skriva till displayen. Displayen håller aldrig bussen då displayen bara läser data. Minnet lägger ut data på bussen först efter att signalen OEM aktiverats från CPLD:n. Detta gör att bussen kan delas mellan displayen, minnet och CPLD:n.

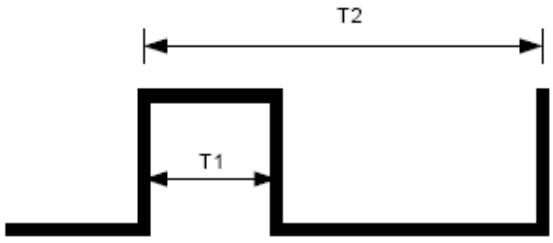
6.2 Displayen – LCD
För att visa vattenpassets lutning användes en LCD-display på 16 tecken × 1 rad, ASCII. En 8-bitars databuss (DB0-DB7) används för att skicka instruktioner och skriva tecken till LCD:n vilket gör gränssnittet enkelt och praktiskt. Insignalen RS indikerar om det är data som ska läsas/skickas (RS hög) eller en instruktion som ska skrivas (RS låg).

Innan man kan börja skriva tecken på displayen måste den initieras. Här följer en beskrivning av själva initieringsfasen. Under de första 15ms efter att strömmen slagits på (startup-tiden) kan ingen data skickas eller tas emot från displayen. När displayen är igång är nästa steg i initieringen att skicka fyra instruktioner i följd: *Display Clear*, *Function Set, Display ON/OFF*, *Entry Mode Set*. *Display Clear* rensar displayen samt sätter teckenmarkören (som indikerar var tecken skrivs) till ursprungspositionen. *Function Set* sätter vilket interface som ska användas (8/4-bitar) och om en eller två rader ska användas. *Display ON/OFF* aktiverar displayen samt konfigurerar markören. *Entry Mode Set* sätter huruvida displayen skiftar till höger eller vänster när tecken skrivs och om markören ska flyttas fram ett steg när ett tecken skrivs eller inte. Varje instruktion tar en stund att utföra (*Execution Time*) och om man inte väntar den tiden innan man skickar nästa instruktion kan displayen sluta fungera.

För att skriva tecken på displayen ska insignalen RS sättas hög och en ASCII-kod, som motsvarar det tecken man vill skriva, ska läggas på databussen.

6.3 Accelerometern
är en dubbelaxlad accelerometer som använder sig av gravitationskraften för att mäta lutningsvinkeln på två olika axlar.

Accelerometerns två utsignaler är definierad som följande:



där formeln för att räkna ut accelerationen är A (g) = (T1/T2 – 0.5) / 0.2 och där A är mätt i G. För att omvandla till grader tar man arcsin på accelerationen. T2 är ett fast värde, i vårt fall 10ms. Pulsbredden T1 ändras då lutningsvinkeln ökar eller minskar och ligger mellan 3 och 7 milisekunder. 3 ms är vid – 1 g eller -90 grader, 5 ms när accelerationen är noll dvs 0 grader, och motsvarande då 7 ms vid +90 grader. Accelerometern är bara exakt runt noll accelerationen och ger inte alls korrekta värden långt från noll. Tanken är att man skall använda accelerometerns dubbla axlar för att mäta en lutning, för att på så vis få noggranhet nära noll och nära 1 g. Vi har dock använt den för att mäta lutningen i två plan, sett som ett XY diagram.

6.4 Räknarna

Vi använder oss utav två stycken binära räknare, en fyra- och en fjortonbitsräknare. Dessa räknare har binära utgångar där motsvarande uppräknade värde kan avläsas hela tiden. Minnet vi använder, som räknarna är kopplade till, är ett fjortonbitarsminne. Dock behöver vi bara använda de första tretton bitarna från räknarna för att adressera minnet, men fjorton-bitarsräknaren är kapslad i en 16 pinnars krets och har då bara av utrymmersskäl bara bitutgångarna 0, 3-13 så vi behöver en till räknare som täcker upp de bitar som fattas. Vi tar därför de lägsta fyra bitarna från fyra-bitarsräknaren. Räknarna har ingång för en klockpuls, vilket räknar upp räknaren varje gång en sådan kommer. Denna klockpuls styr vi när den ska komma för att räknarna inte ska räkna när vi inte vill. Fjortonbitsräkarnen har en resetingång som återställer räknaren till noll, detta saknar dock fyrabitsräknaren. Istället har den möjlighet att via fyra stycken binära ingångar ladda räknaren med ett vist värde varje gång man ger den en loadsignal. Vi laddar då räknaren med det binära värdet 0000 vilket motsvarar reset på fjortonbitsräknaren.

6.5 Minnet

Minnet är på fjorton bitar vilket motsvarar 16384 adresser med möjlighet att lagra en byte på varje adress. För att läsa ett värde i minnet lägger man ut den adressen man vill läsa på på adressingångarna och aktiverar sedan minnets CE och OE ingångar. CE betyder Chip Enable och läser in adressen på adressbussen. OE betyder Output Enable och lägger ut datan på databussen som ligger på den aktuella adressen. Minnet programmerar man genom att bränna in datan via en programmerare/brännare. Där laddar man in sin binära datafil och bränner in den. Vill man lägga in en annan data så får man först radera minnet genom att belysa det med UV-ljus en halvtimme. Se appendix för det javaprogram som genererar den binära fil motsvarande det data som ligger i minnet. På adressingången ligger ett binärt värde från räknarna som motsvarar i millisekunder den tid pulsen från accelerometern var hög. Tiden, eller värdet, är mellan 3000-7000. I minnet har vi programmerat in ett gradtal motsvarande varje ”tid” som kan ligga som adress in på minnet. Vi sparar i minnet om värdet är positivt eller negativt, och ett gradtal med två värdesiffror och en decimal. Allt detta får inte plats i en byte utan sparas i två byte. Byte ett lägger vi orginaladressen som vi adresserar direkt från räknaren, dvs 3000-7000. När vi vill läsa byte nummer två hoppar vi iväg med hjälp av den högsta adressbiten i minnet 8192 steg, halva minnets storlek. Original adressen ligger då fortfarande kvar på de lägre adressbitarna. Byte två ligger alltså i intervallet 11192-15192. Signal B2 motsvarar den högsta adressbiten i minnet och aktiveras från CPLD:n. Följande struktur sparas i minnet:

Byte ett innehåller följande: 000P FFFF.

Där P är 1 om talet är negativt och 0 om talet är positivt.

FFFF är första siffran i gradtalet.

Byte två innehåller följande: SSSS DDDD

SSSS är andra siffran i gradtalet.

DDDD är decimalsiffran i gradtalet.

Gradtalet är bcdkodat i minnet, vilket innebär att man tar varje siffra för sig och gör om det binärt. Detta på grund av att när vi sedan ska skriva ut ett gradtal på displayen, som tar emot ett asciivärde, så behöver vi bara läsa ut varje siffra och lägga till värdet 48, vilket motsvarar ”0” i ascii, för att få motsvarande asciitecken för den siffran. Antag att vi har gradtalet -37,4 som vi vill spara i vår datastruktur. Eftersom att talet är negativt ska P i byte nummer ett vara 1. Första siffran i talet är en trea, som vi skriver om på binär form: 0011.
Vi har då byte nummer ett: 00010011.
Nästa siffra är en sjua, och på binär form: 0111. Och till slut decimalsiffran: 0100.
Byte nummer två är då följande: 00110100.

7. Felanalys

Den största felkällan är accelerometern. Enligt databladet ska den ge en puls med bredden $3ms \leq T1 \leq 7ms$, men i praktiken är den $3.75ms \leq T1 \leq 6.25ms$. Vidare är accelerometerns noggrannhet störst runt 0 grader och avtar kraftigt med ökad lutningsvinkel.

Konsekvensen av accelerometerns felaktiga pulsbredd är att vi bara kan mäta inom intervallet -38 till +38 grader istället för -90 till +90 grader. Accelerometerns bristande noggranhet gör att vi endast kan visa någorlunda korrekta värden i intervallet -10 till +10 grader. I det här intervallet har vi en noggranhet på ±0.1 grader. Utanför detta intervall kan denna noggranhet inte garanteras.

Åtgärder man kan vidta för att få bättre noggranhet och större mätintervall är att:

- kompensera pulsbreddsfelet genom att göra linjär regression.
- byta accelerometer mot en med bättre precision.
- Bara mäta en lutningsvinkel med hjälp av båda axlarna.

8. Slutsats

Vattenpasset fungerar som planerat för specifikationen vi angivit. Dock brukar "vanliga" vattenpass fungera i 180 (-90 till 90) grader. Man kan öka precisionen genom att använda accelerometerns båda axlar till att bestämma vinkeln i en axel. Vi kom inte undan "trial and error"-metoden då displayen visade sig vara problematisk. I övrigt fungerade allt som det skulle och vi är nöjda med resultatet.

Appendix