|  |
| --- |
| TEIS AB |
| Avståndsmätare |
| Examensarbete TEIS 2021 |
|  |
|  |
| **Ashraf Tumah** |
| **21-07-02** |

Sammanfattning: En avståndsmätare konstruerad i tidigare uppgift ska återanvänds med en del optimeringar. Avståndsmätaren är en HC-SR05 Ultrasonic Range Sensor som monterad på en servo motor skannar av en yta i 180 grader. Data läses av ett inbäddat datorsystem som av olika värden genererar pixeldata på en monitor vilket visar var ett objekt är placerat. För servemotorn har en IP-komponent konstruerats vilket gör det möjligt att styra motorn direkt från mjukvaran. Avståndsmätaren operar i hårdvaran där distansen läses in i mjukvaran via PIO. Tidsplanen fullföljdes vilket hamnade på 160h heltidsjobb med en total kostnad på 128 000kr.  
Följande länk är ett videoklipp på youtube som beskriver och validerar konstruktionen: <https://www.youtube.com/watch?v=zPpom8koREg>

# INNEHÅLLSFÖRTECKNING

[INNEHÅLLSFÖRTECKNING 3](#_Toc31553)

[1 Inledning 5](#_Toc22080)

[2 KRAVSPECIFIKATION 6](#_Toc19087)

[3 Tidplan 7](#_Toc15917)

[4 Testprotokoll 8](#_Toc731)

[5 Konstruktionsbeskrivning HW 10](#_Toc10021)

[5.1 DE10-Lite 11](#_Toc27191)

[5.2 Konstruktionsbeskrivning HW 11](#_Toc15454)

[5.2.1 Delsystem 13](#_Toc12868)

[5.3 Servo HW IP 16](#_Toc4150)

[5.3.1 IP Komponent 16](#_Toc26979)

[6 Konstruktionsbeskrivning SW 18](#_Toc12029)

[6.1 Tasks 19](#_Toc11323)

[6.1.1 Task\_avg\_Range 19](#_Toc17936)

[6.1.2 Task\_Range 19](#_Toc4309)

[6.1.3 Task\_Range\_Print 19](#_Toc16617)

[6.1.4 Task\_Servo\_Position 19](#_Toc13301)

[6.2 Drivrutiner 19](#_Toc29608)

[6.2.1 Init\_screen 19](#_Toc16895)

[6.2.2 Pattern 19](#_Toc26329)

[6.2.3 TestFunction 19](#_Toc24604)

[6.3 Kodstorlek 20](#_Toc13964)

[7 Utrustning 21](#_Toc16261)

[7.1 HC-SR05 Ultrasonic Range Sensor 21](#_Toc18160)

[7.2 SG90 Micro Servo 22](#_Toc13336)

[8 Optimering, Settings & Assignments 24](#_Toc25368)

[8.1 Advisor 24](#_Toc12450)

[8.2 IO-pinnar 25](#_Toc16018)

[9 Verifiering/Validering 28](#_Toc24205)

[9.1 Verifiering Ultraljudssensor 28](#_Toc22765)

[9.1.1 Testfall 0-1 28](#_Toc29640)

[9.1.2 Testfall 2 28](#_Toc31349)

[9.2 Verifiering Servomotor 29](#_Toc16676)

[9.2.1 Testfall 0 30](#_Toc31647)

[9.2.2 Testfall 1 30](#_Toc1158)

[9.2.3 Testfall 2 31](#_Toc21755)

[9.2.4 Testfall 3 31](#_Toc9615)

[9.3 Resultat från validering 31](#_Toc9134)

[10 Analys 33](#_Toc30895)

[10.1 Effektförbrukning 33](#_Toc28351)

[10.2 Tidsanalys 33](#_Toc17351)

[10.2.1 Kortast Slack 33](#_Toc7096)

[11 Slutsats 35](#_Toc22713)

[12 REFERENSER 36](#_Toc13677)

# 

# Inledning

Målet med det här projektet var att kombinera en tidigare komponent utförd i VHDL2\_Systemkurs och efterlikna gränssnittet i C\_task\_14 där varje task har en ruta med dess instruktioner. Ena tasken var tänkt att i rutan visa det klassiska radarmönstret, gröna cirklar där en yta på 360grader scannas av och en punkt målas ut där ett objekt skannas. Dilemmat med en upplösning på 320\*240 och en servomotor som har en vinkel på max 180 grader är att idén inte var effektiv vilket resulterade till tidspress mot slutet av projektet.

# KRAVSPECIFIKATION

Tabell 1. Kravspecifikation

|  |  |  |
| --- | --- | --- |
| **Krav** | **Beskrivning** | **Utfört** |
| ***Konstruktionskod*** | | | |
| 1 | Som är definierat i VHDL- och C-kursen | Ja | |
| ***Dokumentationskrav och struktur på mapparna*** | | | |
| 2 | Leverans med strukturerad standardrapport   1. Rapport (**OBS** filnamn: fornamn\_efternamn\_exjobb\_B)    1. Leverans i formatet Word, svenska eller engelska.    2. Max 30 sidor | Ja | |
| 3 | Presentation av resultat; Skapa en film, max 10 minuter (lägg upp den på Youtube),   1. Introducera dig själv 2. Översikt av projektet 3. Visa HW/SW 4. Demonstrera systemet 5. Gärna slutsatser av projektet (vad var bra, problem med mera) 6. Trevligt om taggen ”TEIS” läggs till, då blir det enkelt att få fram alla filmerna från TEIS-utbildningen.   Det finns inga krav på kvalitén på filmen, men det är bra träning att presentera ett projekt som är genomfört.  **OBS! Om du inte vill att den ska publiceras, skriv det tydligt efter filmlänken.** | Ja | |
| ***Leveranskrav*** | | | |
| 4 | Slutleveransen kan innehålla följande mappar med följande dokument:   * **Mappen Dokumentation**   + Rapporter   + Fristående bilagor som kod, kretsschema och datablad * **Mappen Konstruktionsunderlag**   + Arkiverat Quartus projekt,   + IP-komponenter i en map * **Mappen Konfigureringsfiler**   + configuration.sof, pof * **Mappen Demoexempel**   + Kan vara en testarkitektur. Kan vara samma som testbenches. * **Mappen Testbenches**   + SW program för att validera konstruktionen * **Mappen diverse**   + Datablad på kretsar med mera   Leveransen ska ske till plattformen Itslearning. Namnet på filen ska vara ”fornamn\_efternamn\_exjobb.zip” (en fil). | Ja | |
| 5 | Leveransen bedöms ut följande steg (kan vara i olika ordning):   * VHDL * C * Arkitektur * Dokumentation (Tekniska rapporten)   Varje steg kan max 5 komplettering göras om inte kunden (läraren) anser något annat. Om antalet kompletteringar överstiger 5 gånger för en granskning (t.ex. VHDL), kräver kunden en ny leverans av ett nytt projekt. | Ja | |
| UK/G/VG | Betyg sätts utifrån en samlad bedömning från de som examinerar de olika stegen. |  | |

# Tidplan

Examensarbetet innefattar fyra veckors heltidsjobb vilka uppskattas ta totalt 160h med en timkostnad på 800kr. Arbetet kommer pågå under fyra veckor. Första veckan går åt förstudie och förberedande. Två veckor går åt konstruktion,verifiering validering och optimering. Under sista veckan sker en dokumentation och eventuella optimeringar.  
  
Tabellen nedan visar beräknad tidsåtgång och kostnad för olika moment genom projektet.

Tabell 2. Tidplan och kostnad

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Aktivitet** | **V1** | **V2** | **V3** | **V4** | **Totalt** |
| **Studie och Förberedande** | 30h |  |  |  | 30h |
| **Konstruktion** |  | 50h |  |  | 50h |
| **Verifiering Validering** |  |  | 20h |  | 20h |
| **Optimering** |  |  | 15h | 10h | 35h |
| **Dokumentation** | 5h |  |  | 30h | 35h |
| Summa | 35h | 50h | 40h | 40h | 160h |
| Kostnad | 28 000kr | 40 000kr | 32 000kr | 32 000kr | 128 000kr |

# Testprotokoll

Konstruktionen ska med en ultraljudssensor sända ljudvågor vilka reflekteras tillbaka och fångas upp av mottagaren då de kolliderar med ett objekt. En modul ska räkna ut tiden mellan sändarpuls och mottagar puls vilket generar avståndsvärden. För att göra konstruktionen mer fördelaktig monteras ultraljudssensorn till en servo motor vilket gör det möjligt att samla data av en yta på 180 grader. Tabell 3-5 visar testprotokoll genom olika moment.

Tabell 3. Testprotokoll Simulering Ultraljudssensor

|  |  |  |  |
| --- | --- | --- | --- |
| **Testfall** | **Beskrivning** | **Acceptans** | **Godkänt (J/N)** |
| **0** | Reset Aktiv Låg | 7-segdisplay visar streck Distans = 0 |  |
| **1** | Reset Aktiv Hög | 7-segdisplay visar avståndsvärde |  |
| **2** | Virtuell Echo Puls med ökad fördröjning för varje puls | Distansen ökar 7-segdisplay visar avståndsvärde |  |

Tabell 4. Testprotokoll Simulering Servomotor

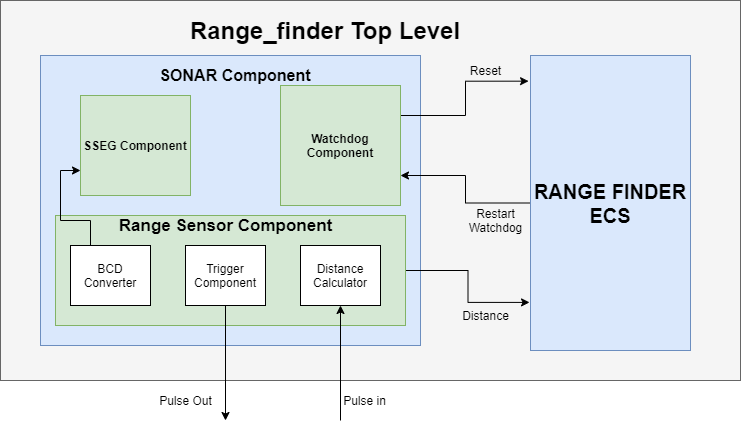
|  |  |  |  |
| --- | --- | --- | --- |
| **Testfall** | **Beskrivning** | **Acceptans** | **Godkänt (J/N)** |
| **0** | Reset=0 | Ingen position ut |  |
| **1** | Chipselect=0 Write enable=0 Reset=1 Data in=16 | Puls ut =0,5ms (Servo vinkel 0°) |  |
| **2** | Chipselect=0 Write enable=0 Reset=1 Data in=50 | Puls ut=1,5ms (Servo vinkel 90°) |  |
| **3** | Chipselect=0 Write enable=0 Reset=1 Data in=80 | Puls ut=2,5ms (Servo vinkel 180°) |  |

Tabell 5. Testprotokoll Validering

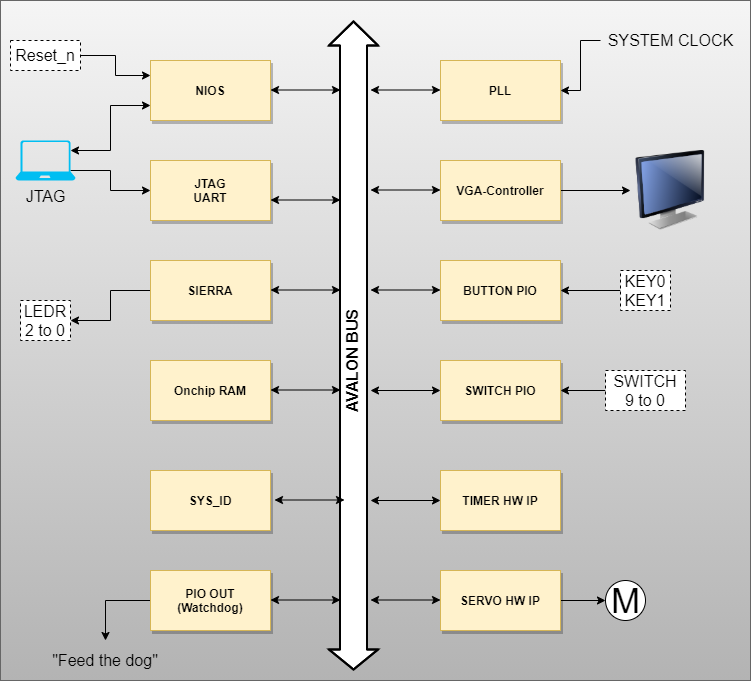
|  |  |  |  |
| --- | --- | --- | --- |
| **Testfall** | **Beskrivning** | **Acceptans** | **Godkänt (J/N)** |
| **0** | Start Skärmen | Information visas på VGA-särmen: Konstruktör, företag, Datum |  |
| **1** | Start knapp Tryck KEY1 | Informationen försvinner Mönster målas på skärmen |  |
| **2.1** | Motorposition 0 grader | Markera riktning blå cirkel Gul cirkel reagerar på visualiserar avstånd. |  |
| **2.2** | Motorposition 25 grader | Markera riktning blå cirkel Gul cirkel reagerar på visualiserar avstånd. |  |
| **2.3** | Motorposition 50 grader | Markera riktning blå cirkel Gul cirkel reagerar på visualiserar avstånd. |  |
| **2.4** | Motorposition 75 grader | Markera riktning blå cirkel Gul cirkel reagerar på visualiserar avstånd. |  |
| **2.5** | Motorposition 90 grader | Markera riktning blå cirkel Gul cirkel reagerar på visualiserar avstånd. |  |
| **2.6** | Motorposition 115 grader | Markera riktning blå cirkel Gul cirkel reagerar på visualiserar avstånd. |  |
| **2.7** | Motorposition 140 grader | Markera riktning blå cirkel Gul cirkel reagerar på visualiserar avstånd. |  |
| **2.8** | Motorposition 165 grader | Markera riktning blå cirkel Gul cirkel reagerar på visualiserar avstånd. |  |
| **2.9** | Motorposition 180 grader | Markera riktning blå cirkel Gul cirkel reagerar på visualiserar avstånd. |  |

# Konstruktionsbeskrivning HW

Systemet består av två komponenter. Ett inbäddat datorsystem och ett ultraljudsensorsystem.

* Sonar Component - Komponenten mäter avstånd i cm vilka visas på DE10-Lite kortets 7-segmentdisplayer. Komponenten innehåller även en watchdog timer vilket återställer hårdvara och mjukvara om systemet hänger sig. Figuren nedan visar anslutningar mellan komponenterna i hårdvarunivå.  
  

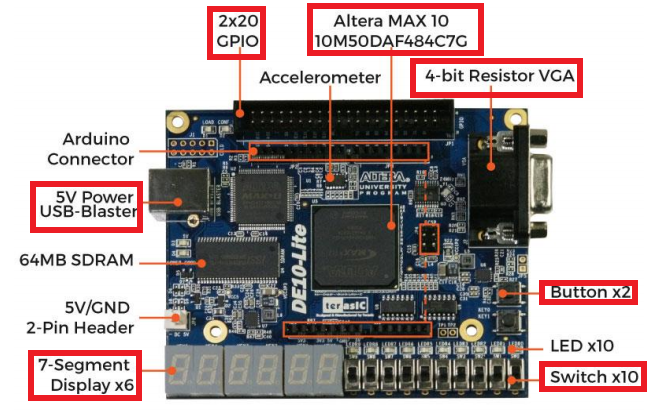
Figur 1. HW-Arkitektur

* RangeFinder ECS - Det inbäddade datorsystemet styr en servomotor samtidigt som avståndsvärden läses in vilket gör det möjligt att samla data av en yta på 180 grader. För att kontrollera servomotorn har IP-komponenten SERO HW IP skapats.  
  Figuren nedan visar komponenter anslutna mot Avalon busen.  
  

Figur 2. Systemarkitektur

## DE10-Lite

Systemet konstrueras, verifieras och valideras på utvecklingskortet DE10-Lite som ingår bland MAX 10 FPFA familjen. Hårdvara/mjukvara laddas upp till kortet via USB-blaster. Genom JTAG UART sker en kommunikation mellan kortet och en pc vilket används för att verifiera och felsöka konstruktionen.  
Figuren nedan visar kortet med de delar av kortet som ska användas.

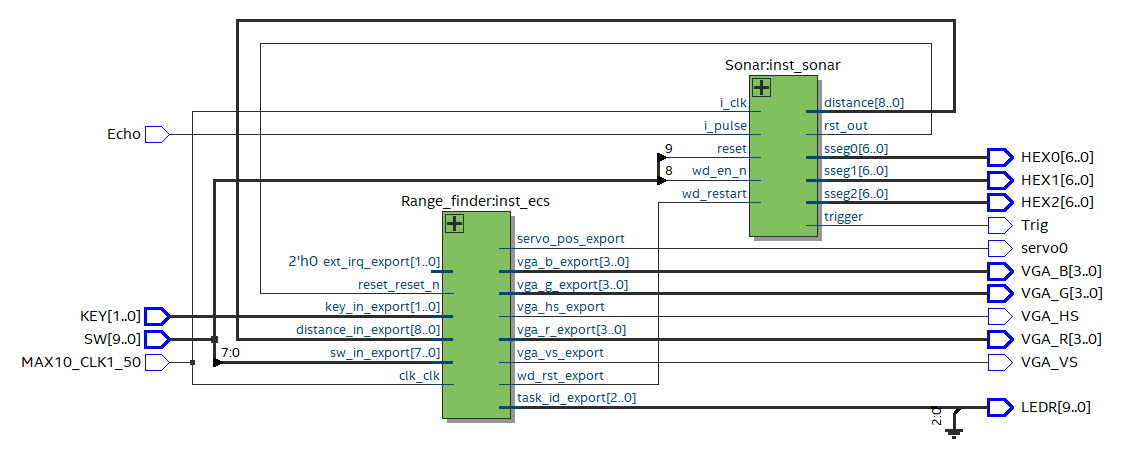


Figur 3. DE10-Lite[1]

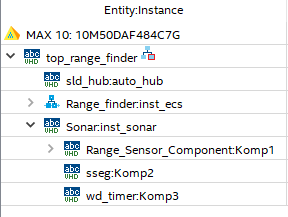
* **Altera MAX10 -** Laddar ner hårdvaran i .sof/.pof
* **4-bit Resistor VGA -** Används för att visa värden och objekt på en monitor
* **Button -** Tryckknapp för att gå vidare i systemet efter välkomstskärm
* **Switch -** För att ställa in olika tillstånd
* **7-Segment Display -** Visa avståndet i cm direkt på utvecklingskortet
* **5V Power USB-Blaster -** För att programmera och debugga systemet
* **GPIO -** För inkoppling av hårdvara(HC-SR05 och servomotor)

## Konstruktionsbeskrivning HW

Figur 4 visar systemet i RTL-nivå och figur 5 visar komponenthierarkin.

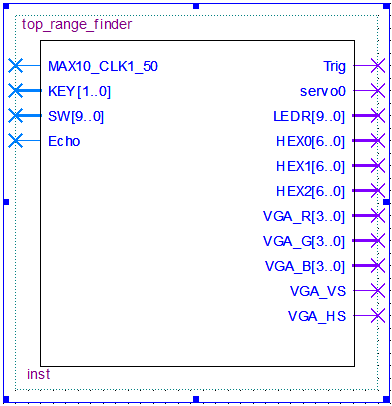


Figur 4.RTL Top-nivå



Figur 5.Komponenthierarki

Figuren nedan visar systemets in och ut signaler och tabell 6 ger en signalbeskrivning.



Figur 6. Symbolblock top-nivå

Tabell 6. Signalbeskrivning

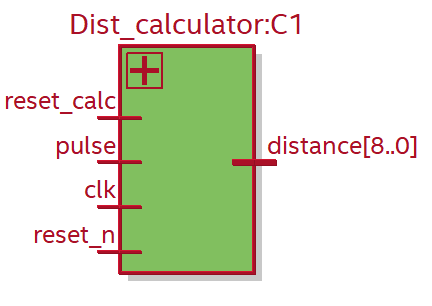
|  |  |  |  |
| --- | --- | --- | --- |
| **Namn** | **Beskrivning** | **Riktning** | **Typ** |
| MAX10\_CLK1\_50 | Systemklocka 50MHz | IN | STD\_LOGIC |
| KEY | Tryckknappar | IN | STD\_LOGIC\_VECTOR(1 downto 0) |
| SW | Switchar | IN | STD\_LOGIC\_VECTOR(9 downto 0) |
| Echo | Ultraljud puls | IN | STD\_LOGIC |
|  |  |  |  |
| Trig | Ultraljud puls | UT | STD\_LOGIC |
| Servo0 | PWM servo position | UT | STD\_LOGIC |
| LEDR | Led Röd | UT | STD\_LOGIC\_VECTOR(9 downto 0) |
| HEX0, HEX1, HEX2 | 7-segmentdisplay | UT | STD\_LOGIC\_VECTOR(6 downto 0) |
| VGA\_R | VGA Röd | UT | STD\_LOGIC\_VECTOR(3 downto 0) |
| VGA\_G | VGA Grön | UT | STD\_LOGIC\_VECTOR(3 downto 0) |
| VGA\_B | VGA Blå | UT | STD\_LOGIC\_VECTOR(3 downto 0) |
| VGA\_HS | Horisontell synk | UT | STD\_LOGIC |
| VGA\_VS | Vertikal synk | UT | STD\_LOGIC |

### Delsystem

#### Distanskalkylator

Komponentnamn: Dist\_calculator  
Instansnamn: C1

Distanskalkylatorn återställer och räknar antal klockcykler då trigger komponenten sänder en trigger puls. När mottagaren genererar en puls kalkyleras värdet av centimeter genom att omvandla de räknade cyklarna till mikrosekunder vilka divideras med 58. Figur 7 visar komponenten och tabell 7 ger en signalbeskrivning.



Figur 7. Distanskalkylator

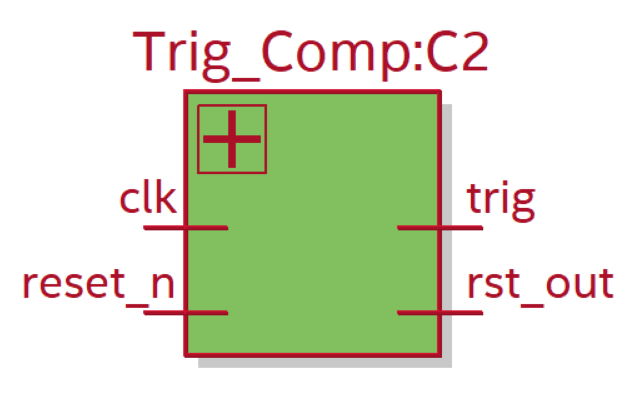
Tabell 7. Signalbeskrivning

|  |  |  |  |
| --- | --- | --- | --- |
| **Namn** | **Beskrivning** | **Riktning** | **Typ** |
| clk | Systemklocka 50Mhz | IN | STD\_LOGIC |
| reset\_n | Reset signal | IN | STD\_LOGIC |
| pulse | Ultraljud Puls | IN | STD\_LOGIC |
| reset\_calc | Reset av räknare | IN | STD\_LOGIC |
|  |  |  |  |
| distance |  | UT | STD\_LOGIC\_VECTOR |

#### Trigger Kompontent

Komponentnamn: Trig\_Comp  
Instansnamn: C2

Trig\_Comp sänder en ultraljudspuls var 250ms. För att transmittern på sensorn ska kunna sända ljudvågorna måste triggersignalen vara aktiv i minst 10 µS. Dock i den här konstruktionen är den aktiv i 100µS var 250ms. I komponenten stabiliseras även reset signalen. Trigger signalen återställer även räknaren i Distanskalkylatorn. Figur 8 visar komponenten och tabell 8 ger en signalbeskrivning.



Figur 8. Trigger komponent

Tabell 8. Signalbeskrivning

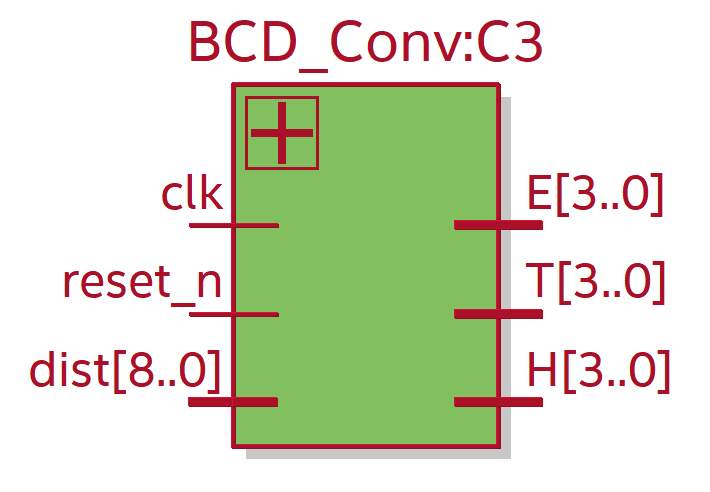
|  |  |  |  |
| --- | --- | --- | --- |
| **Namn** | **Beskrivning** | **Riktning** | **Typ** |
| clk | Systemklocka 50Mhz | IN | STD\_LOGIC |
| reset\_n | Reset signal | IN | STD\_LOGIC |
|  |  |  |  |
| rst\_out | Stabil Reset Signal | UT | STD\_LOGIC |
| trig | Ultraljud Puls | UT | STD\_LOGIC |

#### BCD Konverter

Komponentnamn: BCD\_Conv

Instansnamn: C3

BCD avkodar det binära värdet av distansen till heltal som representerar ental, tiotal och hundratal. Heltalen ansluts till en 7-segments komponent som visar värdena på 7segmentdisplay 0,1 och 2. Figur 9 visar komponenten och tabell 9 ger en signalbeskrivning.



Figur 9. BCD Konverter

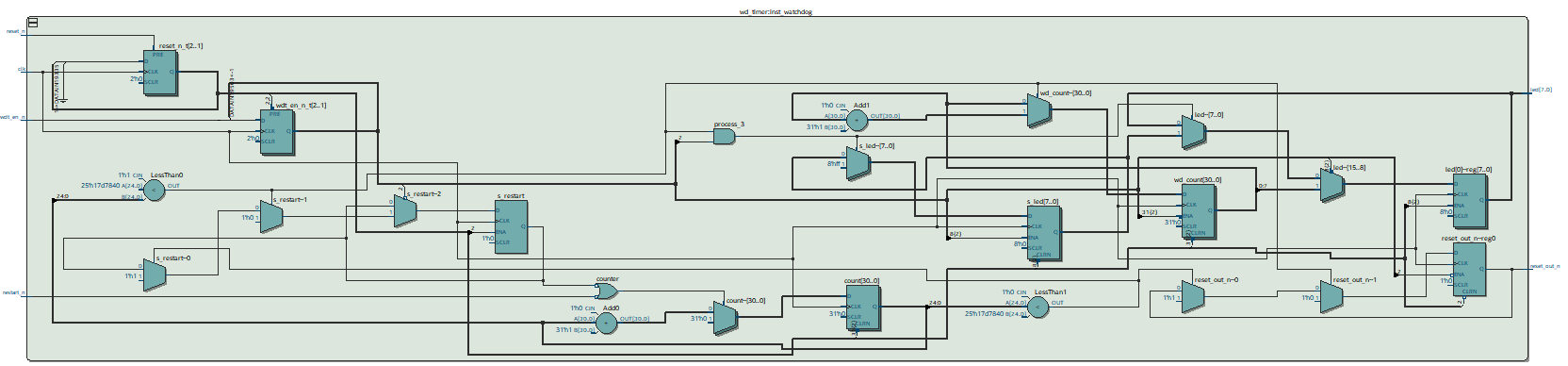
Tabell 9. Signalbeskrivning BCD\_Conv

|  |  |  |  |
| --- | --- | --- | --- |
| **Namn** | **Beskrivning** | **Riktning** | **Typ** |
| clk | Systemklocka 50Mhz | IN | STD\_LOGIC |
| reset\_n | Reset signal | IN | STD\_LOGIC |
| dist | Distansvärden 9bit | IN | STD\_LOGIC\_VEVTOR |
|  |  |  |  |
| E | Ental | UT | STD\_LOGIC\_VEVTOR |
| T | Tiotal | UT | STD\_LOGIC\_VEVTOR |
| H | Hundratal | UT | STD\_LOGIC\_VEVTOR |

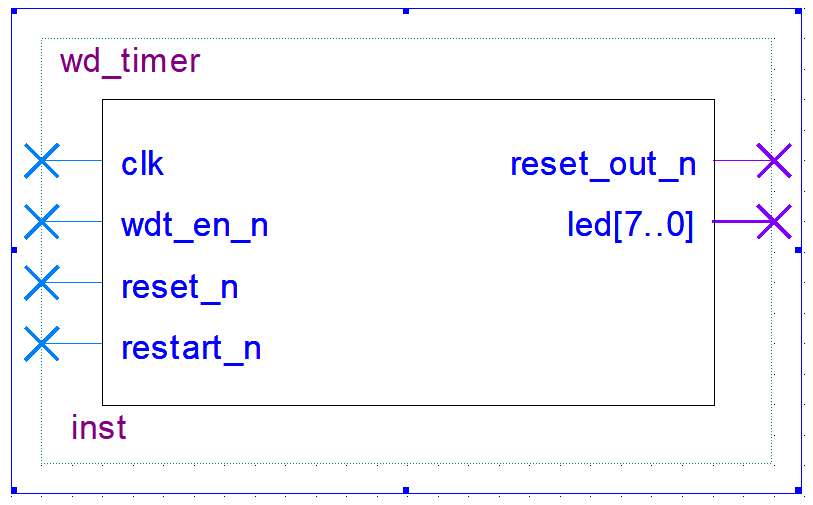
#### Watchdog Component

Komponentnamn: wd\_timer  
Instansnamn: inst\_watchdog

Komponenten aktiveras genom signalen wdt\_en\_n vilken är ansluten till switch 0. Wd\_timern räknar upp till 50 000 000 vilket med en klockfrekvens på 50MHz är en sekund. Då räknaren når den angivna tidsgränsen och komponenten är aktiv sänds en signal som är ansluten till mjukvarukomponentens reset. Figuren nedan visar komponenten i rtl-nivå, figur 11 visar komponentens symbol och tabell 10 ger en signalbeskrivning.



Figur 10. Watchdog Timer RTL



Figur 11. Watchdog Timer symbol

Tabell 10. wd\_timer signalbeskrivning

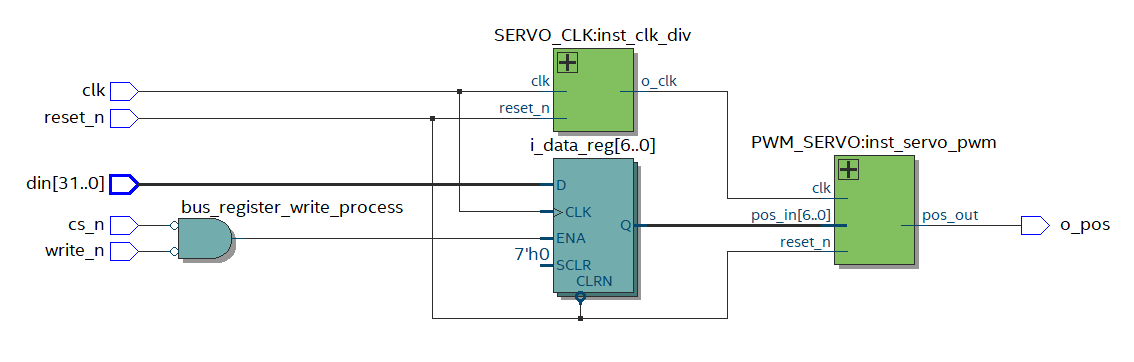
|  |  |  |  |
| --- | --- | --- | --- |
| **Namn** | **Beskrivning** | **Riktning** | **Typ** |
| clk | Systemklocka 50Mhz | IN | STD\_LOGIC |
| reset\_n | Reset signal | IN | STD\_LOGIC |
| wdt\_en\_n | Watchdog Enable | IN | STD\_LOGIC |
| restart\_n | Återställer räknare | IN | STD\_LOGIC |
|  |  |  |  |
| reset\_out\_n | Återställer systemet | UT | STD\_LOGIC |
| led | Led ut | UT | STD\_LOGIC\_VECTOR |

## Servo HW IP

Servo IP tolkar in värden till en PWM signal vilket sänds till en servomotor via en av I/O pinnarna på utvecklingskortet.

### IP Komponent

IP komponenten består av en servo klocka på som genererar en klocka på 64kHz och en PWM som beroende av data aktiverar en puls hög i 0,5-2,5ms var 20ms. PWM komponenten får värden direkt från ett dataregister.



Figur 12. SERVO IP RTL

Tabell 11. Signalbeskrivning SERVO IP

|  |  |  |  |
| --- | --- | --- | --- |
| **Namn** | **Beskrivning** | **Riktning** | **Typ** |
| clk | Systemklocka 50Mhz | IN | STD\_LOGIC |
| reset\_n | Reset signal | IN | STD\_LOGIC |
| din | Avalon Bus Data | IN | STD\_LOGIC\_VECTOR(31 downto 0) |
| cs\_n | Chip select | IN | STD\_LOGIC |
| write\_n | Write enable | IN | STD\_LOGIC |
|  |  |  |  |
| o\_pos | Servomotor position | UT | STD\_LOGIC |

# Konstruktionsbeskrivning SW

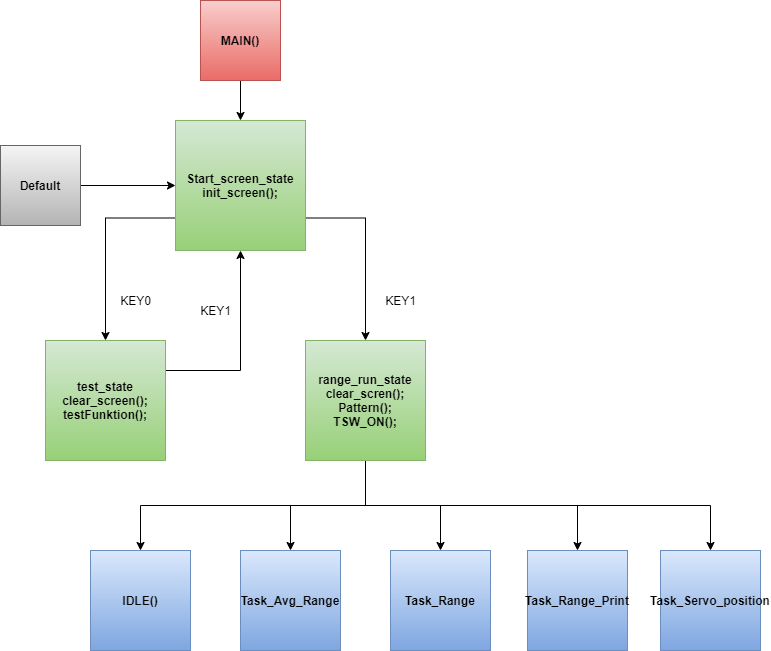
Mjukvaran är ett enkelt program som mäter avstånd. I kombination av avståndsdata och servomotorposition visualiseras avståndet av olika riktningar på en monitor. Main programmet består av en tillståndsmaskin med tre tillstånd. Start\_screen\_state, test\_state och range\_run\_state.

**Start\_screen\_state**: Vid uppstart av mjukvara visas en startsida med konstruktörens namn och information.  
För att välja nästa tillstånd används tryckknappar. Key0 för test\_state och Key1 för range\_run\_state.

**Test\_state**: Med switchar testas servomotorns position åt olika vinglar. Switch 0 = 0°, switch 1 = 90° och switch 2 = 180°. För att gå tillbaka till startskärmen måste key1 tryckas ned.

**Range\_run\_state**: Avståndsmätning startar och ett mönster målas ut för för varje vinkel avståndsmätningen sker. En gul cirkel målas ut på monitorn i förhållande till vinkel och avstånd för att visualisera avståndet till ett objekt.

Nedan visas tillståndsdiagrammet.



Figur 13. Tillståndsdiagram

## Tasks

### Task\_avg\_Range

Räknar genomsnittlig och visar värdet längst ned på skärmen var tionde sekund.

### Task\_Range

Skriver avståndsvärdet längst ned på skärmen varannan sekund. Task\_range återställer även watchdog räknaren som finns i hårdvaran.

### Task\_Range\_Print

I förhållande till servo position och avståndsvärde målas en gul cirkel på monitorn.

### Task\_Servo\_Position

Ger servomotorn ny position. Motorn position ökar 0-180° och sedan minskar 180-0°.

## Drivrutiner

### Init\_screen

Genererar välkomstskärm med information

Funktionsanrop: init\_screen();

Pseudokod:

* Rensa skärmen, clear\_screen(färg);
* Information på skärmen, tty\_print(position,färg)

### Pattern

Genererar ett mönster med gröna linjer på skärmen

Funktionsanrop: pattern();

Pseudokod:

* Draw\_hline(position,färg)
* Draw\_vline(position,färg)
* Draw\_agnled\_line(position,färg)

### TestFunction

Genererar information på skärmen. Med switchar 0,1 och 2 testas genereras positioner till servomotorn.

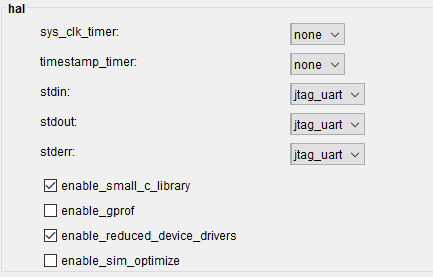
Funktionsanrop: testFunction();

Pseudokod:

* Int key,sw;
* Tty\_print(position,färg);
* If sw0= position1, if sw1=position, if sw2=position
* If key1 push and release, break;

## Kodstorlek

Figuren nedan visar de instälnigar i BSP som användes vila är *small C library*  och *reduced device drivers*.



Figur 14. Inställnigar i bsp

Mjukvarans kodstorlek är följande:

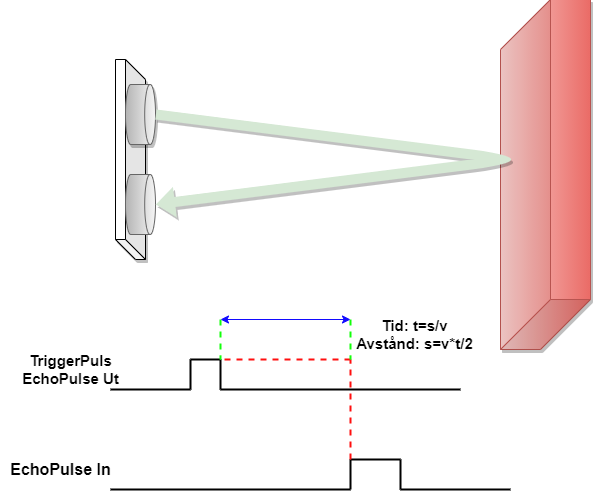
Tabell 12. Kodstorlek med BSP inställningar

|  |  |  |
| --- | --- | --- |
| **Minnestyp** | **Storlek Hex** | **Storlek Dec** |
| .entry &  .exceptions | 29C | 688 |
| .text | 4F18 | 20248 |
| .rodata | 59C | 1436 |
| .rwdata | 2FC | 764 |
| .bss | 1538 | 5432 |
| Totalt: | 6F98 | 28568 |

# Utrustning

## HC-SR05 Ultrasonic Range Sensor

Sensorn HC-SR05 används som transmitter och mottagare för ultraljud. En trigger signal ska enligt sensorns datablad vara aktiv i minst 10μs för att ljudvågor med en frekvens på 40kHz ska sändas. Trigger signalen aktiverar samtidigt en räknare som räknar tills att ljudvågorna reflekteras tillbaka vilket gör att sensorns mottagare genererar en “eko”puls, visas i figur 15.



Figur 15. Ultraljudssensor

Figur 16 visar sensorn och dess specifikation nedanför.

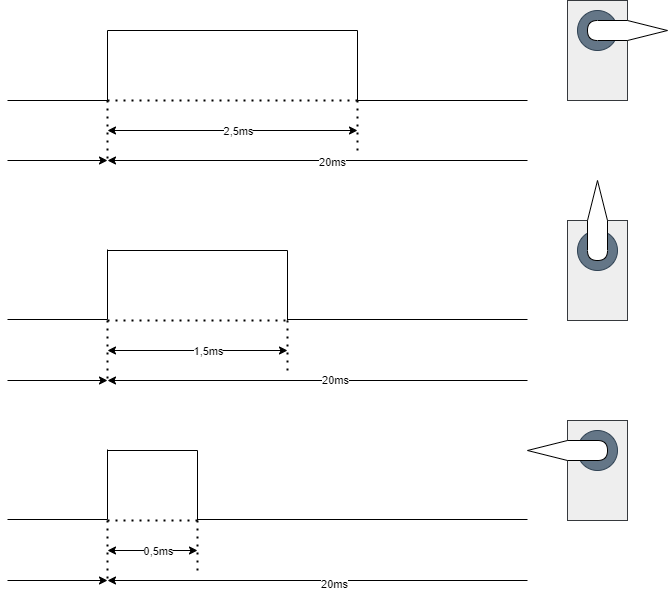


Figur 16. Sesnor för avståndsmätare[1]

voltage.........................................................................................................4.5 to 5.5 VDC sound frequency ...................................................................................................... 40 KHz measurement resolution ............................................................................................0.3 cm measurement angle ...................................................................................................... 15° supply current.................................................................................................. 10 to 40 mA trigger pin format ..............................................................................................10 µS pulse connector .......................................................................................................... 5-pin male detection distance............................................................................................. 2 to 450 cm dimensions ............................................................................................... 45 x 20 x 13 mm

## SG90 Micro Servo

I projektet används SG90 Servo för att göra avståndsmätning åt flera riktningar genom att montera ultraljudssensorn på motorn. För att få utslag åt olika riktningar ska motorn enligt databladet få en puls hög under en specifik period. Figuren nedan beskriver acceptansen av olika faser.



Figur 17. Acceptans Aktiv puls 0-180°

Figuren nedan visar servomotorn och dess specifikation nedanför.



Figur 18. SG90 Micro Servo

Rotationsvinkel: 180°

Vridmoment: 1,8 kg/cm (11 Ncm) (at 4,8 V).

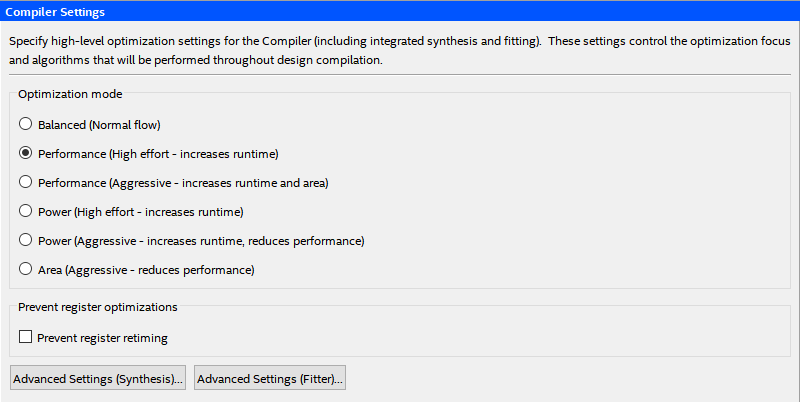
Vridmoment: 2,4 kg/cm (15 Ncm) (at 6 V).

Hastighet: 0,12 sec/60° (at 4,8 V).

Hastighet: 0,10 sec/60° (at 6 V).

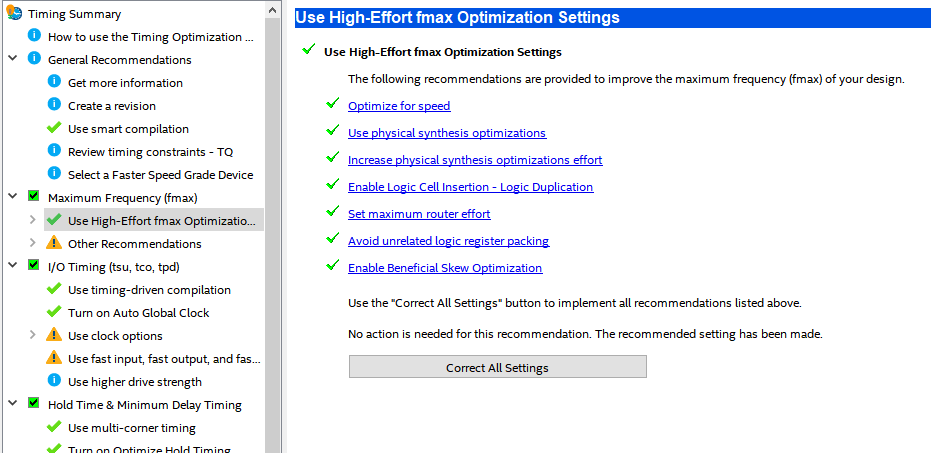
# Optimering, Settings & Assignments

I settings har kompileringsinställningarna ändrats till prestanda vilket gör att verktyget implementerar algoritmer i systemet grundas efter prestanda. Nedan visas Compiler Settings fönstret med ändrade optimerings läge från balanserad till prestanda.

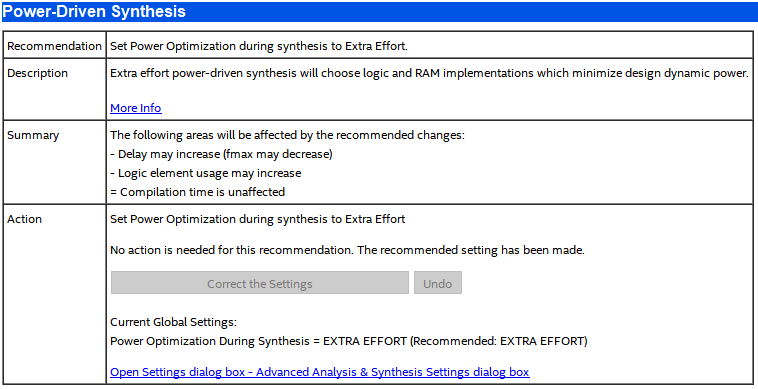


Figur 19. Compiler Settings

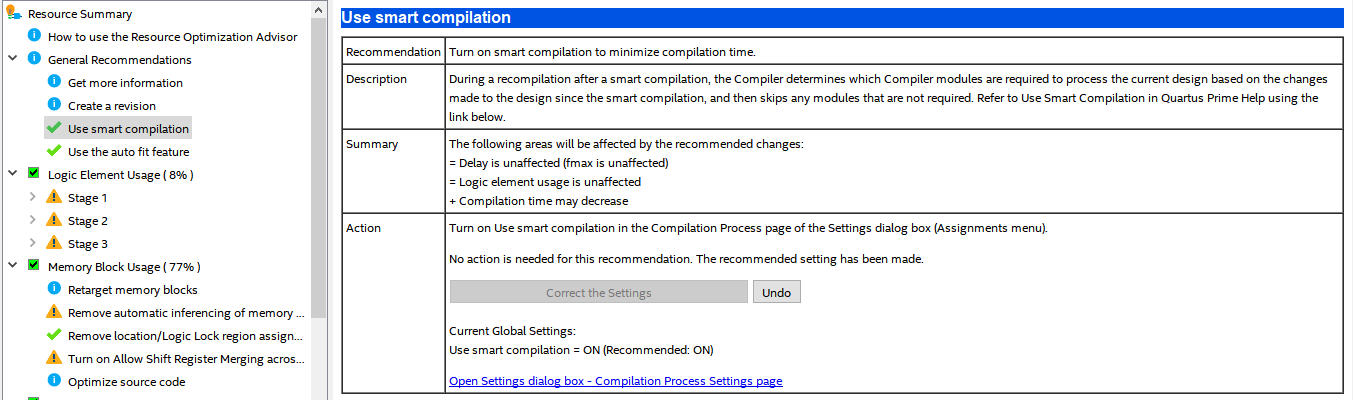
## Advisor

Konstruktioner använder Timing, Power och Resource advisors. Advisors med inställningar visas i figur 20-22.  


Figur 20. Timing Advisor



Figur 21. Power Advisor



Figur 22. Resource Advisor

## IO-pinnar

Konstruktionen använder:

* Total logic elements 3,698 / 49,760 ( 7 % )
* Total registers 2085
* Total pins 61 / 360 ( 17 % )

I/O-PINS:

Echo,Input,PIN\_V10,3,B3\_N0,PIN\_V10,2.5 V,,,,,

HEX0[6],Output,PIN\_C17,7,B7\_N0,PIN\_C17,2.5 V,,,,,

HEX0[5],Output,PIN\_D17,7,B7\_N0,PIN\_D17,2.5 V,,,,,

HEX0[4],Output,PIN\_E16,7,B7\_N0,PIN\_E16,2.5 V,,,,,

HEX0[3],Output,PIN\_C16,7,B7\_N0,PIN\_C16,2.5 V,,,,,

HEX0[2],Output,PIN\_C15,7,B7\_N0,PIN\_C15,2.5 V,,,,,

HEX0[1],Output,PIN\_E15,7,B7\_N0,PIN\_E15,2.5 V,,,,,

HEX0[0],Output,PIN\_C14,7,B7\_N0,PIN\_C14,2.5 V,,,,,

HEX1[6],Output,PIN\_B17,7,B7\_N0,PIN\_B17,2.5 V,,,,,

HEX1[5],Output,PIN\_A18,7,B7\_N0,PIN\_A18,2.5 V,,,,,

HEX1[4],Output,PIN\_A17,7,B7\_N0,PIN\_A17,2.5 V,,,,,

HEX1[3],Output,PIN\_B16,7,B7\_N0,PIN\_B16,2.5 V,,,,,

HEX1[2],Output,PIN\_E18,6,B6\_N0,PIN\_E18,2.5 V,,,,,

HEX1[1],Output,PIN\_D18,6,B6\_N0,PIN\_D18,2.5 V,,,,,

HEX1[0],Output,PIN\_C18,7,B7\_N0,PIN\_C18,2.5 V,,,,,

HEX2[6],Output,PIN\_B22,6,B6\_N0,PIN\_B22,2.5 V,,,,,

HEX2[5],Output,PIN\_C22,6,B6\_N0,PIN\_C22,2.5 V,,,,,

HEX2[4],Output,PIN\_B21,6,B6\_N0,PIN\_B21,2.5 V,,,,,

HEX2[3],Output,PIN\_A21,6,B6\_N0,PIN\_A21,2.5 V,,,,,

HEX2[2],Output,PIN\_B19,7,B7\_N0,PIN\_B19,2.5 V,,,,,

HEX2[1],Output,PIN\_A20,7,B7\_N0,PIN\_A20,2.5 V,,,,,

HEX2[0],Output,PIN\_B20,6,B6\_N0,PIN\_B20,2.5 V,,,,,

KEY[1],Input,PIN\_A7,7,B7\_N0,PIN\_A7,2.5 V,,,,,

KEY[0],Input,PIN\_B8,7,B7\_N0,PIN\_B8,2.5 V,,,,,

LEDR[9],Output,PIN\_B11,7,B7\_N0,PIN\_B11,2.5 V,,,,,

LEDR[8],Output,PIN\_A11,7,B7\_N0,PIN\_A11,2.5 V,,,,,

LEDR[7],Output,PIN\_D14,7,B7\_N0,PIN\_D14,2.5 V,,,,,

LEDR[6],Output,PIN\_E14,7,B7\_N0,PIN\_E14,2.5 V,,,,,

LEDR[5],Output,PIN\_C13,7,B7\_N0,PIN\_C13,2.5 V,,,,,

LEDR[4],Output,PIN\_D13,7,B7\_N0,PIN\_D13,2.5 V,,,,,

LEDR[3],Output,PIN\_B10,7,B7\_N0,PIN\_B10,2.5 V,,,,,

LEDR[2],Output,PIN\_A10,7,B7\_N0,PIN\_A10,2.5 V,,,,,

LEDR[1],Output,PIN\_A9,7,B7\_N0,PIN\_A9,2.5 V,,,,,

LEDR[0],Output,PIN\_A8,7,B7\_N0,PIN\_A8,2.5 V,,,,,

MAX10\_CLK1\_50,Input,PIN\_P11,3,B3\_N0,PIN\_P11,2.5 V,,,,,

servo0,Output,PIN\_W5,3,B3\_N0,PIN\_W5,2.5 V,,,,,

SW[9],Input,PIN\_F15,7,B7\_N0,PIN\_F15,2.5 V,,,,,

SW[8],Input,PIN\_B14,7,B7\_N0,PIN\_B14,2.5 V,,,,,

SW[7],Input,PIN\_A14,7,B7\_N0,PIN\_A14,2.5 V,,,,,

SW[6],Input,PIN\_A13,7,B7\_N0,PIN\_A13,2.5 V,,,,,

SW[5],Input,PIN\_B12,7,B7\_N0,PIN\_B12,2.5 V,,,,,

SW[4],Input,PIN\_A12,7,B7\_N0,PIN\_A12,2.5 V,,,,,

SW[3],Input,PIN\_C12,7,B7\_N0,PIN\_C12,2.5 V,,,,,

SW[2],Input,PIN\_D12,7,B7\_N0,PIN\_D12,2.5 V,,,,,

SW[1],Input,PIN\_C11,7,B7\_N0,PIN\_C11,2.5 V,,,,,

SW[0],Input,PIN\_C10,7,B7\_N0,PIN\_C10,2.5 V,,,,,

Trig,Output,PIN\_W10,3,B3\_N0,PIN\_W10,2.5 V,,,,,

VGA\_B[3],Output,PIN\_N2,2,B2\_N0,PIN\_N2,2.5 V,,,,,

VGA\_B[2],Output,PIN\_P4,2,B2\_N0,PIN\_P4,2.5 V,,,,,

VGA\_B[1],Output,PIN\_T1,2,B2\_N0,PIN\_T1,2.5 V,,,,,

VGA\_B[0],Output,PIN\_P1,2,B2\_N0,PIN\_P1,2.5 V,,,,,

VGA\_G[3],Output,PIN\_R1,2,B2\_N0,PIN\_R1,2.5 V,,,,,

VGA\_G[2],Output,PIN\_R2,2,B2\_N0,PIN\_R2,2.5 V,,,,,

VGA\_G[1],Output,PIN\_T2,2,B2\_N0,PIN\_T2,2.5 V,,,,,

VGA\_G[0],Output,PIN\_W1,2,B2\_N0,PIN\_W1,2.5 V,,,,,

VGA\_HS,Output,PIN\_N3,2,B2\_N0,PIN\_N3,2.5 V,,,,,

VGA\_R[3],Output,PIN\_Y1,3,B3\_N0,PIN\_Y1,2.5 V,,,,,

VGA\_R[2],Output,PIN\_Y2,3,B3\_N0,PIN\_Y2,2.5 V,,,,,

VGA\_R[1],Output,PIN\_V1,2,B2\_N0,PIN\_V1,2.5 V,,,,,

VGA\_R[0],Output,PIN\_AA1,3,B3\_N0,PIN\_AA1,2.5 V,,,,,

VGA\_VS,Output,PIN\_N1,2,B2\_N0,PIN\_N1,2.5 V,,,,,

# Verifiering/Validering

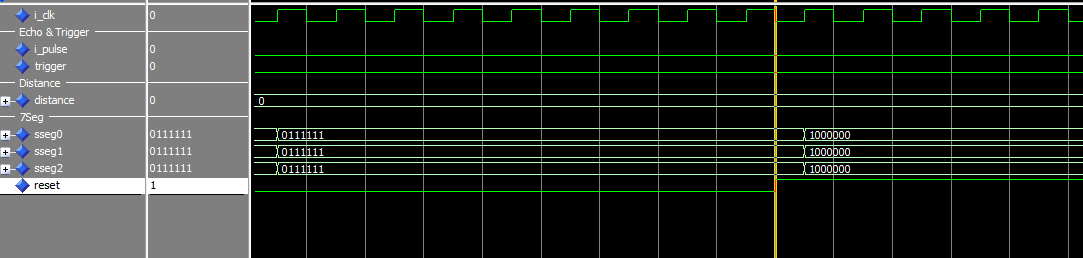
## Verifiering Ultraljudssensor

Tabell 13. Testfall Verifiering Ultraljudssensor

|  |  |  |  |
| --- | --- | --- | --- |
| **Testfall** | **Beskrivning** | **Acceptans** | **Godkänt (J/N)** |
| **0** | Reset Aktiv Låg | 7-segdisplay visar streck Distans = 0 | Ja |
| **1** | Reset Aktiv Hög | 7-segdisplay visar avståndsvärde | Ja |
| **2** | Virtuell Echo Puls med ökad fördröjning för varje puls | Distansen ökar 7-segdisplay visar avståndsvärde | Ja |

### Testfall 0-1

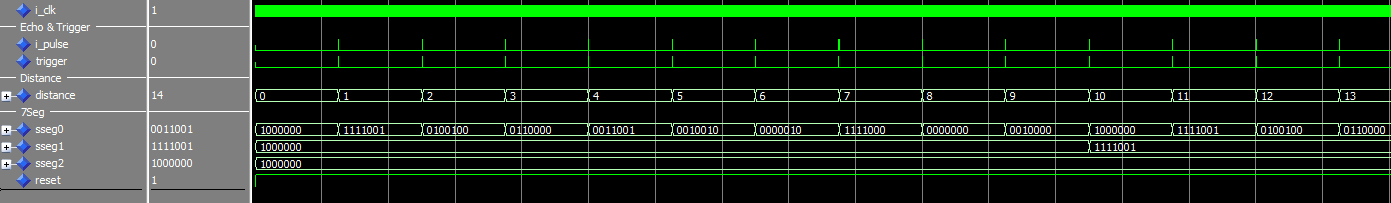
Nedan visas ModelSim där 7-segmentdysplayer går från att visa streck till att visa avståndsvärde.



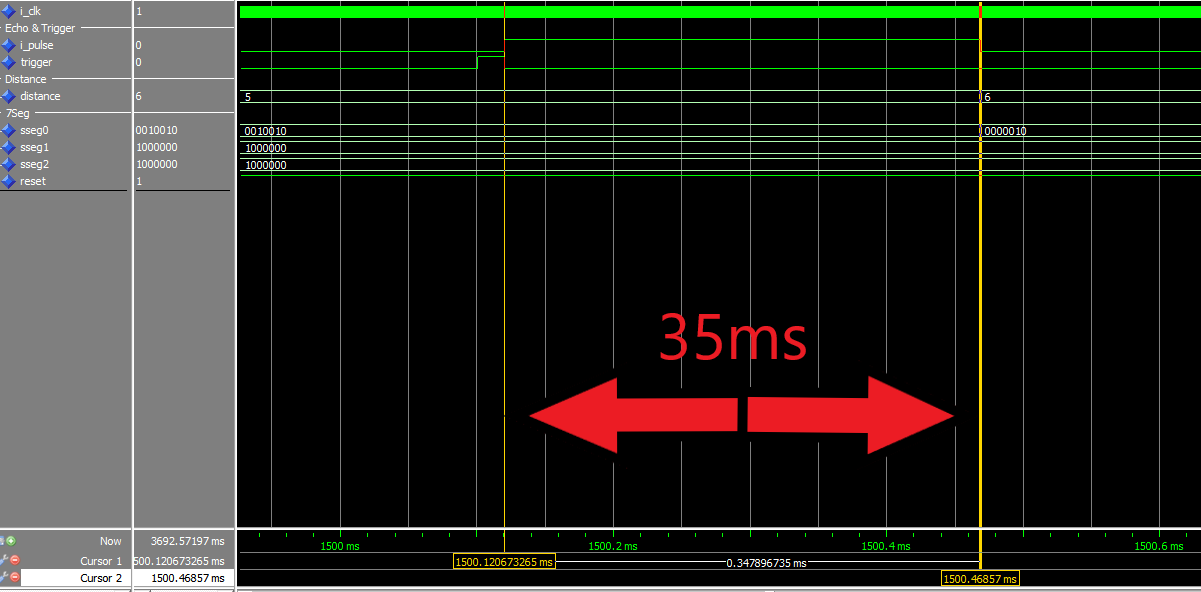
Figur 23. ModelSim Sonar Testfall 0-1

### Testfall 2

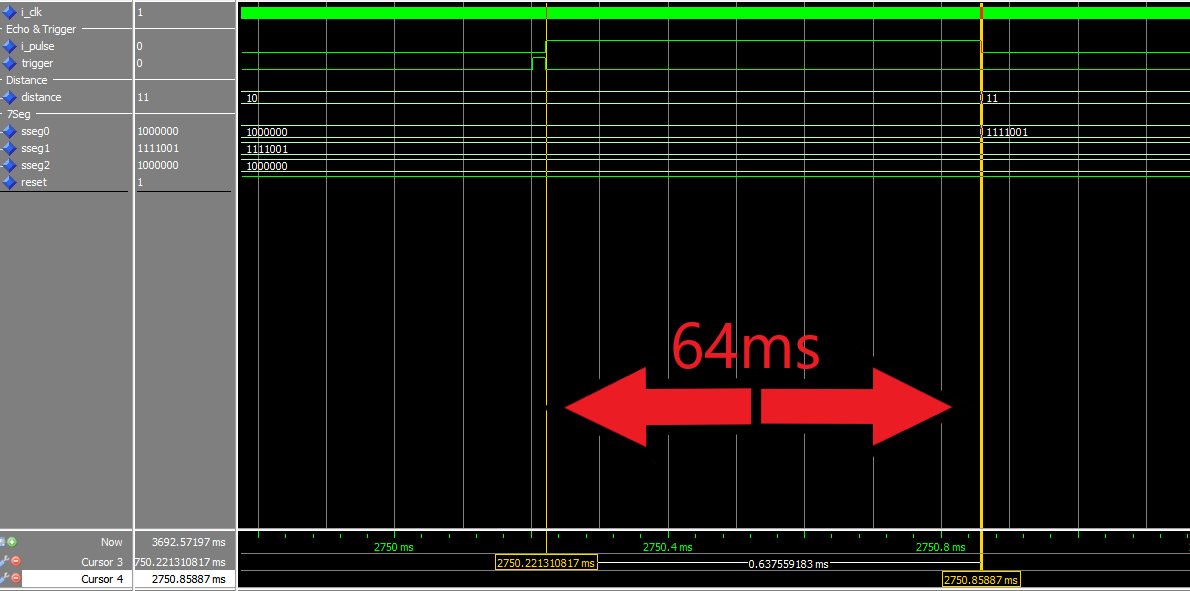
Figuren nedan visar ökad distans. Figur 24 och 25 visar olika värden vid olika fördröjningar.



Figur 24. ModelSim Sonar Testfall2



Figur 25. ModelSim Sonar Testfall 2, 35ms fördröjning



Figur 26. ModelSim Sonar Testfall 2, 64ms fördröjning

## Verifiering Servomotor

Tabell 14. Testfall SERVO\_HW\_IP

|  |  |  |  |
| --- | --- | --- | --- |
| **Testfall** | **Beskrivning** | **Acceptans** | **Godkänt (J/N)** |
| **0** | Reset=0 | Ingen position ut | Ja |
| **1** | Chipselect=0 Write enable=0 Reset=1 Data in=16 | Puls ut =0,5ms (Servo vinkel 0°) | Ja |
| **2** | Chipselect=0 Write enable=0 Reset=1 Data in=50 | Puls ut=1,5ms (Servo vinkel 90°) | Ja |
| **3** | Chipselect=0 Write enable=0 Reset=1 Data in=80 | Puls ut=2,5ms (Servo vinkel 180°) | Ja |

### Testfall 0



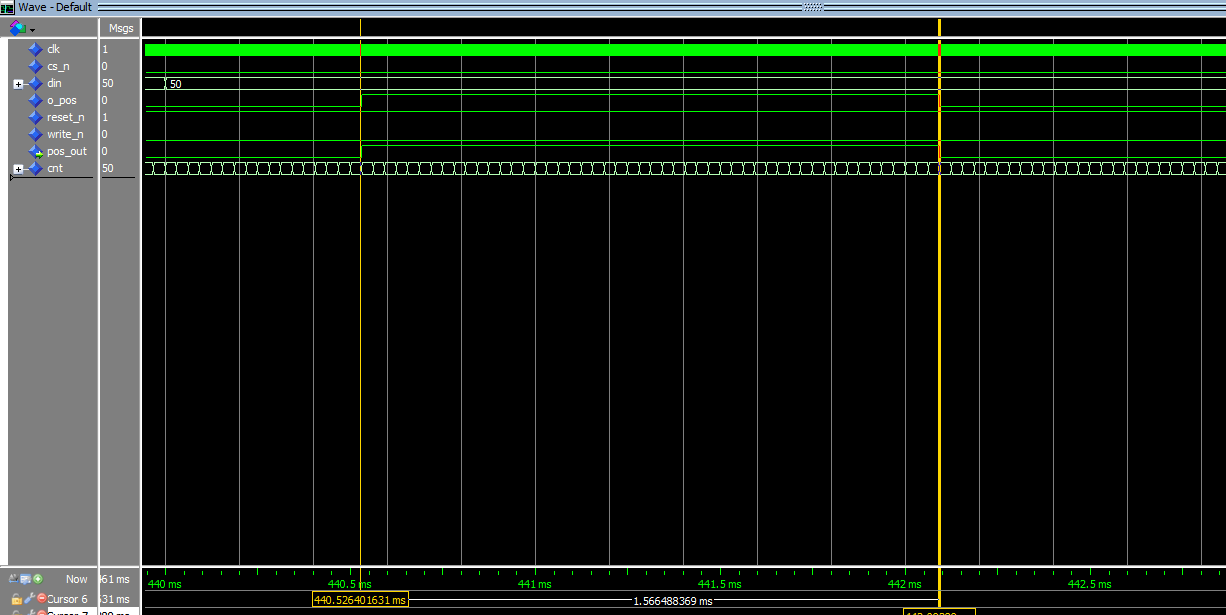
Figur 27. Testfall 0

### Testfall 1



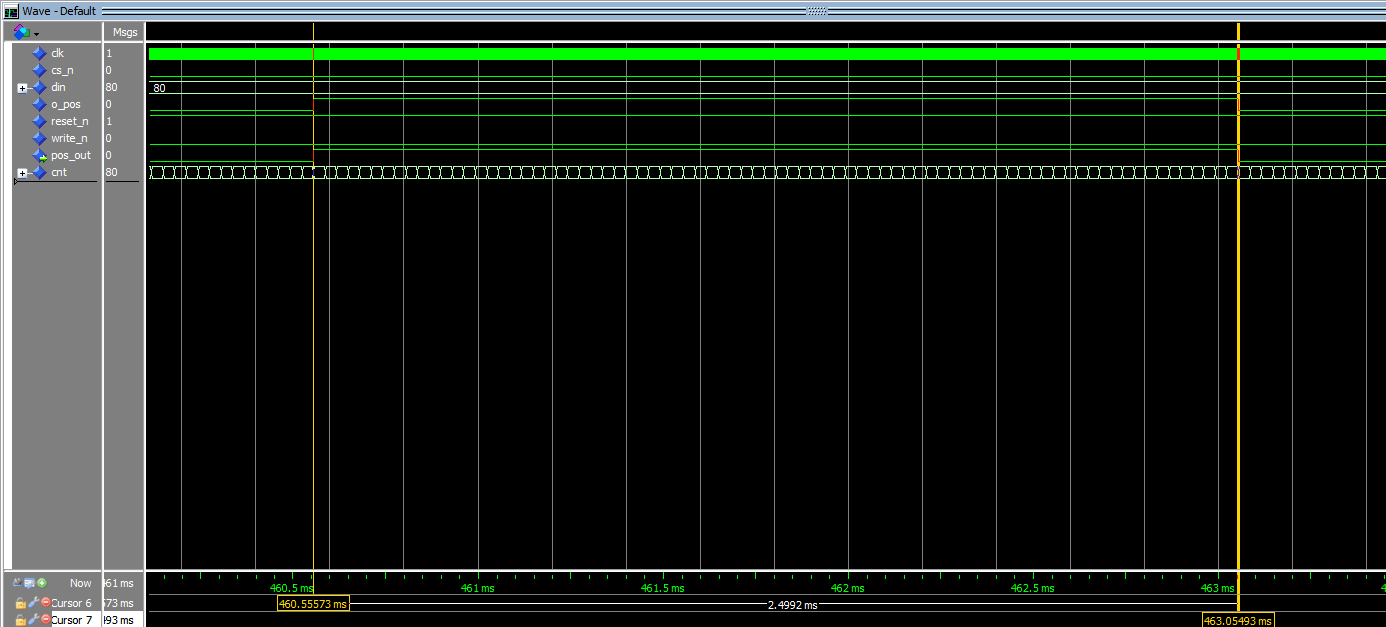
Figur 28. Testfall 1

### Testfall 2



Figur 29. Testfall 2

### Testfall 3



Figur 30. Testfall 3

## Resultat från validering

Tabell 15 visar resultaten efter valideringen med en videolänk nedanför som beskriver och validerar systemet.

Tabell 15. Valideringsresultat

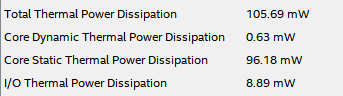
|  |  |  |  |
| --- | --- | --- | --- |
| **Testfall** | **Beskrivning** | **Acceptans** | **Godkänt (J/N)** |
| **0** | Start Skärmen | Information visas på VGA-särmen: Konstruktör, företag, Datum | Ja |
| **1** | Start knapp Tryck KEY1 | Informationen försvinner Mönster målas på skärmen | Ja |
| **2.1** | Motorposition 0 grader | Markera riktning blå cirkel Gul cirkel reagerar på visualiserar avstånd. | Ja |
| **2.2** | Motorposition 25 grader | Markera riktning blå cirkel Gul cirkel reagerar på visualiserar avstånd. | Ja |
| **2.3** | Motorposition 50 grader | Markera riktning blå cirkel Gul cirkel reagerar på visualiserar avstånd. | Ja |
| **2.4** | Motorposition 75 grader | Markera riktning blå cirkel Gul cirkel reagerar på visualiserar avstånd. | Ja |
| **2.5** | Motorposition 90 grader | Markera riktning blå cirkel Gul cirkel reagerar på visualiserar avstånd. | Ja |
| **2.6** | Motorposition 115 grader | Markera riktning blå cirkel Gul cirkel reagerar på visualiserar avstånd. | Ja |
| **2.7** | Motorposition 140 grader | Markera riktning blå cirkel Gul cirkel reagerar på visualiserar avstånd. | Ja |
| **2.8** | Motorposition 165 grader | Markera riktning blå cirkel Gul cirkel reagerar på visualiserar avstånd. | Ja |
| **2.9** | Motorposition 180 grader | Markera riktning blå cirkel Gul cirkel reagerar på visualiserar avstånd. | Ja |

Videolänk på youtube: <https://www.youtube.com/watch?v=zPpom8koREg>

# Analys

## Effektförbrukning

Enligt rapporten av Quartus Power Analyzer uppskattas designen ha en effektförbrukning på 105,69mW.Figur 27 visar en rapport från Power Analyzer.

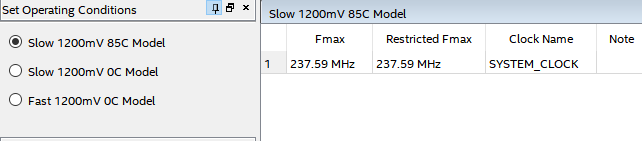


Figur 31. Uppskattad effektförbrukning

## Tidsanalys

Tidsanalysen är endast utförd på SERVO\_HW\_IP komponenten vilket inte visar några avvikelser.

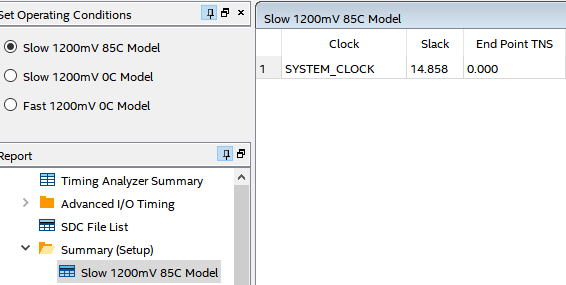
Det valda driftstillståndet är Slow 1200mV 85C Model och visar att komponenten har en Fmax på 237,59MHz vilket visas i figur 28.



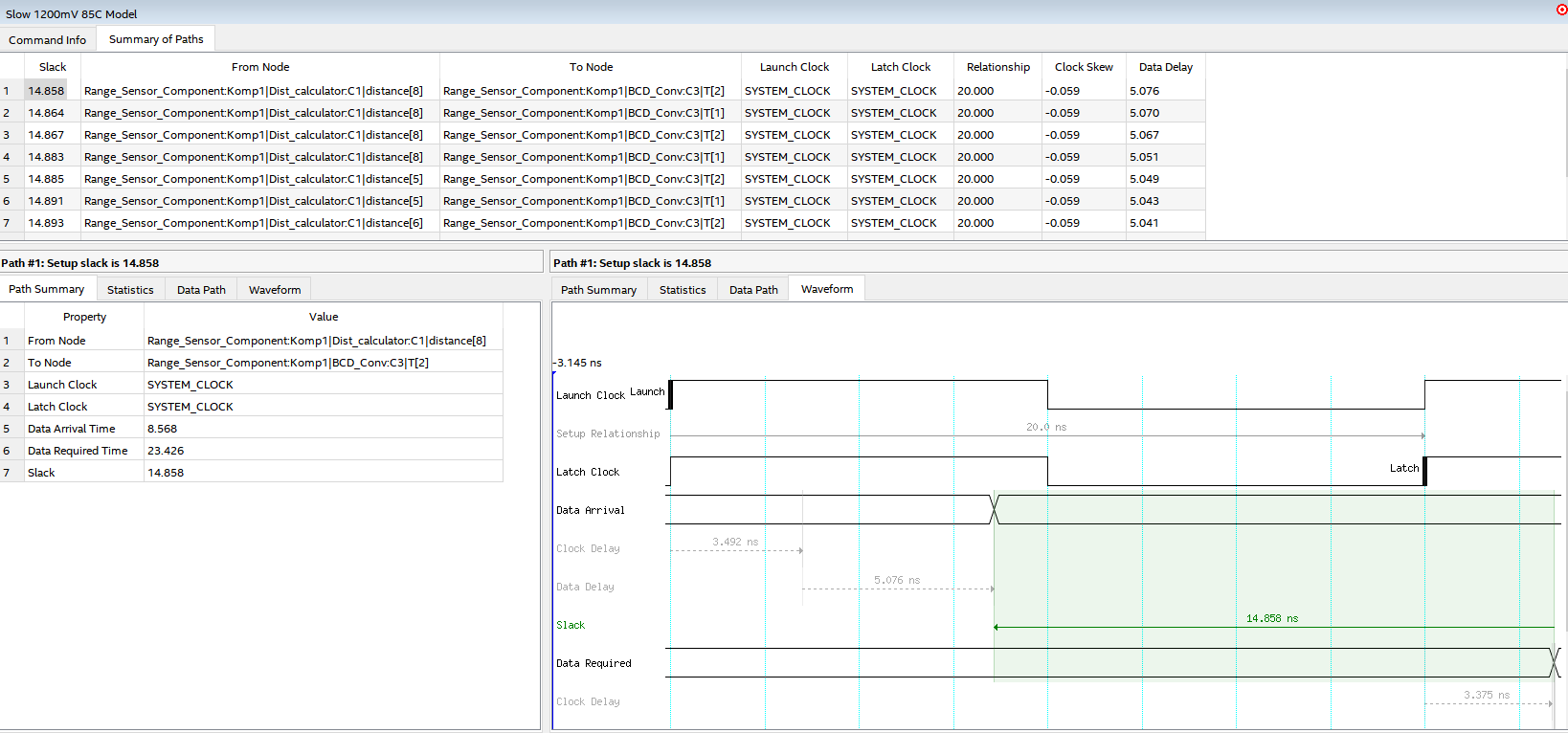
Figur 32. Srvo\_IP FMAX

### Kortast Slack

Nätet med kortast skack hamnar på 14,858ns med data arrival time på 8,568ns vilket visas i figur 29 och 30.



Figur 33. Slow 120mV 85C Slack



Figur 34. Slow 120mV 85C Waveform

# Slutsats

“Bra att jobba med olika steg, eftersom det kan lätt bli komplicerat och många man månader ” vilket verkligen behövdes för att fullfölja tidplanen. Till en början flöt arbetet på bra fram tills att jag insåg att det klassiska radar mönstret inte skulle fungera. För att spara på tid försökte jag att hitta genvägar för att lösa problemet vilket var ett misstag. Det genererade en kedjereaktion av problem som dök upp. Efter mycket strul var lösningen att gå tillbaka ett par steg och jobba stegvis. I slutskede efter en del optimeringar dök det upp lite problem med watchdog räknaren. Ibland hänger sig en task lite för länge och tasken som återställer watchdogen varannan sekund hinner inte alltid återställa räknaren om det blir förseningar.

# REFERENSER

1. DE10-Lite User Manual  
   21-06-10  
   <https://www.terasic.com.tw/cgi-bin/page/archive_download.pl?Language=English&No=1021&FID=a13a2782811152b477e60203d34b1baa>
2. PWM Controller  
   21-06-13  
   <https://vhdlwhiz.com/pwm-controller/>
3. 21-06-17  
   <https://www.fpga4student.com/2017/06/pwm-generator-in-vhdl.html>

21-06-17  
<https://howtomechatronics.com/projects/arduino-radar-project/>