|  |
| --- |
| TEIS AB |
| TEIS ECS - Embedded Computer System - |
|  |
|  |
| **Teis Teisson** |
| **2016-09-19** |

|  |
| --- |
| Sammanfattning: TEIS datorsystem är en enkel men komplett dator som exekveras på ett FPGA-kort. Datorn består av en CPU, adressbuss-decoder, ROM, samt ett ingångsfilter för att välja manuell eller automatisk klockning av processorn. Datorn presenterar även register, bussar och chip select signaler på kortets sjusegmentdisplayer och lysdioder. |

INNEHÅLLSFÖRTECKNING

[1 INLEDNING 4](#_Toc462138010)

[2 KRAVSPECIFIKATION 4](#_Toc462138011)

[3 ÖVERSIKT AV SYSTEMARKITEKTUR OCH KOMPONENTHIERARKI 5](#_Toc462138012)

[3.1 Symbol 5](#_Toc462138013)

[3.2 In/utgångar 5](#_Toc462138014)

[3.3 TEIS Embedded Datorsystem arkitektur (ECS) 6](#_Toc462138015)

[3.4 General package and library 7](#_Toc462138016)

[4 INGÅENDE KOMPONENTER 7](#_Toc462138017)

[4.1 CPU – komponent 7](#_Toc462138018)

[4.1.1 Funktion, arkitektur och tillståndsmaskin 8](#_Toc462138019)

[4.1.2 In/utgångar 8](#_Toc462138020)

[4.1.3 Tillståndsmaskin 9](#_Toc462138021)

[4.1.4 Beskrivning av CPU:ns register, operationer, databuss, adressbuss och kontrollsignaler 10](#_Toc462138022)

[4.1.5 Beskrivning av CPU:ns arbetssätt 10](#_Toc462138023)

[4.2 ROM – komponent 11](#_Toc462138024)

[4.2.1 Funktion och arkitektur 11](#_Toc462138025)

[4.2.2 In/utgångar 12](#_Toc462138026)

[4.2.3 RTL-nivå 12](#_Toc462138027)

[4.2.4 VHDL-nivå 12](#_Toc462138028)

[4.3 LED – komponent 14](#_Toc462138029)

[4.3.1 Funktion och arkitektur 14](#_Toc462138030)

[4.3.2 In/utgångar 14](#_Toc462138031)

[4.3.3 RTL-nivå 14](#_Toc462138032)

[4.3.4 VHDL-kod 14](#_Toc462138033)

[4.4 Adressbussdekoder – komponent 14](#_Toc462138034)

[4.4.1 Funktion och arkitektur 14](#_Toc462138035)

[4.4.2 In/utgångar 14](#_Toc462138036)

[4.4.3 RTL-nivå 14](#_Toc462138037)

[4.4.4 VHDL-kod 14](#_Toc462138038)

[4.5 Ingångsfilter – komponent 14](#_Toc462138039)

[4.5.1 Funktion och arkitektur 14](#_Toc462138040)

[4.6 Status display – komponent 15](#_Toc462138041)

[4.6.1 Funktion och arkitektur 15](#_Toc462138042)

[4.6.2 In/utgångar 16](#_Toc462138043)

[4.7 Sju\_seg\_displayer – komponent 17](#_Toc462138044)

[4.7.1 Funktion arkitektur 17](#_Toc462138045)

[4.7.2 In/Utgångar 18](#_Toc462138046)

[4.7.3 RTL-nivå 18](#_Toc462138047)

[4.8 Sju\_seg\_displayer\_CPU\_STATE 19](#_Toc462138048)

[4.8.1 Funktion arkitektur 19](#_Toc462138049)

[4.8.2 In/Utgångar 19](#_Toc462138050)

[4.8.3 RTL-nivå 19](#_Toc462138051)

[4.8.4 VHDL-kod 19](#_Toc462138052)

[5 GRANSKNING OCH FÖRSLAG PÅ FÖRBÄTTRINGAR 20](#_Toc462138053)

[6 FOOT PRINT 20](#_Toc462138054)

[7 KOSTNAD FÖR PROJEKTET 20](#_Toc462138055)

# 1 INLEDNING

Denna rapport beskriver ett datorsystem skrivet i VHDL. Systemet har analyserats genom simulering och verifiering i ModelSim och därefter validerats på ett lämpligt FPGA-kort.

# 2 KRAVSPECIFIKATION

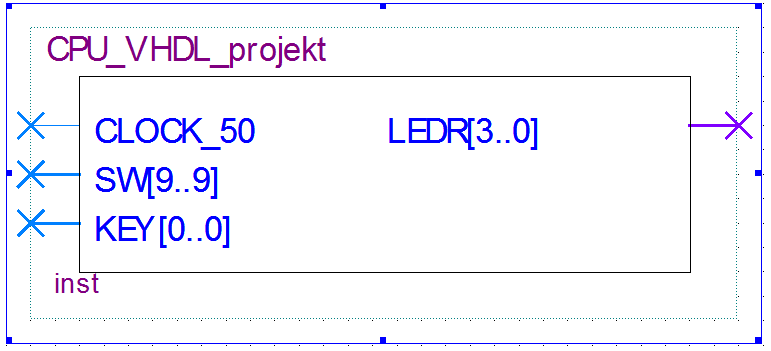
Tabell 1. Kravspecifikation från TEIS uppgift\_vhdl\_7

Klistra in specifikationen från uppgift 7

# 3 ÖVERSIKT AV SYSTEMARKITEKTUR OCH KOMPONENTHIERARKI

TEIS datorsystem består av en CPU, adressbuss-decoder, ROM, samt ett ingångsfilter för att välja manuell eller automatisk klockning av processorn. Resultat presenteras på LEDG 3..0. Datorn klockas normalt manuellt för operatör ska kunna se vad som händer vid varje enskild klockcykel. I Figur 1 visas toppnivån med in och utgångar. Figur 2 visar det kort som används. Figur 3 visar systemarkitekturen och figur 4 hierarkien.

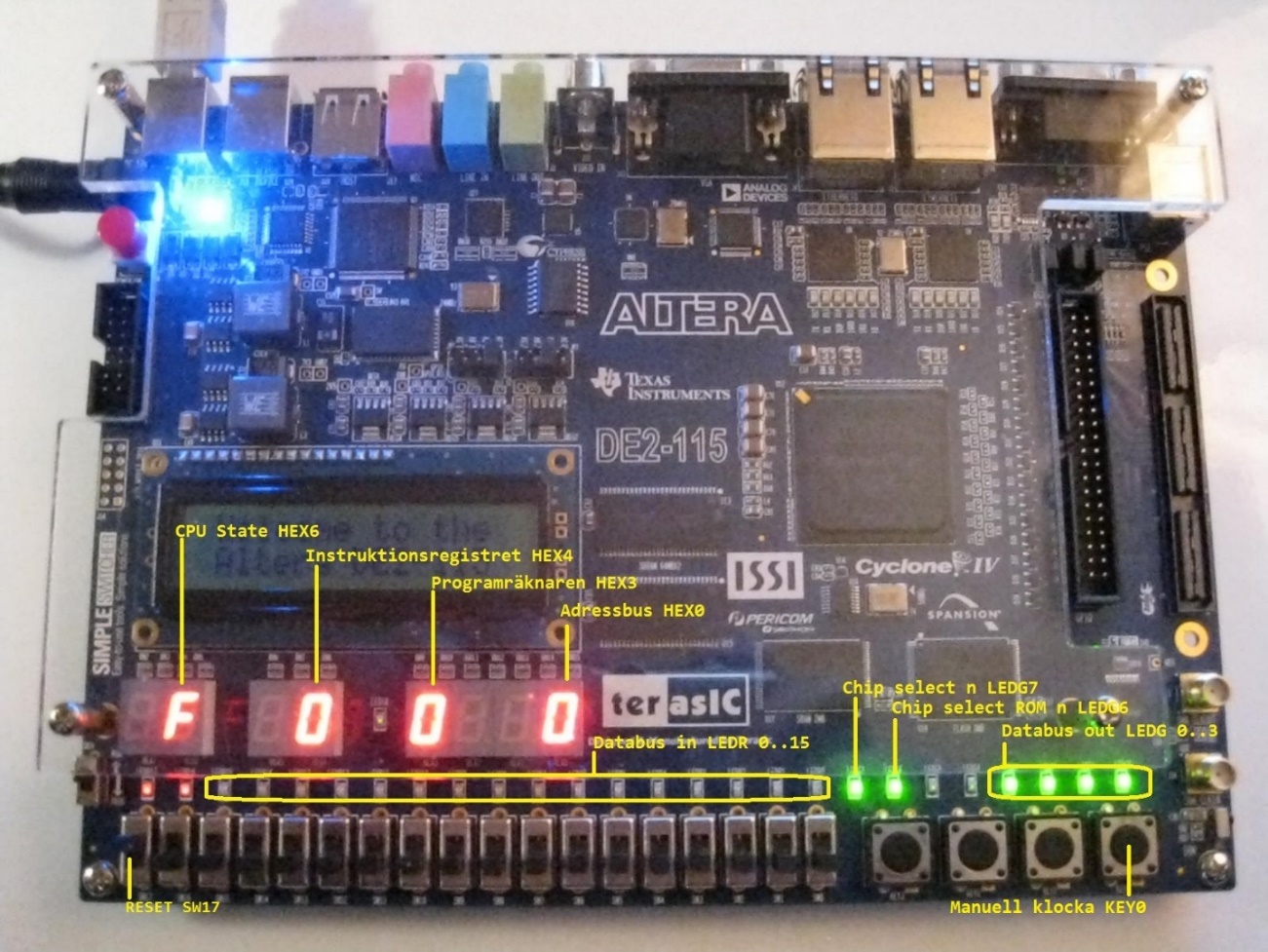
## 3.1 Symbol



Figur 1. Toppnivån för TEIS mikrodatorsystem

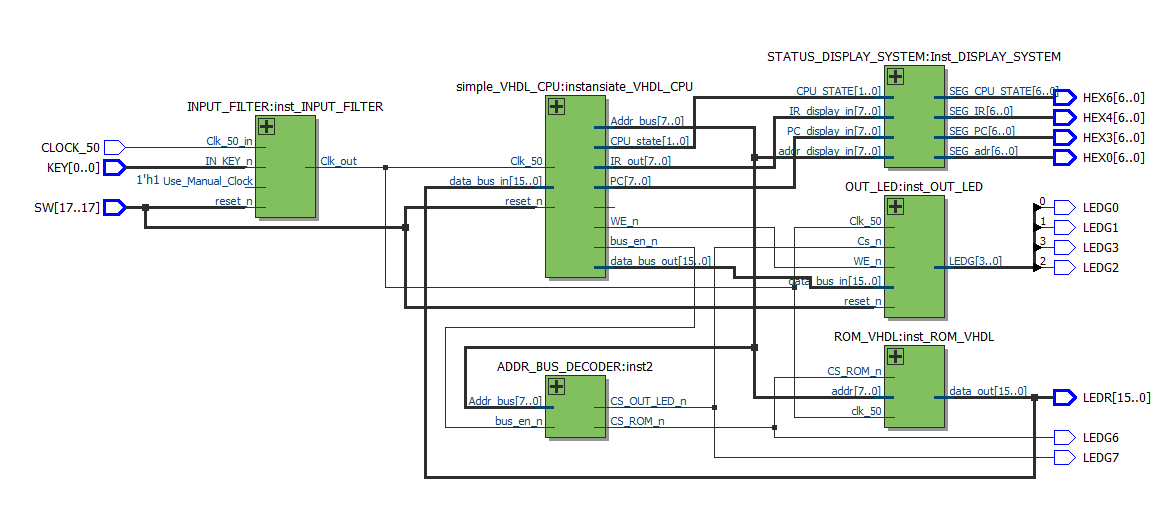
## 3.2 In/utgångar

Beskriv in och utsignaler

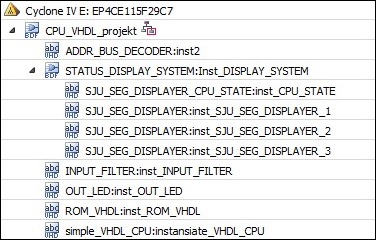


Figur 2. TEIS datorsystem på DE2-115 (byt figuren till det kortet som används i detta projekt)

## 3.3 TEIS Embedded Datorsystem arkitektur (ECS)



Figur 3. Systemarkitekturen för TEIS datorsystem



Figur 4. Komponenthierarki i TEIS datorsystem

Systemet använder ett ROM för att lagra programkoden i. Se minnesmappen i tabell 2. Resultat lagras i systemets register.

Tabell 2. Minnesmapp (komplettera med I/O)

|  |  |  |  |
| --- | --- | --- | --- |
| **Typ** | **Minnesområde** | **Storlek** | **Minnestyp** |
| Programkod | 0x0 – 0xF | 15 | ROM |

## 3.4 General package and library

Svara på frågorna:

1. Vilka ”library” och ”package” som används.

En kort presentation av varje ”package”

# 4 INGÅENDE KOMPONENTER

## 4.1 CPU – komponent

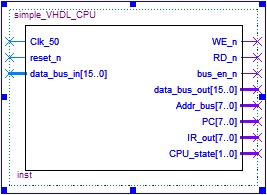
**Komponent namn:** simple\_VHDL\_CPU

**Instansnamn:** instansiate\_VHDL\_CPU

### 4.1.1 Funktion, arkitektur och tillståndsmaskin

CPU-komponentens in och utgångar visas i nästa figur. CPU:n styrs av reset eller klocksignal. CPU:n arbetar med en tillståndsmaskin som kan exekvera instruktionerna NOP, LOAD, STORE och JMP. Internt använder CPU:n programräknaren, instruktionsregistret och dataregistret. Programräknaren pekar ut vilken instruktion som skall hämtas från ROM.

Vid reset initieras programräknaren, bussar och register till 0 medan enable-signaler initieras till 1. Vid positiv klocksignal initieras enable-signalerna till 1 och går sedan in i tillståndsmaskinen. Se figur 5.



Figur 5. CPU symbol

### 4.1.2 In/utgångar

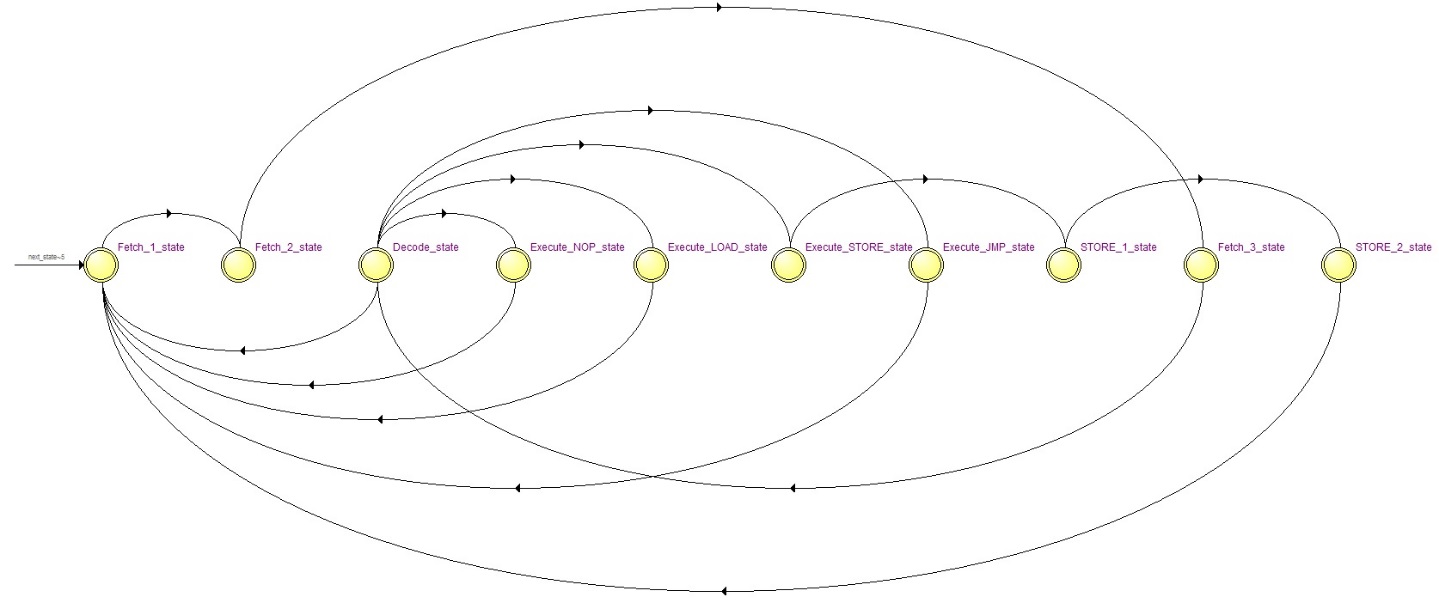
CPU:ns in och utsignaler visas i Tabell 3.

Tabell 3. CPU:ns in och utsignaler

|  |  |  |  |
| --- | --- | --- | --- |
| **Signal** | **Namn** | **Riktning** | **Typ** |
| Clocksignal 50 MHz | Clk\_50 | in | std\_logic |
| Reset | reset\_n | in | std\_logic |
| Write enable | WE\_n | ut | std\_logic |
| Read enable | RD\_n | ut | std\_logic |
| Bus enable | bus\_en\_n | ut | std\_logic |
| Databus ut | data\_bus\_out | ut | std\_logic\_vector(15 downto 0) |
| Databus in | data\_bus\_in | ut | std\_logic\_vector(15 downto 0) |
| Adressbuss | Addr\_bus | ut | std\_logic\_vector(7 downto 0) |
| Programräknare | PC | ut | std\_logic\_vector(7 downto 0) |
| Instruktionsregister | IR\_out | ut | std\_logic\_vector(7 downto 0) |
| Tillstånd | CPU\_state | ut | std\_logic\_vector(1 downto 0)) |

### 4.1.3 Tillståndsmaskin

Tillståndsmaskinen arbetar i fyra tillstånd, fetch, decode, execute och store. Fetch hämtar nästa instruktion. Decode avkodar vad som ska göras. Execute utför det som beslutats i decode-fasen. Store lagrar data till minnet. Figur 6 nedan visar tillståndsmaskinen i detalj.



Figur 6. Tillståndsmaskinen (rita om den så tillstånden kan urskiljas)

Tillstånds beskrivning:

* **Fetch**

CPU\_state får värdet ”00”. Data hämtas in till instruktionsregistret. Värdet på programräknaren läggs ut på adressbussen. rd och bus enable sätts till 0. I IR läggs instruktionen in från databussen.

* **Decode:**

Först ökas programräknaren med ett och CPU\_state får värdet ”01”. Berode på vilken instruktion som ligger i instruktionsregistret väljs vilken instruktion som skall exekveras i fasen Execute

* **Execute**

CPU\_state får värdet”10”. Om NOP exekveras görs ingenting. Om Load exekveras läggs IR in i dataregister. Om Store exekveras kopieras registret till databussen och adressbussen får värdet i IR. Om JMP exekveras får programräknaren värdet i IR

* **Store**

CPU\_state får värdet ”11”. Först sätts write enable och bus enable till 0 och sedan sätts de till 1.

### 4.1.4 Beskrivning av CPU:ns register, operationer, databuss, adressbuss och kontrollsignaler

CPU:n arbetar internt med tre olika register. Programräknare, instruktionsregister och dataregister. Se Tabell 4 för detaljerad beskrivning.

Tabell 4. CPUns interna register

|  |  |  |
| --- | --- | --- |
| **Register** | **Namn** | **Beskrivning** |
| Programräknare | PC\_reg | Innehåller vilken adress CPU ska läsa från ROM |
| Instruktionsregister | IR | Innehåller vilken instruktion CPU ska utföra |
| Dataregister | CPU\_REG\_0 | Internt register för att överföra data |

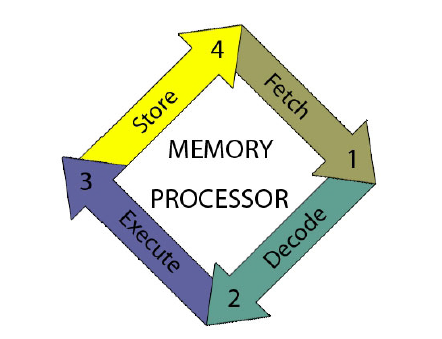
Tabell 5: Assembler instruktioner, OP-kod och kodning

En tabell med alla assemblerinstruktioner

### 4.1.5 Beskrivning av CPU:ns arbetssätt

Svara kortfattat på frågorna:

1. Vad är maskinkoden?
2. Vad är Assembler?
3. Hur är principen för sättet CPU:n jobbar på?



Figur 7: CPU:ns arbetsätt

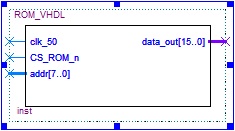
## 4.2 ROM – komponent

**Komponent namn:** ?

**Instansnamn:** ?

### 4.2.1 Funktion och arkitektur

ROM används för att lagra maskinkoden som CPU:n ska exekvera. Vid positiv klockflank läggs data ut på databussen på den adress ROM har som insignal. Chip select används inte internt i ROM. ROM-komponentens in och utgångar visas i Figur 8 och innehållet i ROM visas i Tabell 6. RTL-nivån på ROM visas i Figur 9.



Figur 8. ROM Symbol

Tabell 6. Innehåll i ROM

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Adress** | **Maskinkod [HEX]** | **Assemblerkod** | **Instruktion [HEX]** | **Data [HEX]** |
| 0 | 0000 | NOP | 0 | 000 |
| 1 | 100A | LOAD\_R0 #A | 1 | 00A |
| 2 | 2010 | STORE\_R0 #10 | 2 | 010 |
| 3 | 1001 | LOAD\_R0 #1 | 1 | 001 |
| 4 | 2010 | STORE\_R0 #10 | 2 | 010 |
| 5 | 3001 | JMP #1 | 3 | 001 |
| 6 | 0000 | NOP | 0 | 000 |
| 7 | 0000 | NOP | 0 | 000 |
| 8 | 0000 | NOP | 0 | 000 |
| 9 | 0000 | NOP | 0 | 000 |
| 10 | 0000 | NOP | 0 | 000 |
| 11 | 0000 | NOP | 0 | 000 |
| 12 | 0000 | NOP | 0 | 000 |
| 13 | 0000 | NOP | 0 | 000 |
| 14 | 0000 | NOP | 0 | 000 |
| 15 | 0000 | NOP | 0 | 000 |

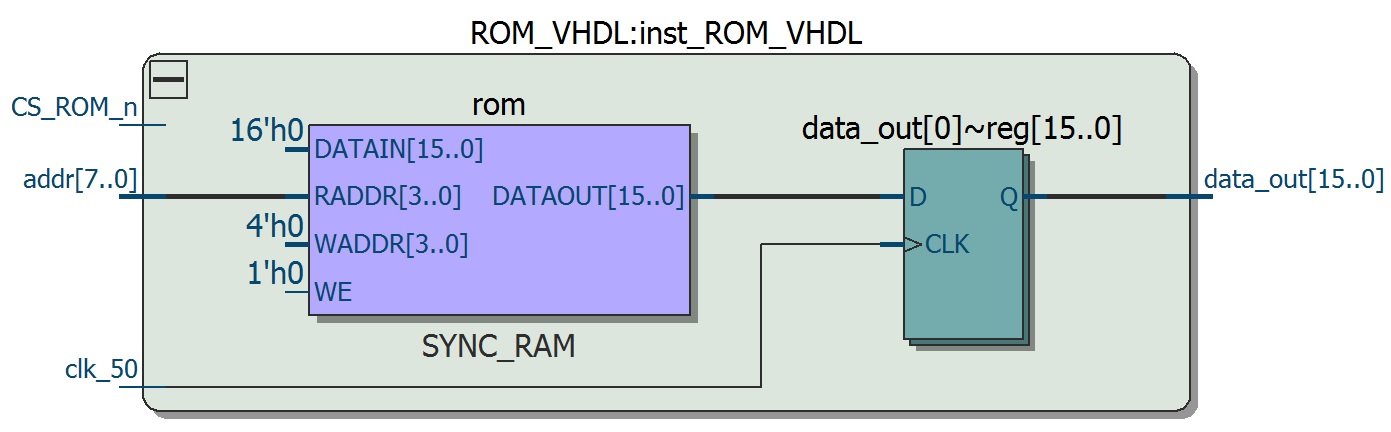
### 4.2.2 In/utgångar

In- och utsignalerna till ROM enligt Tabell 7.

Tabell 7. In- och utsignalerna till ROM

|  |  |  |  |
| --- | --- | --- | --- |
| **Signal** | **Namn** | **Riktning** | **Typ** |
| Klocksignal 50 MHz | clk\_50 | in | std\_logic |
| Chipselect | CS\_ROM\_n | in | std\_logic |
| Adressbuss | addr | in | std\_logic\_vector(7 downto 0); |
| Data ut | data\_out | ut | out std\_logic\_vector(15 downto 0) |

### 4.2.3 RTL-nivå



Figur 9. RTL-nivån på ROM

### 4.2.4 VHDL-nivå

entity ROM\_VHDL is

port

(

clk\_50, CS\_ROM\_n : in std\_logic;

addr : in std\_logic\_vector(7 downto 0);

data\_out : out std\_logic\_vector(15 downto 0)

);

end entity;

architecture rtl of ROM\_VHDL is

-- Build a 2-D array type for the RoM

subtype word\_t is std\_logic\_vector(15 downto 0);

type memory\_t is array(0 to 15) of word\_t;

signal rom : memory\_t := memory\_t'(

X"0000", -- Adress 0; NOP

X"100A", -- Adress 1; LOAD\_R0 #A

X"2010", -- Adress 2; STORE\_R0 #10

X"1001", -- Adress 3; LOAD\_R0 #1

X"2010", -- Adress 4; STORE\_R0 #10

X"3001", -- Adress 5; JMP #1

X"0000",

X"0000",

X"0000", -- Adress 8

X"0000",

X"0000",

X"0000",

X"0000", -- Adress 12

X"0000",

X"0000",

X"0000"); -- Adress 15

begin

process(clk\_50)

begin

if(rising\_edge(clk\_50)) then

data\_out <= rom(to\_integer(unsigned(addr(3 downto 0))));

end if;

end process;

end rtl;

## 4.3 LED – komponent

**Komponent:**

**Instansnamn:**

### 4.3.1 Funktion och arkitektur

### 4.3.2 In/utgångar

### 4.3.3 RTL-nivå

### 4.3.4 VHDL-kod

## 4.4 Adressbussdekoder – komponent

**Komponent:**

**Instansnamn:**

### 4.4.1 Funktion och arkitektur

### 4.4.2 In/utgångar

### 4.4.3 RTL-nivå

### 4.4.4 VHDL-kod

## 4.5 Ingångsfilter – komponent

**Komponent:**

**Instansnamn:**

**Generisk parameter:**

### 4.5.1 Funktion och arkitektur

Ingångsfiltret används för att generera en klocksignal till CPU. Klocksignalen kan väljas som intern eller manuell klockning med tryckknapp. Om Use\_Manual\_Clock = 0 används kortets klocka annars används tryckknapp KEY0 på kortet. Filtret använder en generic, cnt\_high för att bestämma hur många klockpulser som behövs för att signalen ska anses som stabil.

o.s.v.

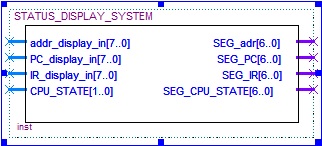
## 4.6 Status display – komponent

**Komponent:**

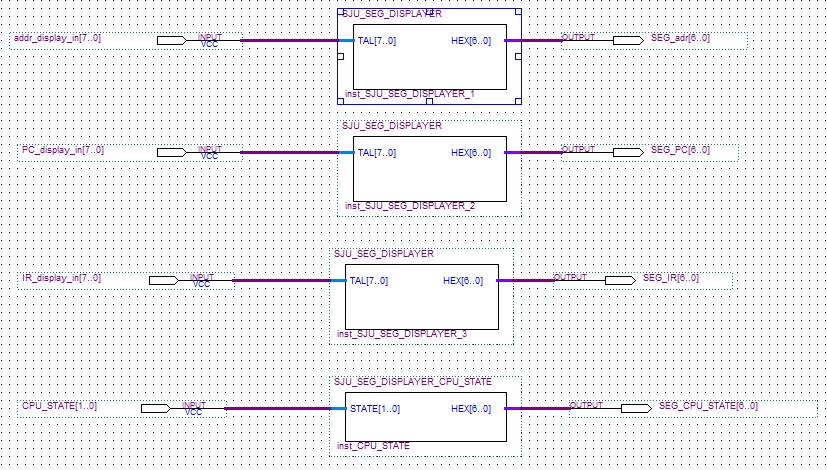
**Instansnamn:**

### 4.6.1 Funktion och arkitektur

Status displaysystemet presenterar adressbussen, programräknaren, instruktionsregistret och CPU:ns state på fyra 7-segmentdisplayer. På DE2-115 kortet finns det, om det inte finns på ett kort så går de inte att koppla in. Status display är uppdelad i fyra olika delsystem. CPU state (SJU\_SEG\_DISPLAYER\_CPU) är en del. De övriga tre är instansieringar av en komponent (SJU\_SEG\_DISPLAYER). I Figur 10 och Tabell 8 visas in och utsignalerna i status display system. Subsystemen visas i Figur 11. Hur informationen presenteras visas i Tabell 9.



Figur 10. Status display symbol



Figur 11. Status display system subsystem (arkitektur)

### 4.6.2 In/utgångar

Tabell 8. In- och utsignalerna i status display system

|  |  |  |  |
| --- | --- | --- | --- |
| **Signal** | **Namn** | **Riktning** | **Typ** |
| Adressbuss | Addr\_display\_in | in | std\_logic\_vector(7 downto 0) |
| Programräknare | PC\_display\_in | in | std\_logic\_vector(7 downto 0) |
| Instruktionsregistret | IR\_display\_in | in | std\_logic\_vector(7 downto 0) |
| CPU-tillstånd (STATE) | CPU\_state | in | std\_logic\_vector(1 downto 0) |
| Adressbussens värde omtolkat  till 7-segmentinformation | SEG\_adress | ut | std\_logic\_vector(6 downto 0) |
| Programräknarens värde omtolkat  till 7-segmentinformation | SEG\_PC | ut | std\_logic\_vector(6 downto 0) |
| Instruktionsregistrets värde omtolkat  till 7-segmentinformation | SEG\_IR | ut | std\_logic\_vector(6 downto 0) |
| CPU-tillstånd (STATE) värde omtolkat  till 7-segmentinformation | SEG\_CPU\_STATE | ut | std\_logic\_vector(6 downto 0) |

Tabell 9. Information på 7-segmentdisplayerna

|  |  |  |
| --- | --- | --- |
| Namn | Presenterad information | Display på DE2-115 |
| SJU\_SEG\_DISPLAYER\_1 | Adressbuss | HEX 0 |
| SJU\_SEG\_DISPLAYER\_2 | Programräknare | HEX 3 |
| SJU\_SEG\_DISPLAYER\_3 | Instruktionsregistret | HEX 4 |
| CPU\_STATE | CPUns tillstånd | HEX 6 |

## 4.7 Sju\_seg\_displayer – komponent

**Komponent:** SJU\_SEG\_DISPLAYER

**Instansnamn:** inst\_SJU\_SEG\_DISPLAYER\_1, inst\_SJU\_SEG\_DISPLAYER\_2 och inst\_SJU\_SEG\_DISPLAYER\_3

### 4.7.1 Funktion arkitektur

Komponenten SJU\_SEG\_DISPLAYER\_1 visar adressbussen på 7-segmentdisplayen hex0. Insignalen TAL omtolkas till utsignalen HEX så att 7-segementdisplayen visar TALs värde. Displayen uppdateras då TAL ändrar tillstånd. In- och utsignalerna i sjusegment display 1 visas i Figur 12 och Tabell 10. RTL-nivån för sjusegmentdisplayen visas i Figur 13.



Figur 12. Sjusegment display 1

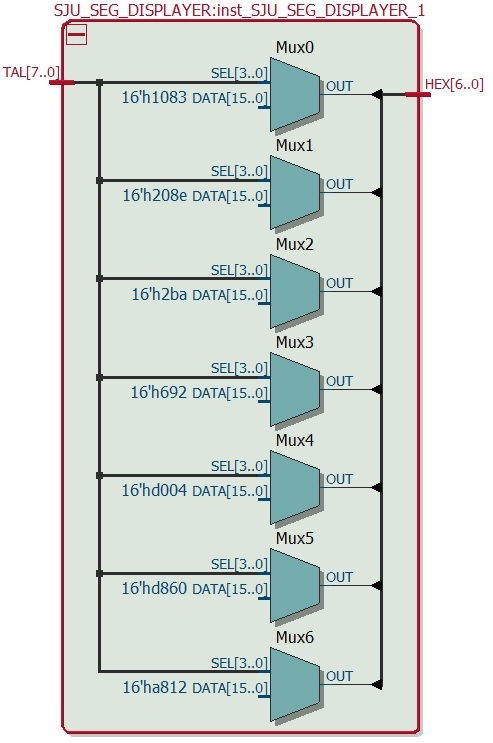
### 4.7.2 In/Utgångar

#### 

Tabell 10. Signaler för presentation av adressbussen

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Signal** | **Extern namn** | **Internt namn** | **Riktning** | **Typ** |
| Adressbussen | addr\_display\_in | TAL | in | std\_logic\_vector(7 downto 0) |
| Styrdata till HEX6 | SEG\_adr | HEX | ut | out std\_logic\_vector(6 downto 0) |

### 4.7.3 RTL-nivå



Figur 13. RTL-nivån för sjusegmentdisplay 1

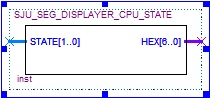
## 4.8 Sju\_seg\_displayer\_CPU\_STATE

**Komponent:**

**Instansnamn:**

### 4.8.1 Funktion arkitektur

Enheten CPU\_STATE presenterar datorns tillstånd på 7-segment display HEX6. Displayen uppdateras då STATE ändrar tillstånd. In- och utsignaler för komponenten CPU\_STATE visas i Figur 14 och tabell 11. Förklaring av text på HEX6 visas i Tabell 11. RTL-nivån på CPU\_STATE visas i figur?



Figur 14. CPU state

Tabell 11. CPUns tillstånd

|  |  |
| --- | --- |
| **Tillstånd** | **HEX6** |
| FETCH | F |
| DECODE | D |
| EXECUTE | E |
| STORE | S |
| ERROR | 8 |

### 4.8.2 In/Utgångar

### 4.8.3 RTL-nivå

### 4.8.4 VHDL-kod

# 5 GRANSKNING OCH FÖRSLAG PÅ FÖRBÄTTRINGAR

Följande förslag på förbättringar eller övervägande till förbättringar av VHDL koden.

**Komponent:**

Simple\_VHDL\_CPU

3. ….

ROM\_VHDL

….

# 6 FOOT PRINT

Skriv ut storleken och hur stor plats den tar på chippet

# 7 KOSTNAD FÖR PROJEKTET

Upparbetade timmar multiplicerat med timtaxan.