VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Turo Id

MIKROPROSESSORIJÄRJESTELMÄN SUUNNITTELU FPGA:LLA

Tekniikka
2018

VAASAN AMMATTIKORKEAKOULU
Tietotekniikka

TIIVISTELMÄ

Tekijä	Turo Id
Opinnäytetyön nimi	Mikroprosessorijärjestelmän suunnittelu FPGA:lla
Vuosi	2018
Kieli	suomi
Sivumäärä	41
Ohjaaja	Santiago Chavez Vega

Opinnäytetyön ideana oli luoda Vaasan ammattikorkeakoulun tietotekniikan linjalle ohjeistusmalli mikroprosessorijärjestelmän suunnittelusta FPGA:lla. Opinnäytetyön aihe valittiin yhteistyössä Vaasan ammattikorkeakoulun tietotekniikan linjan opettajan kanssa.

Opinnäytetyön kokeellinen osuus suoritettiin tutustumalla laitteella erilaisten töiden toteuttamiseen, joista valittiin mielenkiintoinen aihe. Aihe valittiin uskoen, että se innostaa myös tulevia opiskelijoita, jotka käyttävät tehtyä työtä apunaan. Aiheeseen tutustumiseen käytettiin apuna myös Vaasan ammattikorkeakoulun käyttämiä opetusmateriaaleja.

Ohjeistus on mietitty tuleville oppilaille ja asiasta kiinnostuneille, jotka haluavat tutustua ja kehittyä asian tiimoilla. Työssä käydään läpi laitteiston ja ohjelmistojen historiaa sekä toimintaa. Työn lopussa käydään kokeellisen työn avulla ohjelmiston toimintoja läpi.

Opinnäytetyön lopullinen tulos on kattava ja ohjeistus on tehty tarkasti jokainen kohta esiteltynä. Lopputuloksessa on otettu huomioon, että tulevalla työn suorittajalla ei välttämättä ole aikaisempaa kokemusta aiheesta. Opinnäytetyön tulokset antavat hyvän pohjan tuleville FPGA-projekteille.

Avainsanat mikroprosessorijärjestelmä, FPGA, suunnittelu, opetus

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Information technology

ABSTRACT

Author	Turo Id
Title	Design of microprocessor system on FPGA
Year	2018
Language	Finnish
Pages	41
Name of Supervisor	Santiago Chavez Vega

The thesis was to create a microprocessor system design FPGA instructional model for IT line of Vaasa´s University of Applied Sciences. The thesis was selected in co-operation with the teacher of Information technology Embedded system at Vaasan University of Applied Sciences.

The experimental part of the thesis's was performed by exploring the device for a variety of work, of which an interesting subject was chosen. The subject was chosen, believing that it would also inspire future students who use the work done. Teaching materials were also used at Vaasa University of Applied Sciences.

Guidance on the future educational institutions and interested parties who wish to become acquainted with and develop on relevant issues. This thesis examines the history and activities of hardware and software. At the end of the work, experimental work is carried out through the functions of the software.

The result of the Thesis is comprehensive, and the guidance is done carefully each point carefully presented. In the end, it has been taken into account that the future work supervisor may not have previous experience on the subject. The results of the thesis give a good foundation for future FPGA projects.

Keywords microprocessor system, FPGA, design, teaching

SISÄLLYS

KUVALUETTELO

TAULUKKOLUETTELO

1 JOHDANTO

Tämä opinnäytetyö tehdään Vaasan ammattikorkeakoulun Tietotekniikan ”Sulautetut järjestelmät”-linjalle. Opinnäytetyön aihe valittiin yhdessä tietotekniikan linjan opettaja Santiago Chavez Vegan kanssa. Opinnäytetyö vastaa Tietotekniikan ”Sulautettujen järjestelmä”-linjan toiveita, jonka myötä opinnäytetyötä voitaisiin tarvittaessa hyödyntää opetuksessa.

Opinnäytetyön aihe on tulevaisuuden kannalta suhteellisen tärkeä, sillä teknologian kehittyessä alkaa yhä useampi tekniikan alan yritys siirtyy käyttämään tätä tekniikkaa. Sen myötä sitä on hyvä opetella ja opettaa kouluissa jo hyvissä ajoissa, jotta tulevat alan tekijät osaavat toimia näiden laitteiden kanssa.

Opinnäytetyön alussa tutustutaan tekniikan, laitteen ja suurimpien vaikutusten, sekä vaikuttajien historiaan ja toimintaan. Loppuvaiheessa käydään projektin tekovaiheen yhteydessä läpi mitä laitteita ja ohjelmistoja tarvitaan mikroprosessorin suunnittelussa, kun se tehdään FPGA:lla. Vaiheet näytetään tarkasti, jotta ne olisivat jokaiselle helposti ymmärrettävissä ja opittavissa.

2 FPGA

Tässä luvussa käydään läpi mitä FPGA (Field-programmeble gate array) eli helposti uudelleenohjelmoitavan logiikan sisältävä digitaalinen mikropiiri on ja millainen sen rakenne on, sekä miten ja missä sitä voidaan käyttää. Tässä työssä käytetään Altera DE2 FPGA-piiriä./1/

2.1 Historia

FPGA-piirien kehitys on lähtöisin aikaisemmista ohjelmoitavista logiikkapiireistä esim. CPLD- ja PAL-piireistä 1980-luvulla. /1/ Altera on yksi johtavista ja yrityksistä, joka valmistaa FPGA-piirejä ja se julkisti ensimmäisen PLD-piirinsä (Programmable Logic Devices) vuonna 1984, joka oli vuosi yrityksen perustamisen jälkeen. PLD-piiri oli ensimmäisiä askelia kohti nykyistä FPGA-piiriä./2/

2.2 Rakenne

FPGA-piirin väliset kytkennät ja kombinatorinen logiikka ovat ohjelmoitavissa, jotka muodostuvat geneerisistä logiikkaelementeistä. FPGA-piirin toteuttama huipputoiminnallisuus ilmoitetaan logiikkaporttimäärällä, joka vastaa toiminnallisuuden huippua. Sadat tuhannet portit laskevat tämän määrän nykyaikaisissa piireissä. FPGA-piirien valmistajien ilmoittama määrä on kuitenkin vain laskennallinen maksimimäärä, kun todellisuudessa käytännössä tästä jäädään useiden kymmenien prosenttien päähän. Se johtuu pitkistä reititysmatkoista, jotka syntyvät siitä, että optimaalisesti on erittäin vaikeaa tehdä kaikista logiikkaelementtien välisistä reiteistä lyhyitä./1/

Vaikka FPGA-piirit ovat helposti uudelleenohjelmoitavia, ne voivat myös sisältää aritmeettisen sarjan, jota käytetään erilaisiin operaatioihin, digitaalisiin suotimiin, sekä vastaavanlaisia yksinkertaisia piirejä, esim. muistia tai kokonaisia ohjelmoitavia suorittimia, jotka ovat yleiskäyttöisiä./1/

2.3 Toiminta ja käyttö

2.3.1 Toiminta

FPGA-piirejä verrataan yleensä ASIC (sovelluskohtaiseen mikropiiriin)-piiriin, koska näiden kahden piirin toiminta on samankaltaista ja niitä sovelletaan usein samoilla alueilla. Ero näiden kahden piirin välillä näkyy tehon kulutuksessa, lämpenemisessä, koossa ja kellotaajuuden suuruudessa. Vaikka nämä kaikki ovat parempia ASIC-piirissä, niin se on hitaampi ja haastavampi suunnitella ja ottaa käyttöön kuin FPGA-piiri./1/

FPGA-piirissä käytettävä malli kirjoitetaan ohjelmointikielen tapaisella tietokonekielellä ja sen jälkeen se automatisoidaan niin pitkälle, että se syötetään pieneltä oheispiiriltä, johon se on tallentunut. FPGA-piirin käynnistyessä se vapauttaa mallin oheispiiristä itseensä ja tämän jälkeen FPGA-piiri alkaa toimia itsenäisesti./1/

2.3.2 Käyttö

FPGA-piirin muokattavuuden takia sitä on helppo ja hyvä käyttää erilaisissa prototyypeissä, sekä testausvaiheissa. Testaamisen jälkeen, kun virallinen laite on valmis, FPGA-piiri voidaan vaihtaa ASIC-piiriin, joka toimii laitteessa virallisena piirinä. Tällä toiminnalla säästetään paljon aikaa ja rahaa. FPGA-piiriä käytetään ASIC-piirin prototyyppinä olemisen lisäksi, kun halutaan säilyttää kyky uudelleenohjelmoimiseen tai tuote halutaan saada nopeasti kuluttajien käytettäväksi./1/

3 ALTERA QUARTUS II

Tässä luvussa käydään läpi mikä Altera Quartus on ja sen historiaa, sekä sen käyttöä ja mitä ominaisuuksia se sisältää. Tässä työssä käytetään Quratus II:tta ja tarkemmin versiota 13.0.

3.1 Historia

Quartus II on Intelin luoma logiikkalaitteiden suunnitteluohjelma, jossa voidaan ohjelmoida. Ohjelman nimi oli ennen Intel Quartus Prime, mutta se muutettiin nykyiseen nimeen (Altera Quartus II), kun Altera osti Intelin./3/

3.2 Käyttö

Quartus II mahdollistaa kehittäjälle synteesin, jonka tuella käyttäjä kykenee luomaan mallinsa, suorittamaan ajoitusanalyysin, tarkastelemaan RTL-kaavioita, simuloimaan, jonka avulla voidaan testata erilaisten ärsykkeiden vaikutuksen suunniteltuun reaktioon ja kohdelaitteen määrittelyä, sekä analysointia HDL-malleille./3/

3.3 Ominaisuudet

Quartus sisältää VHDL:n ja Verilogin toteutuksen laitteiston kuvauksen simulointia varten, vektori-aaltomuodon ja visuaalisen muokkauksen logiikkapiiriä varten./3/

3.3.1 SOPC Builder

SOPC Builder eli ohjelmoitavan sirunmuodostusjärjestelmän valmistaja on Altera. Sillä automatisoidaan pehmeät laitteistokomponentit, mitkä liitetään tietojärjestelmän täydelliseen luomiseen ja se toimii kaikilla FPGA-siruilla. SOPC Builder sisältää mukautettujen komponenttien liittämiseen käyttöliittymään, sekä kirjastoon, mikä sisältää valmiita komponentteja. Avalon-väylää käytetään liitännöiden tekemiseen./3/

SOPC Builder-työkalu poistaa Quartus II-ohjelmistosta manuaaliset järjestelmäintegraatiotehtävät automaattisesti yhteenliittymislogiikkaa generoimalla ja testejä tekemällä tarkistaa toiminnallisuuden./3/

3.3.2 Qsys

Uuden sukupolven SOPC Builder eli Qsys on järjestelmäintegraatiotyökalu. Arkkitehtuurina se käyttää network-on-chipiä, joka on optimoitu FPGA:lla. Sillä se kaksinkertaistaa fMax-suorituskyvyn SOPC Builderiin verrattuna./3/

3.3.3 SoCEDS

SoCEDS sisältää joukon sovellusesimerkkejä, apuohjelmia, ajo-ohjelmia ja kehitystyökaluja. Näiden avulla voidaan suunnitella ja parannella SoC FPGA sulautetuille järjestelmille./3/

3.3.4 DSP Builder

DSP Builder työkalun avulla luodaan täydellinen saumatton yhteys Simulink-työkalun/MATLAB ja Quartus II-ohjelmiston välille. Näin suunnittelijoiden algoritmien kehittäminen, simulointi ja todentaminen tapahtuu Simulink-järjestelmän/MATLAB-tason suunnittelutyökaluilla./3/

3.3.5 Ulkoinen muisti

Ulkoinen muisti luo rajapinnan, jonka tehtävänä on tunnistaa kalibrointiongelmat ja mitata marginaalit jokaisesta DQS-signaalista./3/

3.3.6 JAM/STAPL-tiedosto

JAM/STAPL-tiedostojen tehtävä on luoda tiedostoja JTAG-piirin laitteille./3/

4 NIOS II

Tässä luvussa käydään läpi mikä Nios II on ja mitkä ovat sen avainominaisuudet./6/

Nios II on Altera-perheen FPGA-piireille erityisesti suunniteltu 32-bittinen sulautettu prosessoriarkkitehtuuri. Sen sisältämän laajan arkkitehtuurin ansiosta se sopii useaan tietojenkäsittelysovellukseen, jotka ovat sulautettuja./6/

Nios II:n käyttämän RISC-soft-core -arkkitehtuurin ansiosta se pystyy toteuttamaan FPGA:n ohjelmoitavan logiikan ja muistilohkot./6/

4.1 Avainominaisuudet

Ennalta määritellyn muistinhallintayksikön tai mukautettujen oheislaitteiden ja mukautettujen ohjeiden avulla voidaan laajentaa Nios II:n perustoimintoja./6/

4.1.1 Muistinhallintayksikkö

Laitteistopohjaiset hakua ja suojausta edellyttäviä käyttöjärjestelmiä pystytään käyttämään Nios II:lla valinnaisen MMU:n avulla. Ilman sitä Nios II: käyttö rajoittuu yksinkertaistettua suojausta ja virtuaalimuistimallia käyttäviin laitteisiin./6/

4.1.2 Mukautetut oheislaitteet

Mukautetut oheislaitteet ovat useampaa CPU-sykliä käyttäviä suorituskykyisiä järjestelmiä, jotka toteuttavat tietyn osan koodista. Oheislaitteet voidaan määrittää vapauttamaan kaikki tai osan ohjelmistoalgoritmin suorituksesta määritellylle ohjelmistohallinnolle kohentaen näin sovelluksen läpäisykykyä tai tehoa./6/

4.1.3 Mukautetut ohjeet

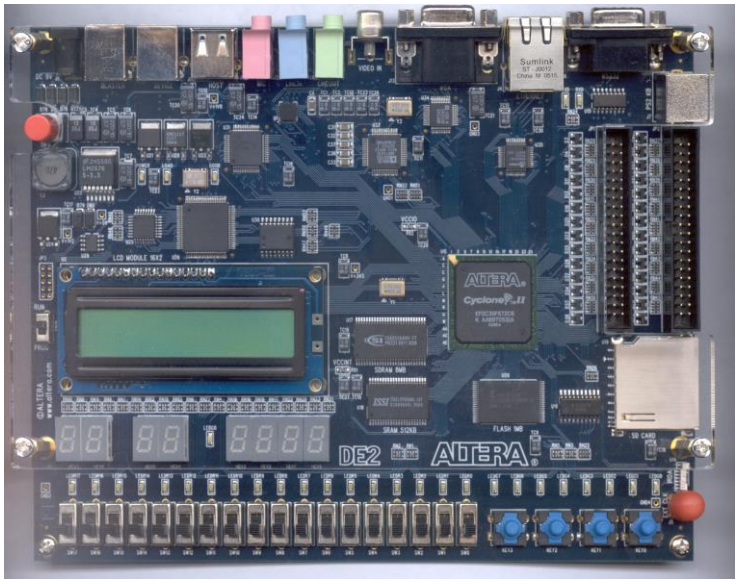
Mukautettuja ohjeita voidaan ohjata C:ssä makrojen avulla. Mukautettujen ohjeiden ansiosta voidaan järjestelmän laitteet hienosäätää täyttämään suorituskykyta-

voitteet, jonka arvot hyväksytään enintään kahdesta 32-bittisestä lähdekirjastosta ja palauttamaan valinnaisesti tuloksen 32-bittiseen kohdekirjastoon./6/

5 LAITTEET JA OHJELMAT

5.1.1 FPGA-piiri Altera DE2

Kun suunnitellaan sulautettua järjestelmää käyttäen apuna FPGA:ta, jolloin yleisin käytetty piiri on Altera DE2-piiri, joka on Altera Cyclone II EP2C35F672 FPGA:lla varustettu piiri. Sen sisältä löytyy seuraavia komponentteja: kelloja, kytkimiä, I/0-liittimiä, LED-valoja ja muisti. Kuva 1 näyttää tässä työssä käytetyn Altera De2-levyn ulkonäön./4/



Kuva 1. Altera DE2-piiri./5/

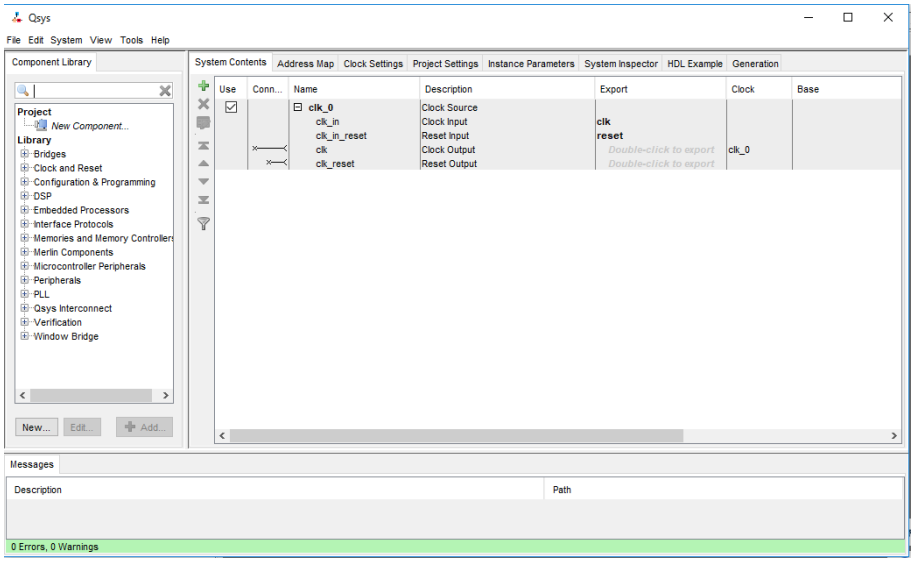
5.1.2 Altera Quartus II

Altera Quartus II on erittäin kattava ohjelmisto, joka tukee kaikkia mallinuksen menetelmiä ja sillä saadaan CAD-järjestelmiin kuvaukset ja FPGA-laitteisiin konfiguroinnit tehtyä. Kuva 2 näyttää tässä työssä käytetyn Altera Quartus II:n ohjelmiston aloitusnäytön./4/

Kuva 2. Altera Quartus II:n aloitusnäkymä

5.1.3 Qsys

Qsys on järjestelmäintegraatiotyökalu, jonka avulla lisätään projektin sisältämät ohjelmoitavat komponentit. Niitä tässä työssä on kello, neljä vihreää lediä, kaksi I/O-porttia ja muisti sekä Nios II. Kuva 3 näyttää tässä työssä käytetyn Qsysin aloitusnäkymän.

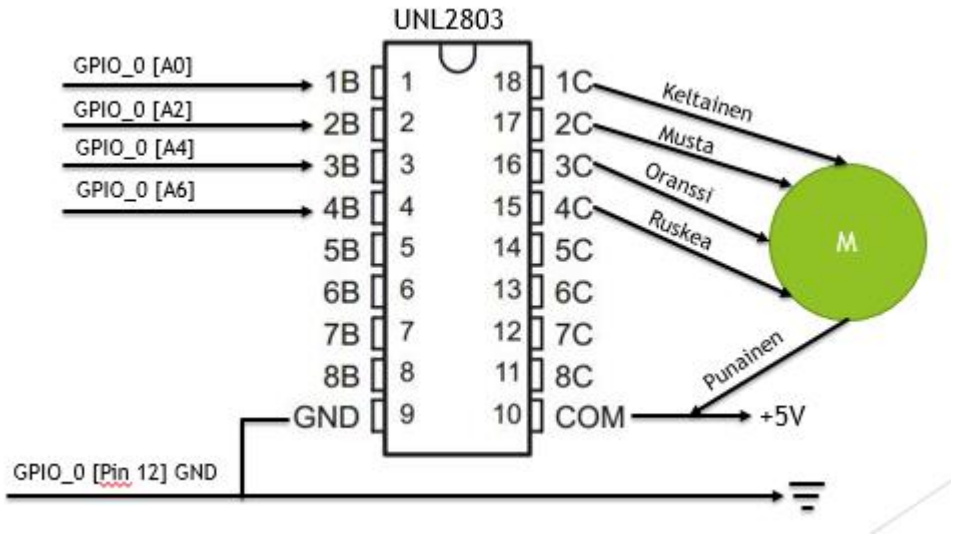


Kuva 3. Qsysin aloitusnäkymä

6 KYTKENTÄ JA OHJELMOINTI

6.1 Kytkennät

Kytkentä tehdään FPGA:n ja moottorin välille. Kytkennälle luodaan kytkentäkuva, joka helpottaa kytkennän tekemistä. Siitä nähdään kuinka FPGA:n kautta tulevat datajohdot kuuluu kytkeä, jotta saadaan FPGA:lla ohjelmoitavat koodit moottorille. Moottorin ja FPGA:n välillä käytetään UNL2803 mikropiiriä, joka toimii suojana ja puskurina. Kuva 4 näyttää tämän työn kytkennän.



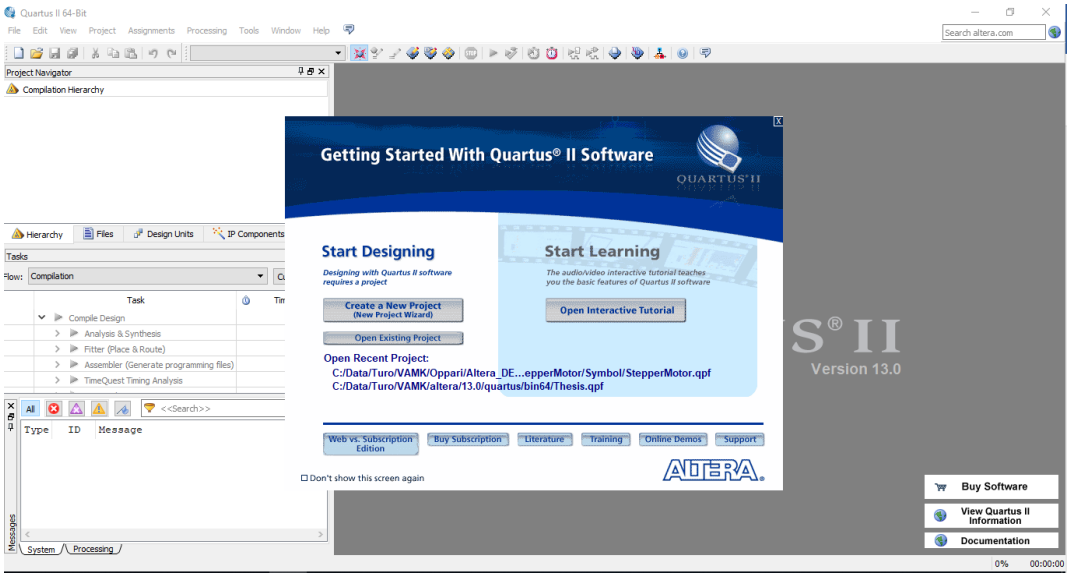
Kuva 4. Kytkentäkuva /8/

6.2 Ohjelmointi

Ohjelmoinnilla luodaan koodi, jossa määritellään toiminnot mitkä halutaan projektissa suorittaa.

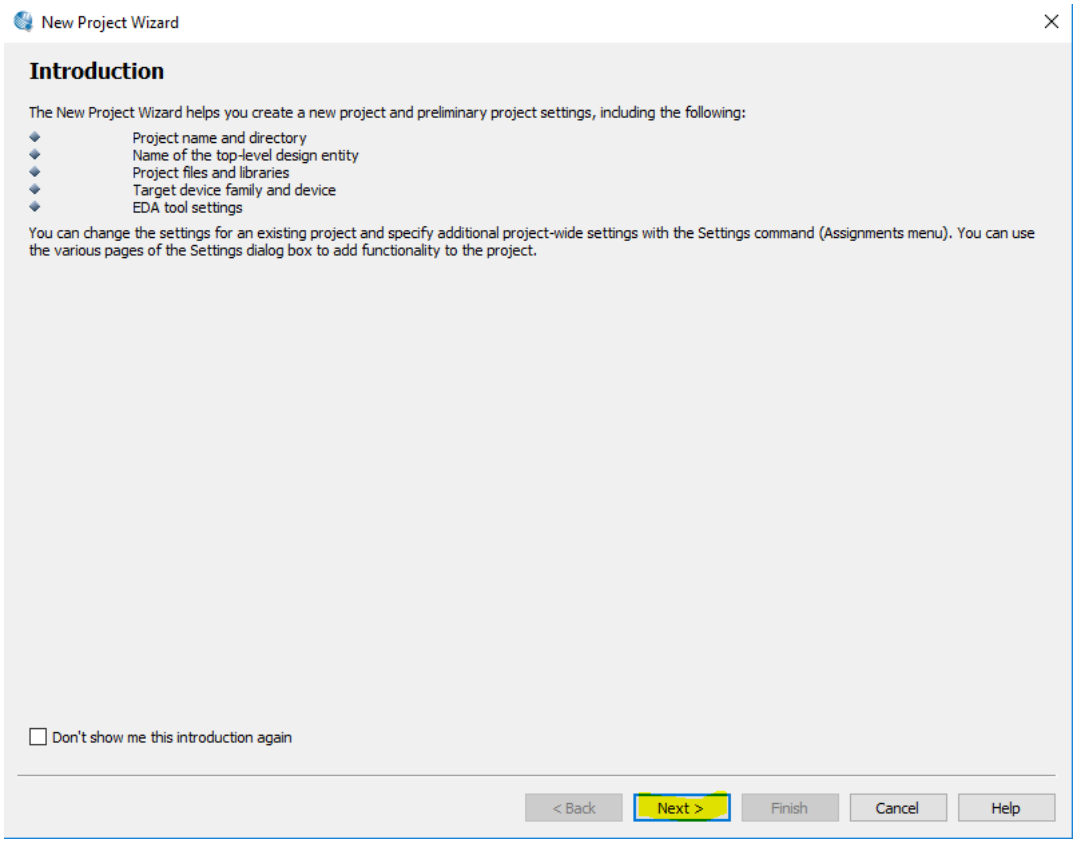
6.2.1 Altera Quartus II

Quartus II:een luodaan projekti, joka sisältää erilaisia suunnittelu- ja ohjelmointisovelluksia. Näillä suunnitellaan ohjelmiston sisäiset ohjelmoitavat kytkennät, sekä se mitä halutaan lähettää FPGA:n ja moottorin välillä. Kuva 5 näyttää tässä työssä käytetyn Altera Quartus II-aloitusnäkymän.

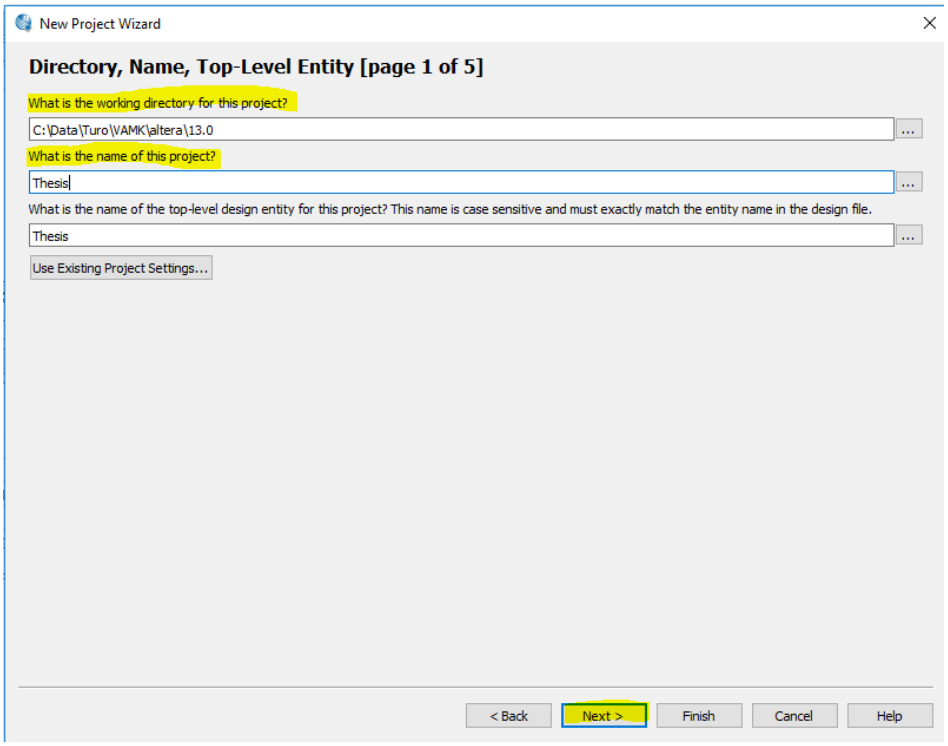


Kuva 5. Quartus II-aloitusnäkymä

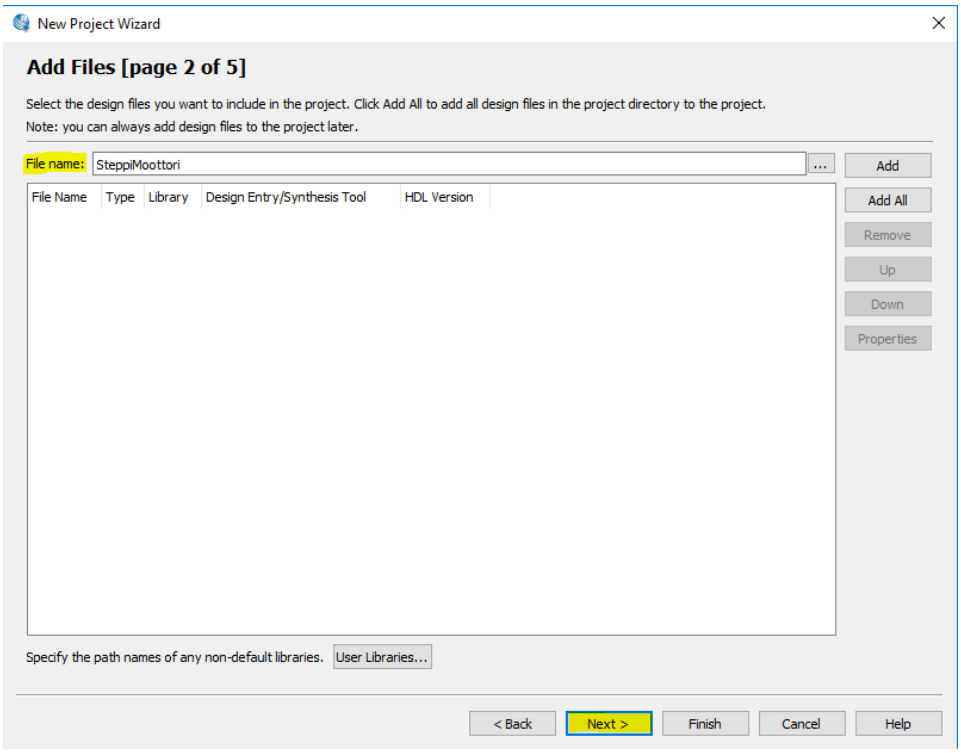
Aloittaessa uusi projekti, Quartus II ohjelmistossa valitaan Greate a New Project, jonka jälkeen seurataan kuvia 6, 7, 8 ja 9, jotka näyttävät projektin aloitus ohjeet:



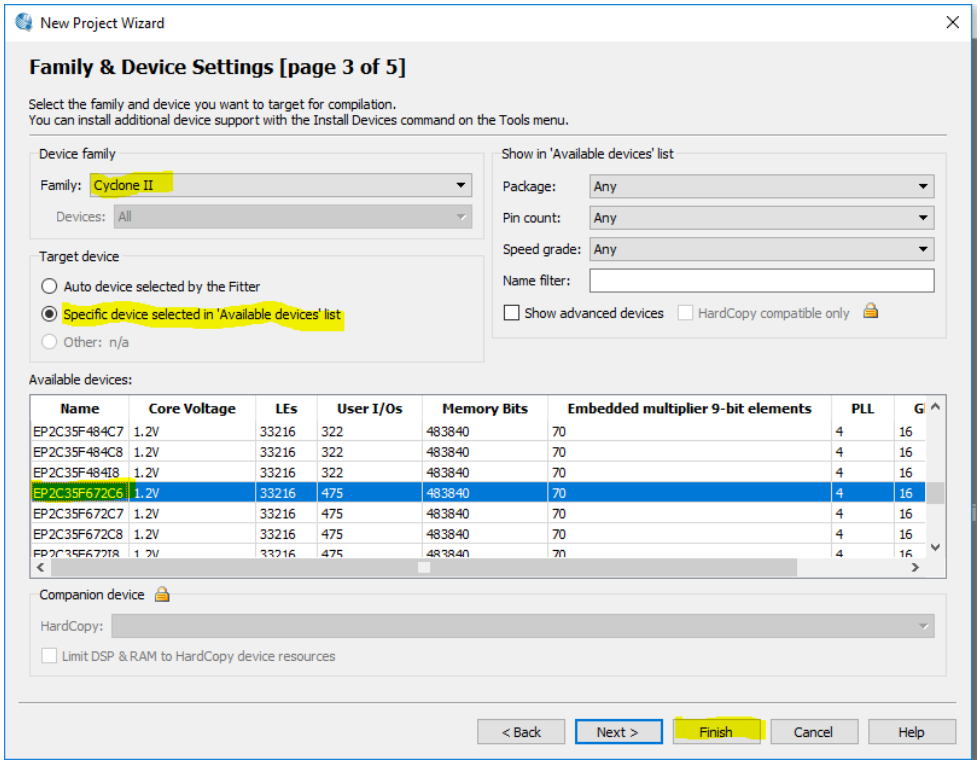
Kuva 6. Projektin aloitus



Kuva 7. Määritellään projektille kansio ja nimi

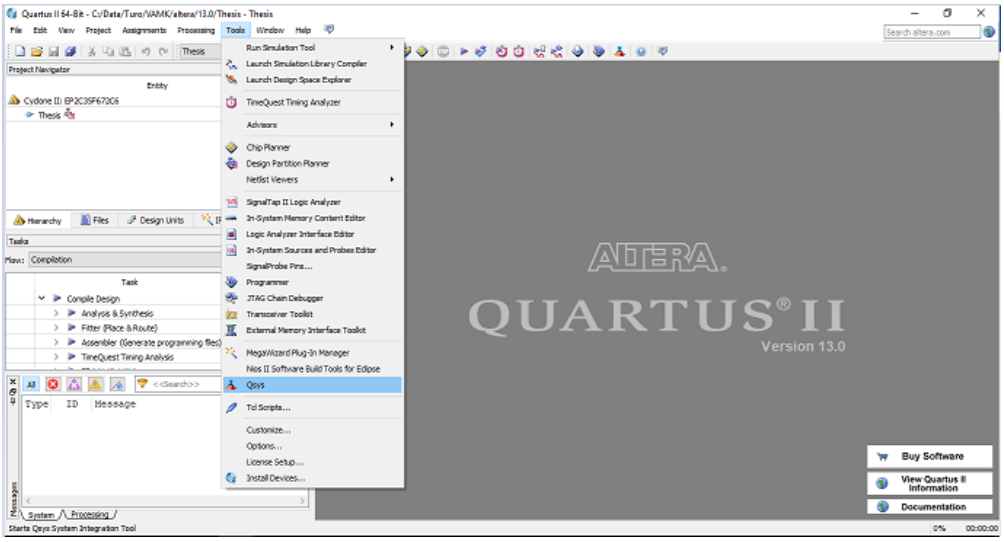


Kuva 8. Määritellään projektin tiedostolle nimi



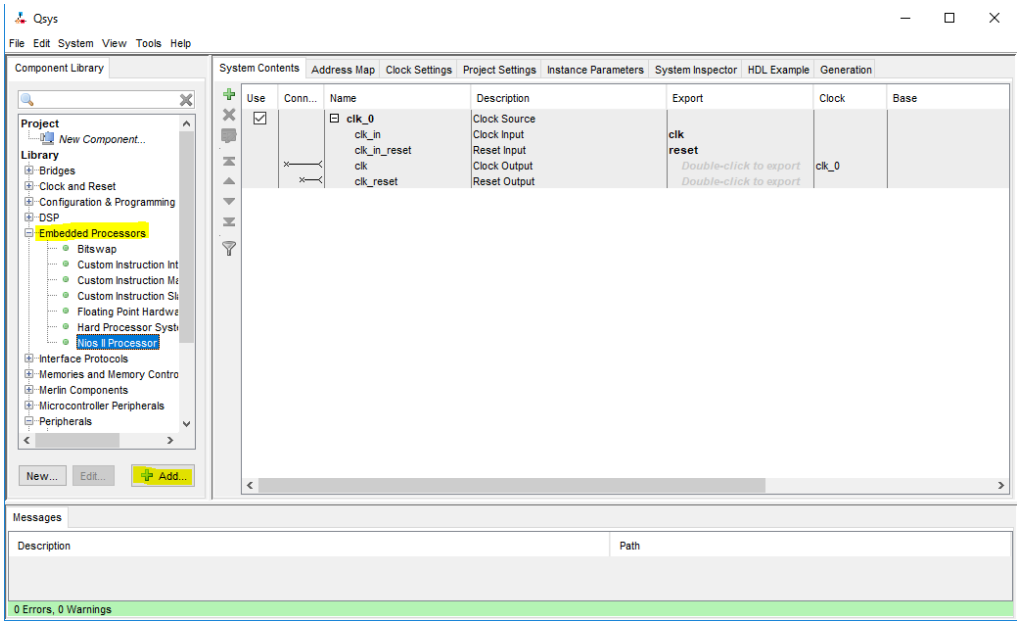
Kuva 9. Family ja Device Settings

Kuvan 9 vaiheessa määritellään projektissa käytettävä FPGA-piiri. Tässä työssä käytetään Cyclone II Family ja FPGA-piirinä EP2C35F672C6.

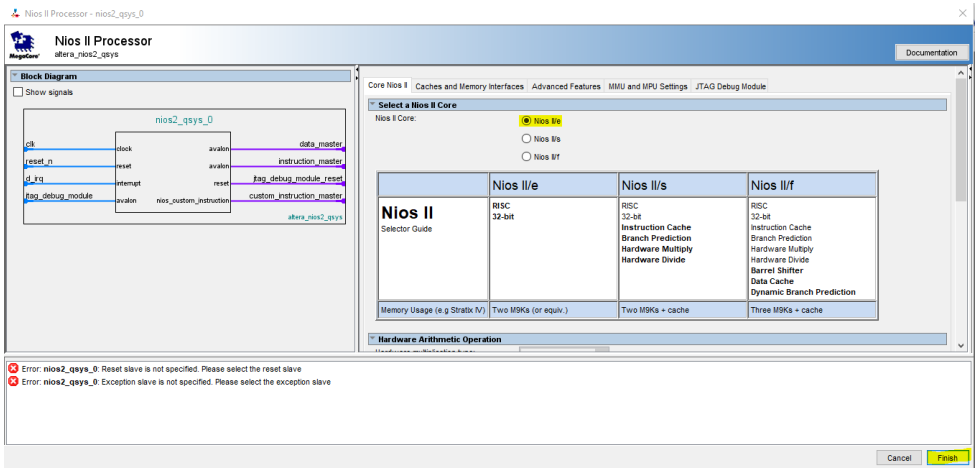


Kuva 10. Qsys-ohjelman avaaminen

Seuraavaksi avataan kuvan 10 mukaan Osys-ohjelma, jossa lisätään kuvien 11-16 mukaisesti projektissa tarvittavia komponentteja.

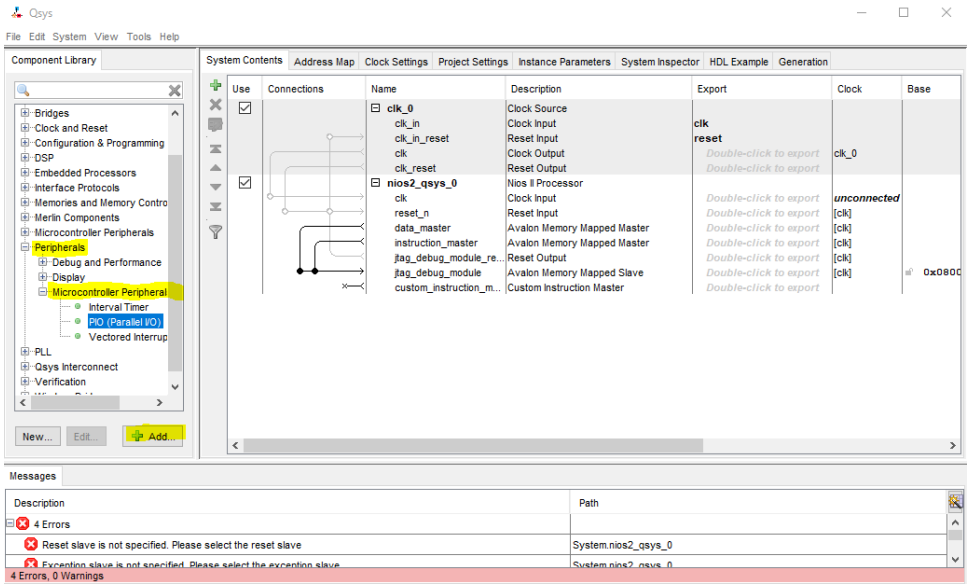


Kuva 11. Nios II-processorin lisäys

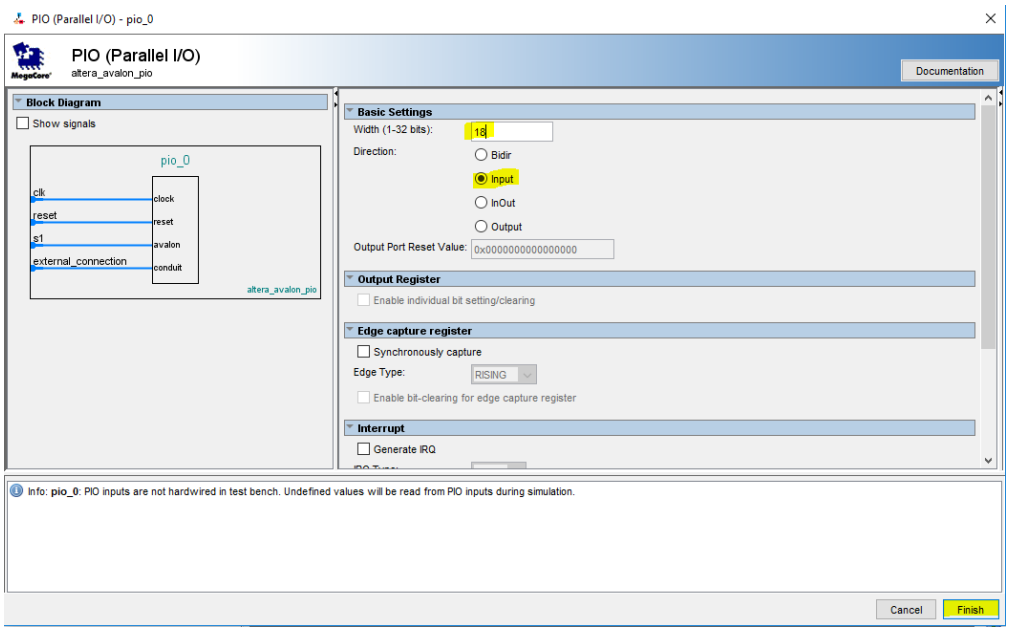


Kuva 12. Nios II-Processor

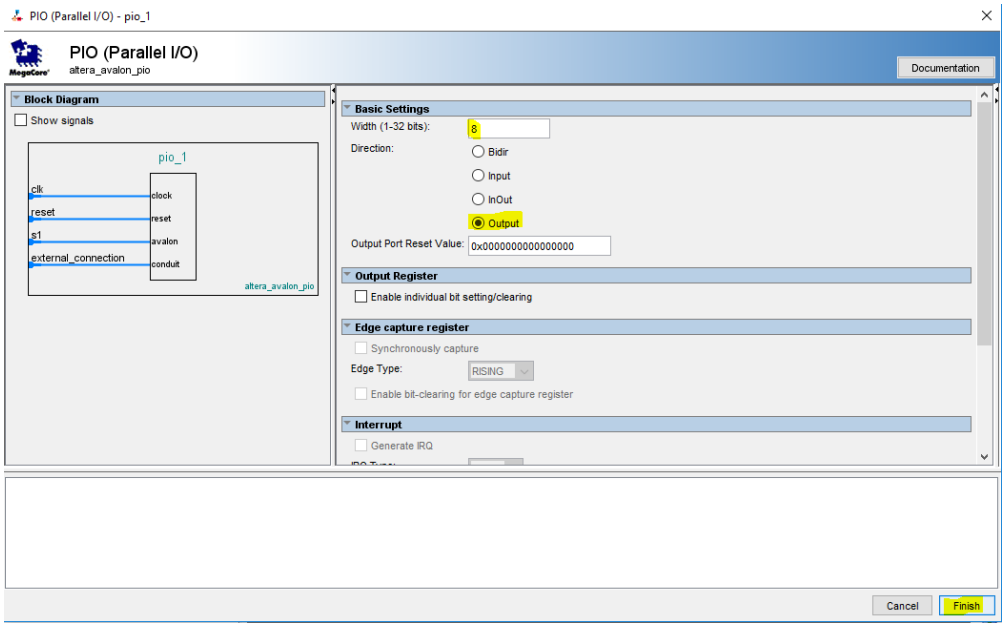
Kuvassa 12 valitaan kolmesta Nios II:n versiosta sopivin ja tässä työssä se on Nios II/e, koska se on ekologisin ja kevyin ajettava.



Kuva 13. I/O-porttien lisäys

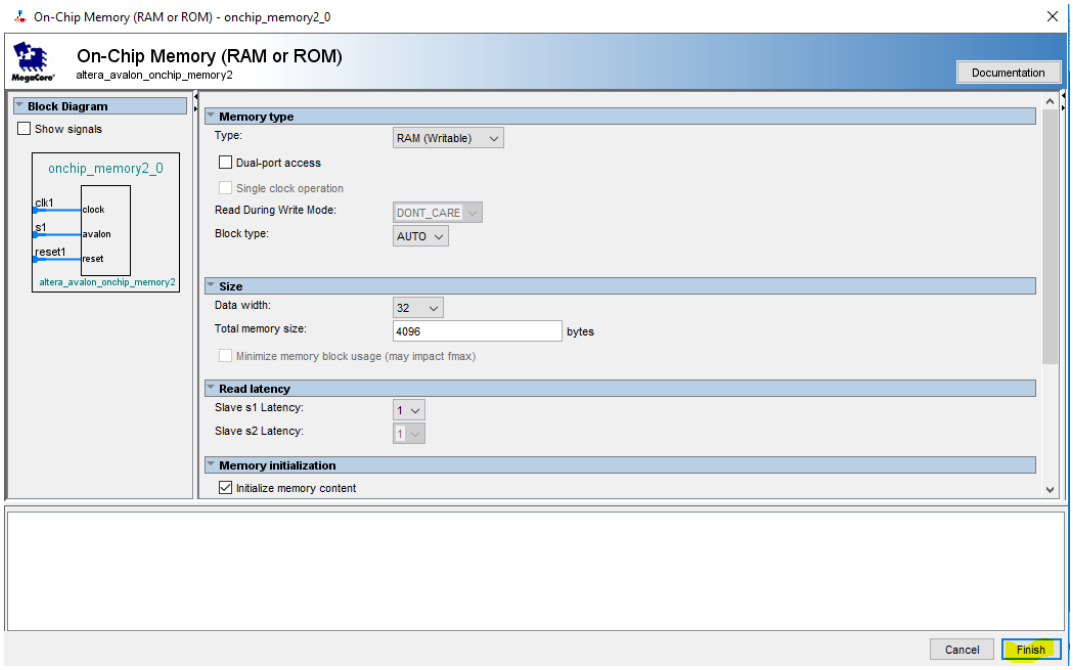


Kuva 14. PIO (Parallel I/O)-Input



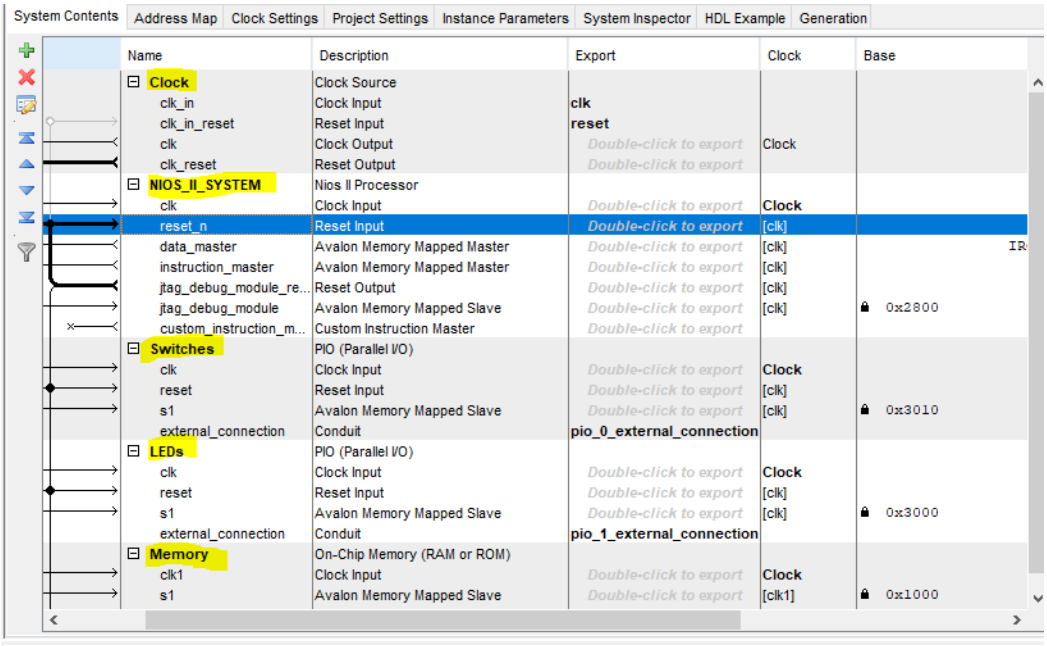
Kuva 15. PIO (Parallel I/O)-Output

PIO (Parallel I/O)-tiedostossa määritellään input ja output kuvien 14 ja 15 mukaisesti. Näistä porteista tehdään tätä projektia varten kytkin ja LED-valoja.



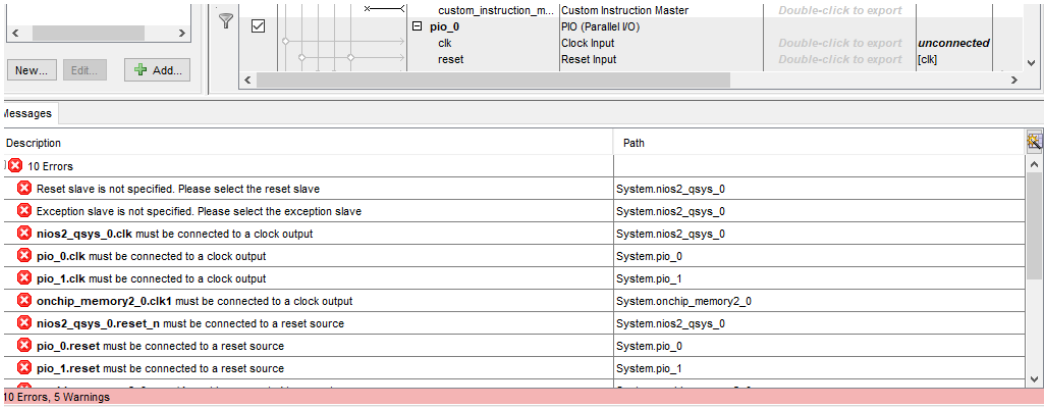
Kuva 16. Lisätään muistikomponentti

On-Chip Memory-tiedostossa lisätään muistikomponentti. Siellä voidaan tarpeen mukaan muuttaa todellisen muistin kokoa. Koko riippuu siitä, mikä muisti on käytössä, mutta tässä projektissa sitä ei muutettu. Tarvittaessa muistin bittimäärä voidaan nostaa 20 400 bittiin, ja näin sille saadaan lisää nopeutta.



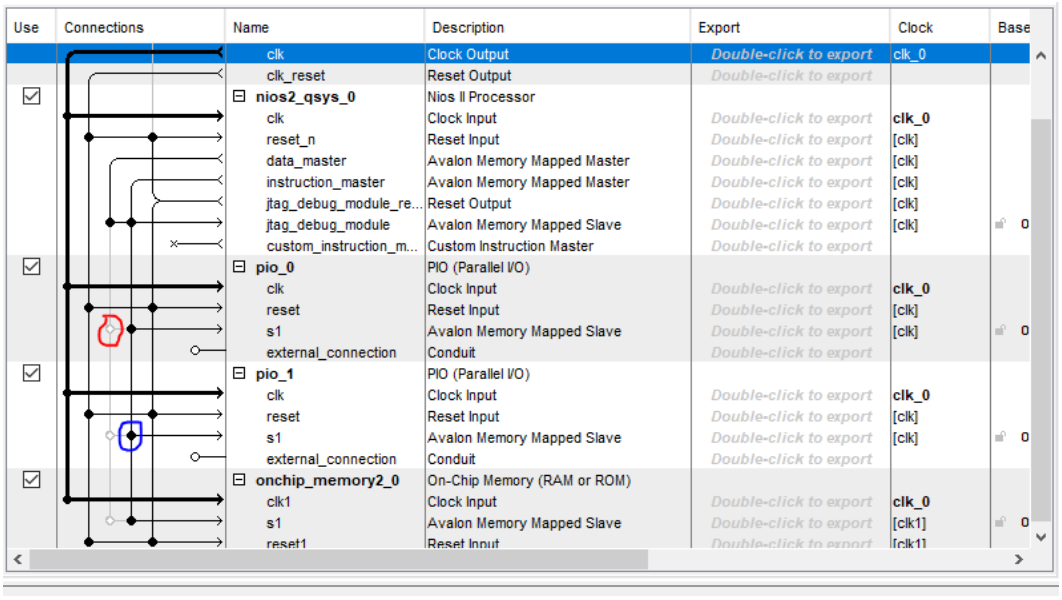
Kuva 17. Komponenttien nimeäminen

Tässä vaiheessa voidaan nimetä aikaisemmin lisätyt komponentit helpommin luettavaksi. Se onnistuu, kun painetaan muutettavan nimen kohdalla hiiren oikeaa painiketta ja valitaan sen jälkeen valikosta Rename. Muutetaan komponenteille kuvan 17 mukaiset nimet.



Kuva 18. Error-lista

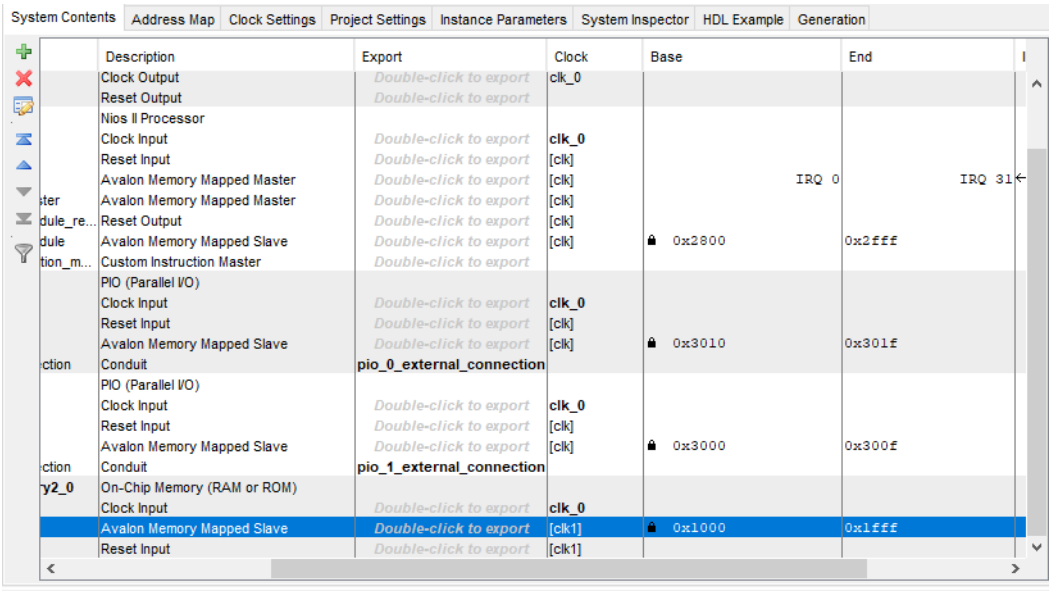
Kaikkien tarvittavien komponenttien lisäysten jälkeen voidaan tarkastella alhaalta kuvan 18 mukaisesta ikkunasta näkyviä virheitä ja suorittaa niissä lukevat asiat, joilla ne saadaan korjattua pois.



Kuva 19. Komponenttien yhdistäminen

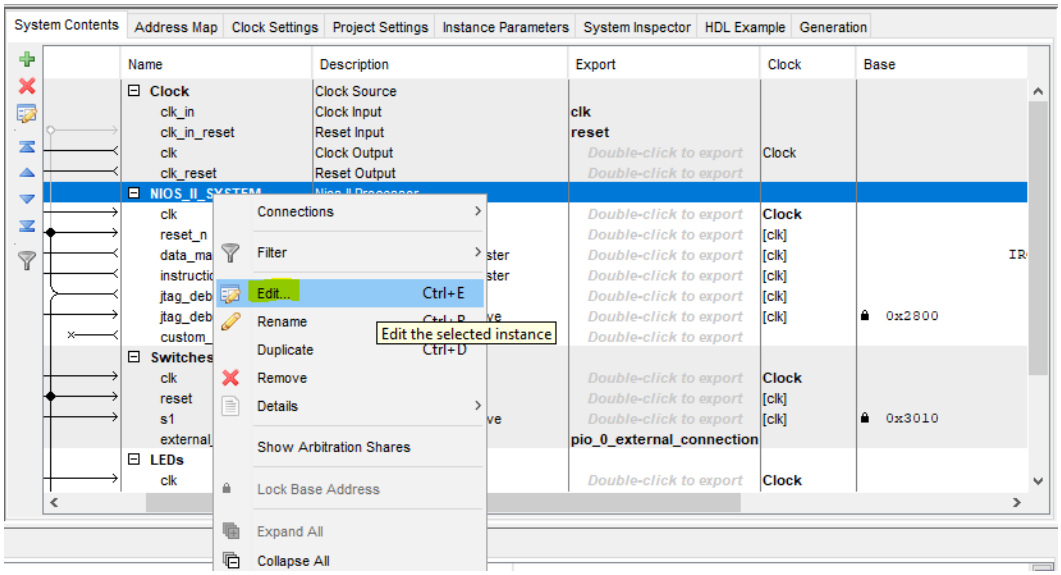
Kytketään komponentit yhteen aikaisemmin tutkitun virheluettelon avulla. Virheilmoituksen ilmoittama puutteellinen kytkös näkyy kuvan 19 punaisen ympyrän sisällä ja korjattu kytkös kuvan 19 sinisen ympyrän sisällä. Kytkös on tekemättä,

kun sen kiinnityspiste on valkoinen pallo ja sitä klikkaamalla kiinnityspiste muuttuu tummaksi palloksi, joka kertoo kytkennän tekemisestä.



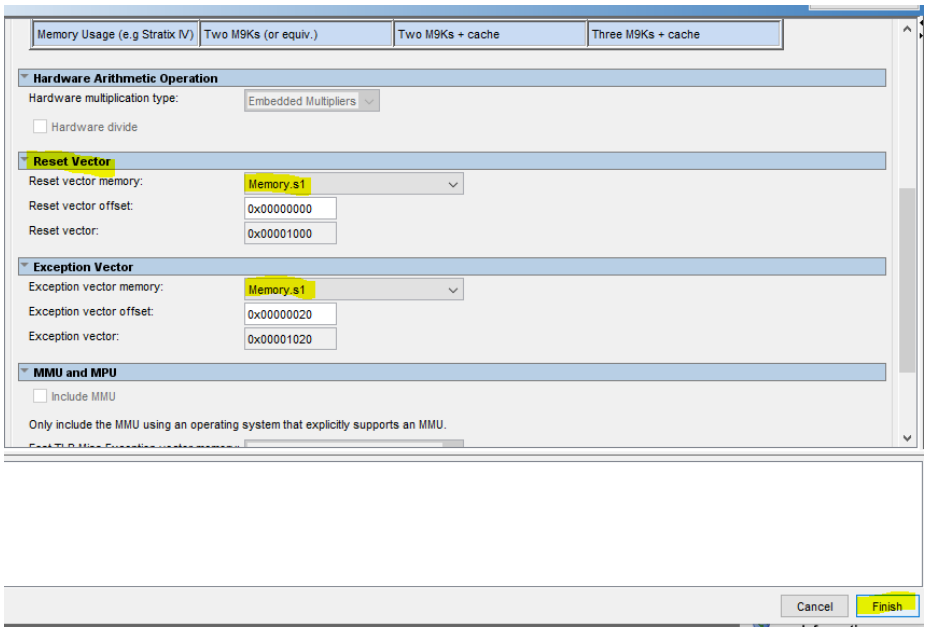
Kuva 20. Exportien ja osoitteiden lukinta

Kytkentöjen jälkeen on vielä virheitä, mutta ne saa korjattua Exportiin pio_0 ja pio_1 tuplaklikkaamalla Export-sarakkeesta komponentin riviä. Sen jälkeen muutetaan vielä kuvassa 20 näkyviin kohtiin paikat, jotka lukitaan lopuksi painamalla auki olevaa lukkoa. Tällöin se muuttuu kuvan mukaiseksi, eli kiinni olevaksi lukoksi. Tämä takaa sen, että kyseinen paikka on varattu tietylle komponentille.

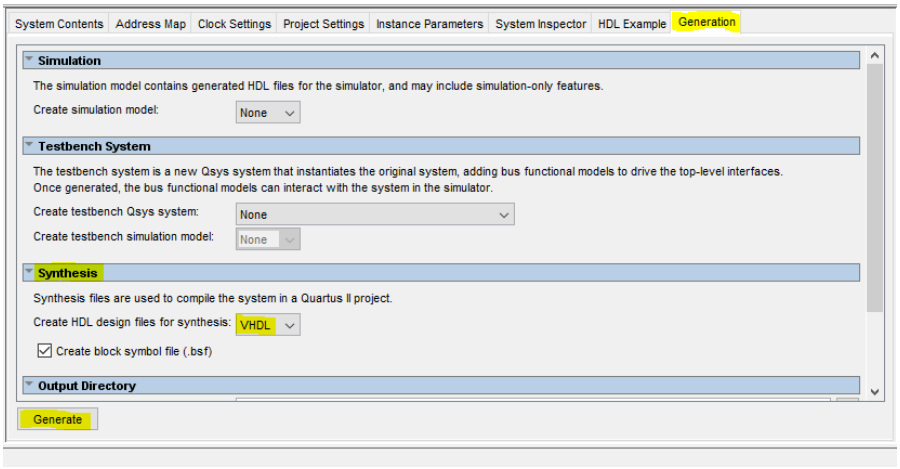


Kuva 21. Nios II Systemin muokkaus

Painetaan kuvan 21 mukaisesti NIO_II_SYSTEMin kohdalla hiiren oikeaa painiketta ja valitaan listasta Edit-kohta. Selataan Nios II Processor-listaa alaspäin, jossa muutetaan kuvan 22 mukaisesti Reset Vector ja Exception Vector kohtaan Memory s1 ja sen jälkeen painnetaan Finish.

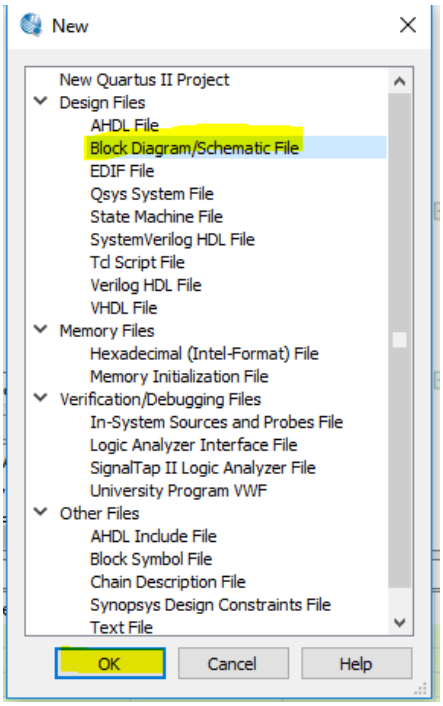


Kuva 22. Reset ja Exception Vector



Kuva 23. Generointi

Seuraavaksi siirrytään kuvan 23 mukaisesti Generation-välilehdelle, jossa Synthesis-kohtaan valitaan VHDL alasvetovalikosta ja sen jälkeen tallenna tiedosto. Tiedoston tallentamisen jälkeen painetaan Generate.

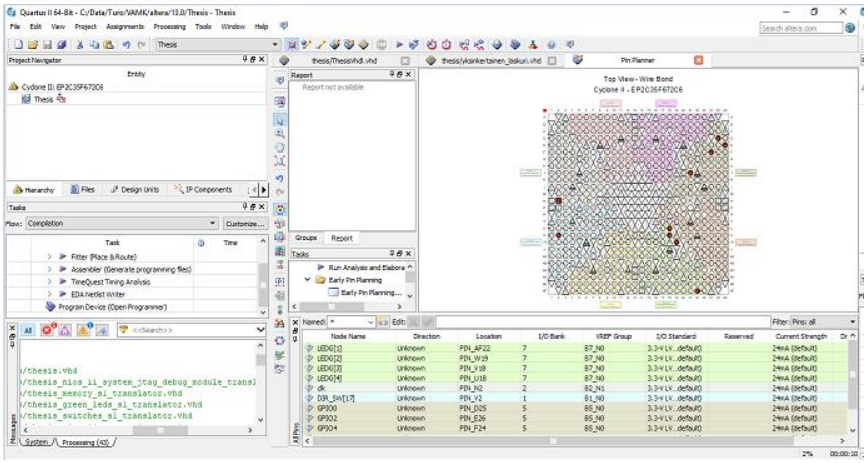


Kuva 24. Schematic-tiedosto

Sen jälkeen, kun Qsys-ohjelman generoitu on valmis, se suljetaan ja palataan takaisin Quartus II-ohjelmaan. Siellä valitaan File-valikosta New, josta valitaan Schematic File kuvan 24 mukaisesti.

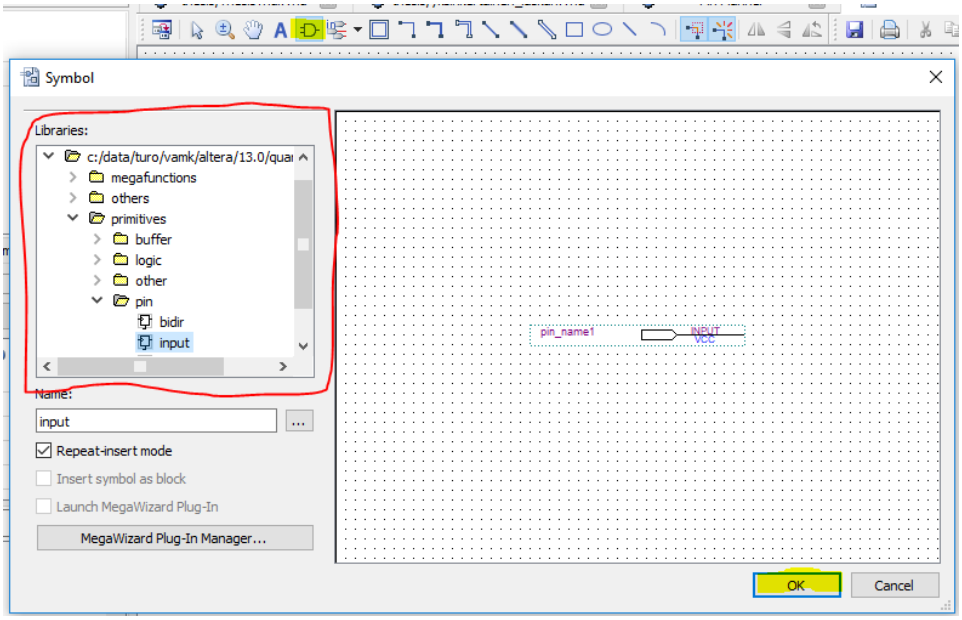
6.2.2 Schematic

Schematic on Quartus II:n sisältämä kytkennän piirto- ja testausohjelmisto. Projektissa sillä suunnitellaan tehtävä kytkentä, jolle tehdään ohjelmointi VHDL Filessä.



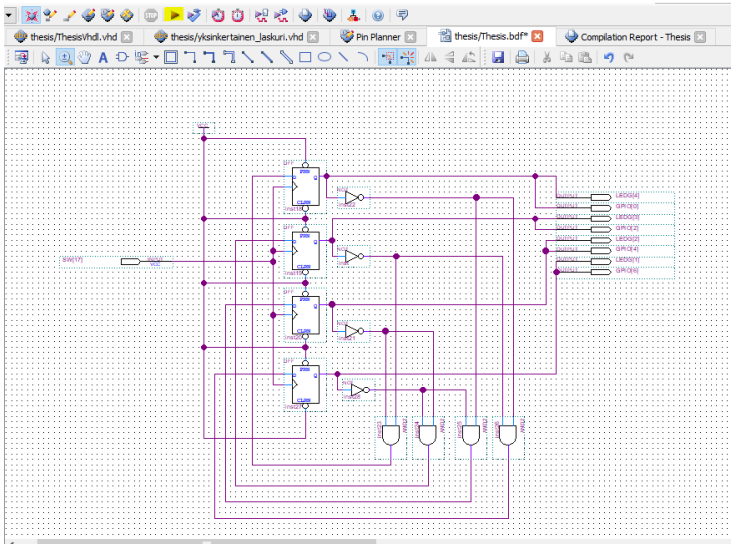
Kuva 25. Pin Planner

Schematic-kuvaa tehdessä on hyvä hyödyntää kuvan 25 Pin Plannerissä näkyviä tietoja, jotka ovat lisätty sinne komponenttien ja FPGA-piirin datalehtien avulla.



Kuva 26. Komponentin lisäys

Valitsemalla symbol-valikko päästään valitsemaan kuvan 26 mukaisesti projektissa tarvittavia symboleja, joilla saadaan piirrettyä projektin kytkentä. Apuja symbolien valintaan saadaan Pin Planneristä, sekä komponenttien datalehdistä. Komponentti valitaan listasta ja sen jälkeen painamalla OK, niitä voidaan lisätä tarpeen mukaan. Lisääminen saadaan lopetettua painamalla ESC-näppäintä. Schematicin tultua valmiiksi, se tallennetaan projektin kanssa samaan paikaan ja se on usein hyvä nimetä samannimiseksi projektin kanssa.

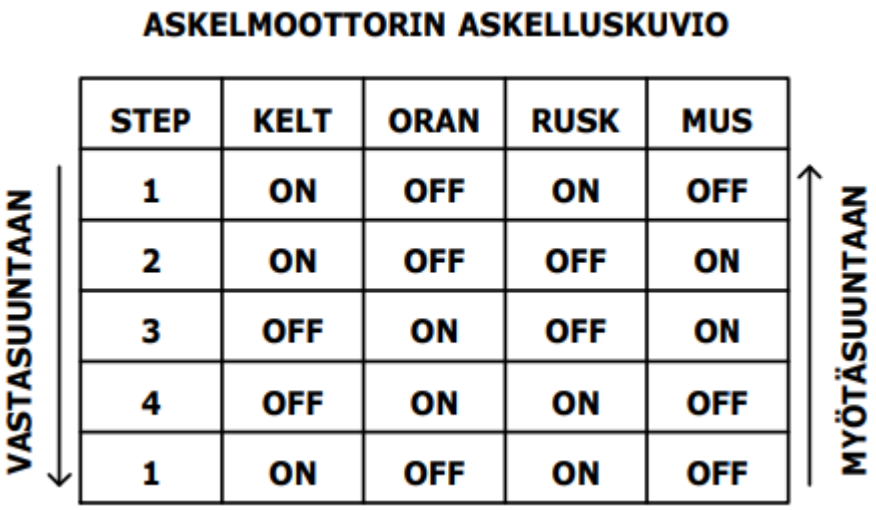


Kuva 27. Schematicin ajo

Schematic-piirustuksen valmiiksi saannin jälkeen se ajetaan ohjelman sisälle kuvan 27 mukaisesti merkattua Start Compilation-painiketta painamalla. Ajaessaan sitä ohjelmalle kytkentä tarkistetaan, että kaikki on kunnossa ja jokainen komponentti on kytketty kunnolla kiinni.

6.2.3 VHDL

VHDL:ssä kirjoitetaan projektin käyttämiä koodeja. Koodin kirjottamisessa käytetään apuna komponenttien datatietoja ja Schematiciä, joka on piirretty aikaisemmin.



Kuva 28. Askelmoottorin askelkuvio /7/

Taulukko 1. Askelmoottorin askelkuvio binäärisenä

STEP	KELT	ORAN	RUSK	MUS
1	1	0	1	0
2	1	0	0	1
3	0	1	0	1
4	0	1	1	0
1	1	0	1	0

Kuvasta 28 ja taulukosta 1 voidaan tarkastella millä tavalla projektin askelmoottori toimii ja tätä tietoa hyödynnetään, kun luodaan taulukon 2 sisältämä koodi.

Taulukko 2. VHDL koodi

```
library IEEE;
 use IEEE.STD_LOGIC_1164.ALL;
 use IEEE.numeric_std.all;
 use IEEE.STD_LOGIC_ARITH.ALL;
 use IEEE.STD_LOGIC_UNSIGNED.ALL;
 entity Thesis is
    Port (
              CLOCK_50 : in  std_logic;
              paikka : out std_logic_vector (3 downto 0);
                        gpio: out std_logic_vector (3 downto 0);
              sw: out std_logic_vector (17 downto 0);
                           ledg: out std_logic_vector (17 downto 0);
              led_katodi     : out std_logic;
              led_anodi       : out std_logic
);
 end Thesis;
 architecture RTL of Thesis is
                           component Thesis is
                              port(
                              ckl_ckl      : in std_logic;
                              reset_reset_n              : in std_logic;
                              leds_external_interface_export : std_logic_vector(17 downto 0);
                              switches_external_interface_export : std_logic_vector(17 downto 0);
                               );
                              end component;
constant max_laskuri : natural := 48000000;
 signal Rst_signal : std_logic;
 begin
```

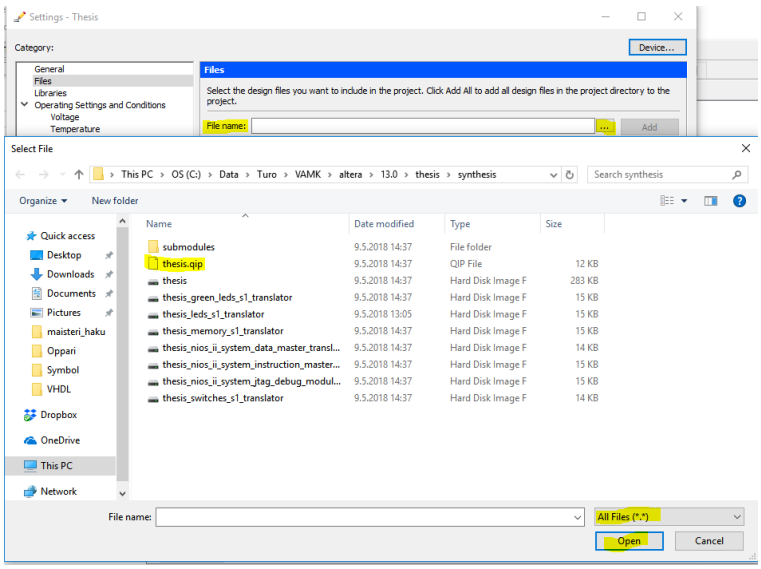
```
    Rst_signal <= '1';
    led_katodi <= '0';
    pulssi : process(CLOCK_50, Rst_signal);
                variable laskuri : natural range 0 to max_laskuri;
                variable sw : std_logic_vector (1 downto 0) := "00";
begin
       if Rst_signal = '0' then
          laskuri := 0;
          led_anodi <= '1';
  else if rising_edge(Clk) then
                          sw:=sw+1;
if laskuri < max_laskuri/2 then
             led_anodi    <='1';
             laskuri := laskuri + 1;
else if laskuri < max_laskuri then
             led_anodi    <='0';
             laskuri := laskuri + 1;
else
             laskuri := 0;
             led_anodi    <='1';
case (sw) is
                when "00" => paikka <= "1000";
                when "01" => paikka <= "0100";
                when "10" => paikka <= "0010";
                when "11" => paikka <= "0001";
 end case;
case (sw) is
                when "00" => gpio <= "1000";
                when "01" => gpio <= "0100";
```

```
                when "10" => gpio <= "0010";

                when "11" => gpio <= "0001";

    end case;

end if;

      end if;

   end process pulssi;

end RTL;
```

Pelkästään yksi VHDL-koodi ei riitä, että projekti saataisiin toimimaan, vaan siihen on hyvä lisätä myös yksinkertainen laskuri, jota voidaan käyttää kaikkien projektin VHDL-koodien taustalla. Siihen on myös hyvä lisätä vilkku ja kellon VHDL-tiedostot. Nämä näyttävät mikä askel on kyseessä, kun ohjelma pyörii.

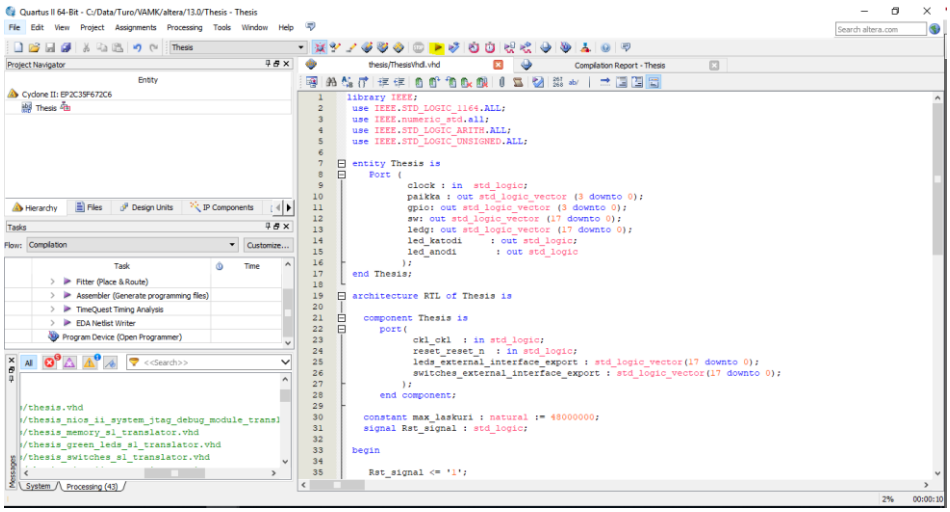
6.2.4 Ohjelman siirto FPGA:lle

VHDL-koodien luonnin jälkeen lisätään projektiin aikaisemmin luotu qip tiedosto, joka löytyy projektin kanssa samasta kansiosta. Tämän jälkeen voidaan ajaa koodi FPGA-laitteelle.



Kuva 29. Qip-tiedosto

Lisätään qip projektiin hakemalla se projektin kansiosta, josta valitaan Add/Remove Files in Project. Valitaan kuvan 29 mukaisesti tiedoston tyypyksi All Files. Näin saadaan theisis.qip-tiedosto näkyviin. Valitaan qip-tiedosto ja painetaan Open, jonka jälkeen painetaan Apply.

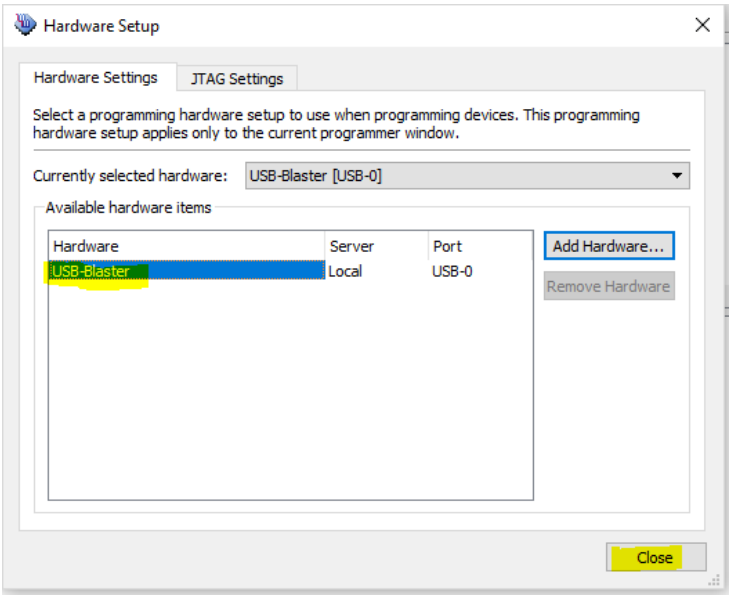


Kuva 30. Ohjelman ajo

Tiedoston lisäämisen jälkeen projekti tallennetaan, jonka jälkeen se voidaan ajaa painamalla kuvan 30 mukaisesti merkitystä Star Compilation-nappia. Tämän jälkeen Quartus II suorittaa projektin ja suorittamisen yhteydessä tarkistetaan kirjoitettuja koodeja. Jos niissä ilmenee ongelmia, tulee niistä ilmoitus, joka kertoo missä ja miksi kyseinen virhe on tapahtunut. Mikäli ongelmia ei ilmene, se suorittaa projektin ajon ilman ilmoituksia.

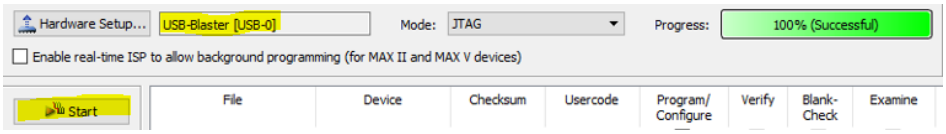
Kuva 31. Työn ajo FPGA:lle

Painamalla kuvan 31 mukaisesti merkittyä Programmer-nappia päästään ajamaan koodia FPGA-piirille. Ensiksi määritellään ohjelmalle piriin käyttämä Hardware.



Kuva 32. Hardware-valinta

Tässä projektissa käytetään kuvan 32 mukaisesti Hardwarena USB-Blasteria. Oikean valinnan jälkeen voidaan ajaa ohjelmisto piirille kuvan 33 mukaisesti merkittyä Start-painiketta painamalla.



Kuva 33. Ohjelman ajo

Ohjelman ajon jälkeen ohjelma on valmis. Sen jälkeen voidaan liittää kytkentäpiuhat ja testata toimintaa käytännössä.

7 YHTEENVETO

Tämä projekti oli erittäin mielenkiintoinen ja mielestäni se oli myös erittäin opettavainen. Se sisälsi paljon koulussa oppimaani tieto ja siinä oli myös paljon uutta tietoa, jota jouduin etsimään ja opettelemaan.

Lopputulos oli onnistunut ja omasta mielestäni myös erittäin mielenkiintoinen, mutta siihen jäi vielä parannettavaa ja kehitettävää.

Se miten sitä voisi kehitellä on se, että siihen voitaisiin lisätä enemmän kytkimiä, joiden myötä voitaisiin lisätä enemmän moottoreita tai joitain muita toimintoja. Siihen voitaisiin lisätä myös muisti, jonka avulla voitaisiin tallentaa ajettu ohjelma laitteelle mahdollisten sähkökatkosten varalta.

Olen kuitenkin tyytyväinen lopputulokseen ja toivon, että tästä työstä olisi hieman apua tulevaisuudessa muille ja itselleni.

LÄHTEET

/1/ Wikipedia FPGA-artikkeli. Wikipedian suomenkieliset verkkosivut. Viitattu 27.1.2018. https://fi.wikipedia.org/wiki/FPGA

/2/ Wikipedian FPGA-artikkeli. Wikipedian englanninkieliset verkkosivut. Viitattu 31.1.2018. https://en.wikipedia.org/wiki/Field-programmable_gate_array

/3/ Wikipedian Altera Quartus-artikkeli. Wikipedia englanninkieliset verkkosivut. Viitattu 10.3.2018. https://en.wikipedia.org/wiki/Altera_Quartus

/4/ Li Jinpengin Opinnäytetyö: Greedy Snake Video Game Based on Nios II System. Viitattu 26.3.2018

/5/ Altera DE2-piirin kuva. Hamlblen verkkosivut. Viitattu 18.4.2018. http://hamblen.ece.gatech.edu/DE2/DE2.jpg

/6/ Wikipedia Nios II-artikkeli. Wikipedian englanninkieliset verkkosivut. Viitattu 27.4.2018. https://en.wikipedia.org/wiki/Nios_II

/7/ Askelmoottorin datalehti. Tietopetrin verkkosivut. Viitattu 6.5.2018. https://www.tietopetri.fi/data/askel.pdf

/8/ UNL2803 datalehti. Electroschematicsin verkkosivut. Viitattu 7.5.2018. https://www.electroschematics.com/wp-content/uploads/2013/07/uln2803a-datasheet.pdf

/9/ Altera DE2 käyttäjä ohje. Alteran verkkosivut. ftp://ftp.altera.com/up/pub/Webdocs/DE2_UserManual.pdf