Algoritmi programmeerimise juhend Basys 3 platvormile

Vivado installeerimine enda arvutisse

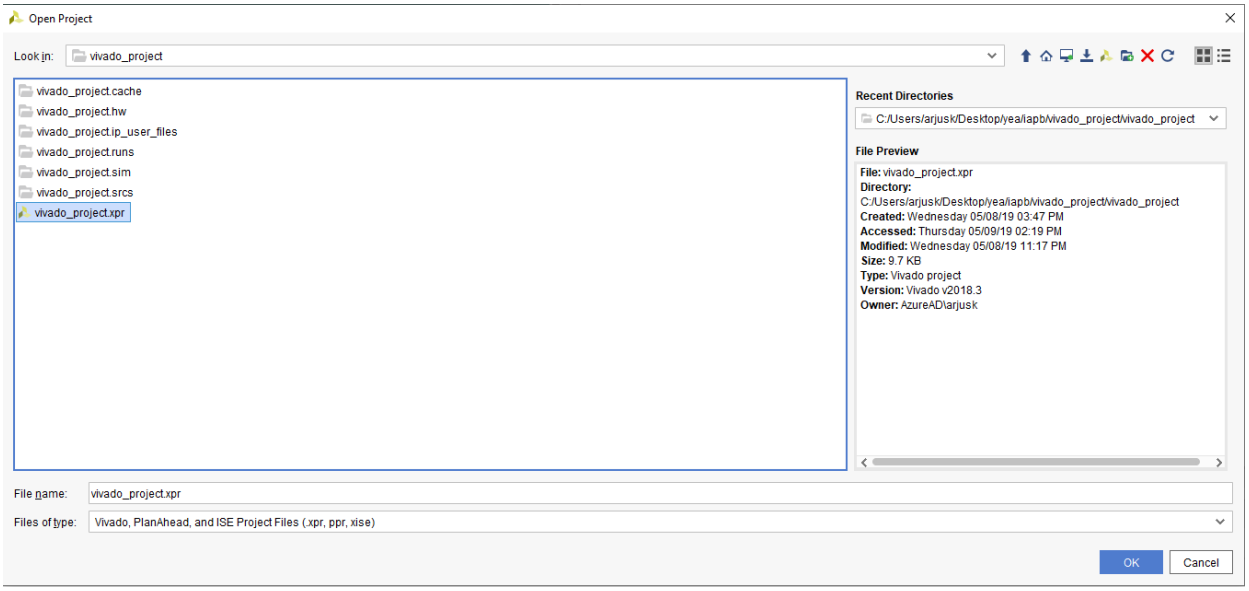
- Vivado laadimiseks minna lehele: https://www.xilinx.com/support/download.html
- Valida **Xilinx Unified Installer** 2019.2: Windows Self Extracting Web Installer (EXE - 65.5 MB)
- Allalaadimise alustamiseks tuleb luua keskkonda konto ning seejärel pärast nime ja aadressi verifitseerimist saab installeerimise faili alla laadida
- Installeerimise valikust valida **Vivado Design Suite** (ning mitte Vitis!). Design Suite valikutest võtta WebPack (ehk tasuta versioon). Kui kasutada ainult Basys3 plaati, siis riistvara tugi peaks olema 7 seeria Artix-7 FPGAle ning võimalusel teised välja lülitada.
- Anda ette sobiv kataloog ning alustada allalaadimisega.

Vivado käiviamine ülikooli arvutis

- Arvuti peab olema Linuxi operatsioonisüsteemis
- Ava terminal sobivas kaustas ning anna järgmised käsud.
    - **cad -> cad -> 4 -> vivado**

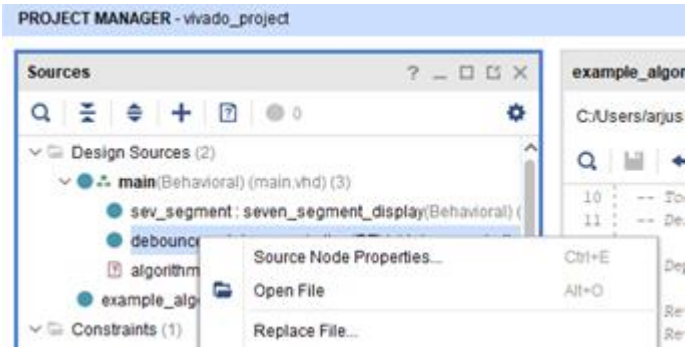
Digitaalsüsteemide #2 Kodutöö projekti avamine

- Kopeeri/klooni või lae alla projekt
    - https://gitlab.cs.ttu.ee/arjusk/iapb.git
- Paki lahti allatõmmatud fail
- Vivados valida „**Open project**“ ning liikuda kausta, kuhu alltõmmatud projekt lahti pakiti
- Leia kaustast fail „**vivado_project.xpr**“, vali fail ning seejärel „**OK**“ (*Joonis 1*)
- Allalaetud pakist on puudu **Basys3_Master.xdc** fail!



Joonis 1. Vivado "Open Project" akna kuva.

Projekti laadimisel võib tekkida olukord, kus faili nimede juurde tekib punane hüümärk. Selle eemaldamiseks tuleb teostada peatüki „Enda algoritmi lisamine projekti“ punktis 2a ja 2b olevad tegevused. Näitena Joonis 2 kuvab valikute menüü avamist, selle vajutada failil paremklõps. Seejärel valida „**Add files**“ ning see peaks automaatselt suunama kausta, kus asuvad kõik „.**vhd**“ failid. Valida tuleb need failid, mille kõrval on hüümärk. Toimub faili linkimine projektis oleva faili prototüübiga.



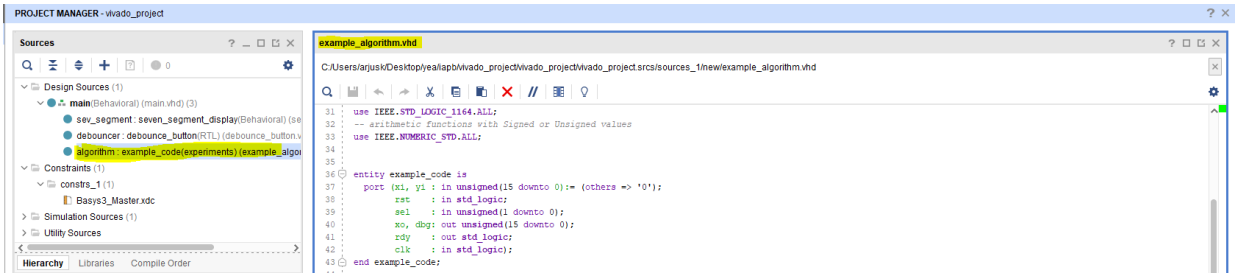
Joonis 2. Projekti faili valikute menüü avamine.

Lisaks täidetavale algoritmile antakse projektiga driver 7-segmendiline indikaatori juhtimiseks ja väreluse vastane programm nuppudelt infot lugemiseks ning piirangute fail, mis seadistab Basys3 füüsiliste osade aadressid ning nimed programmis kasutamiseks. Piirangute failis seadistatud nimed peavad vastama „**main.vhd**“ faili olemis olevatele nimedele (projektis on see kõik juba paigas).

Enda algoritmi lisamine projekti

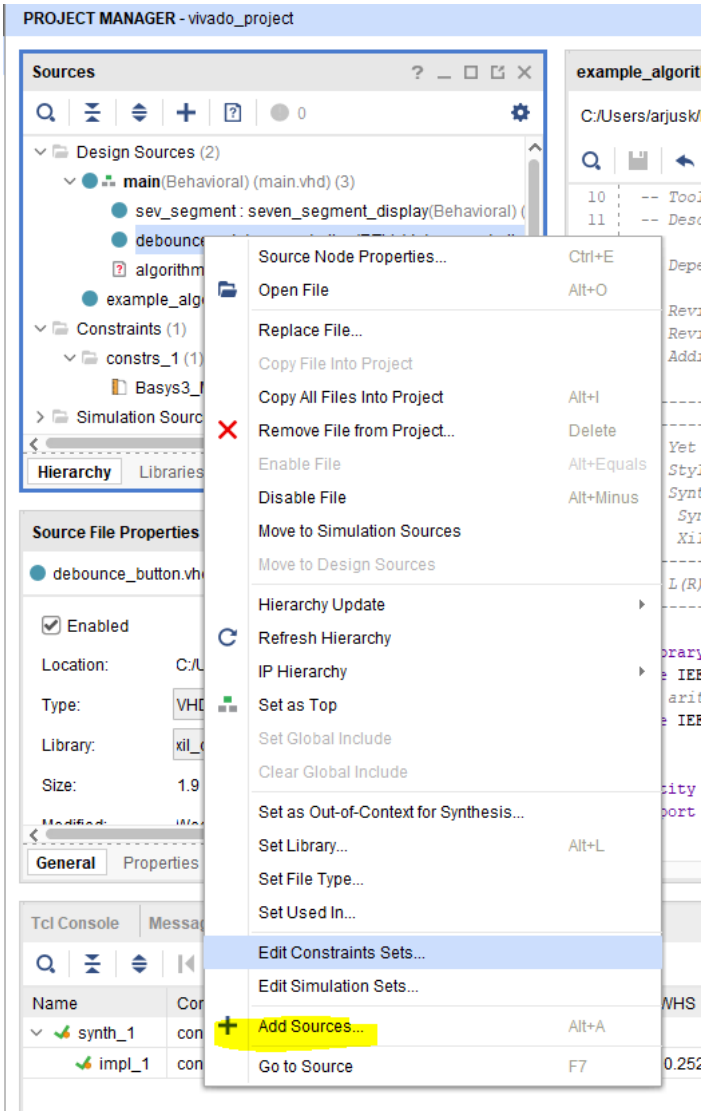
Selleks, et lisada enda algoritm projekti on kaks võimalust:

1. Kõige lihtsam on olemas oleva algoritmi asendamine. Tuleb kopeerida kogu algoritmi osa „**example_algorithm.vhd**“ faili (*Joonis 3*).



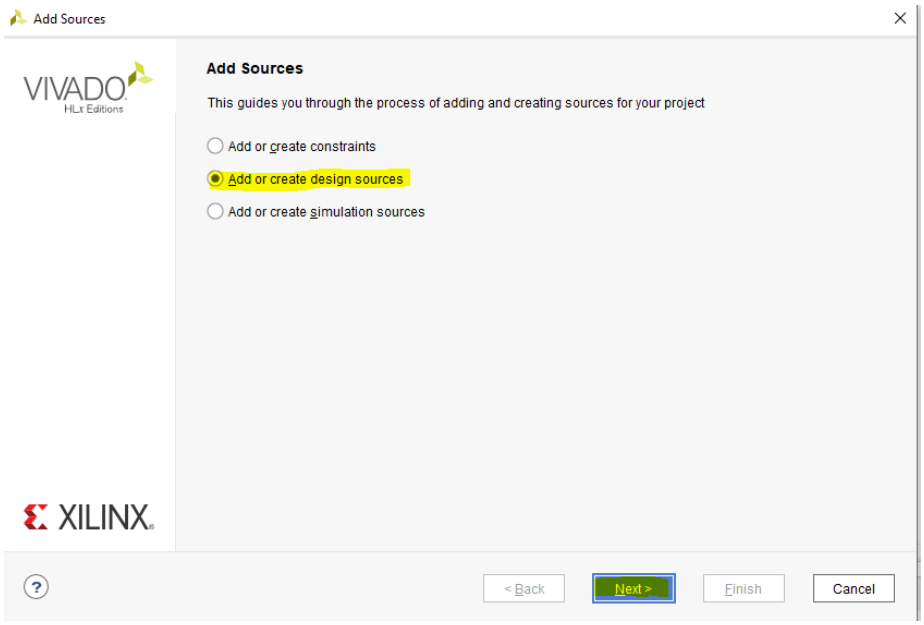
Joonis 3. Enda algoritmi kopeerimine faili "example_algorithm.vhd"

2. Teine variant on lisada uus fail.
    a. Tuleb vajutada Sources aknas parempoolse hiireklahviga ning valida „**Add Source**“ (*Joonis 4*).



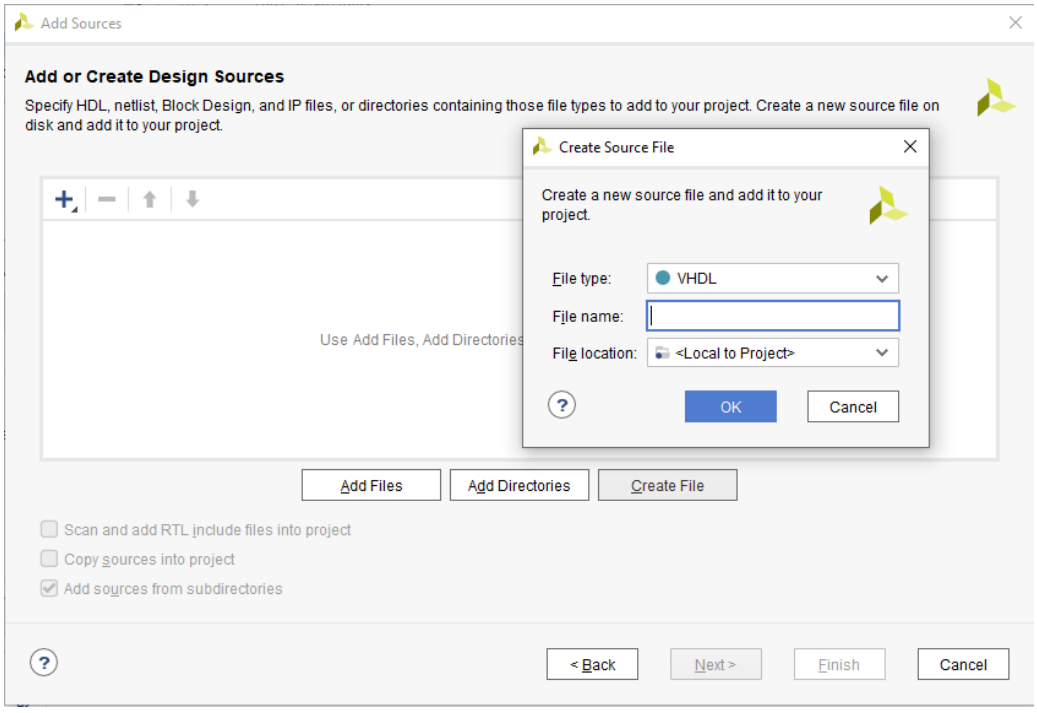
Joonis 4. Faili lisamine projekti.

b. Valida *design sources* ning seejärel vajutada nuppu „**Next**“ (*Joonis 5*).



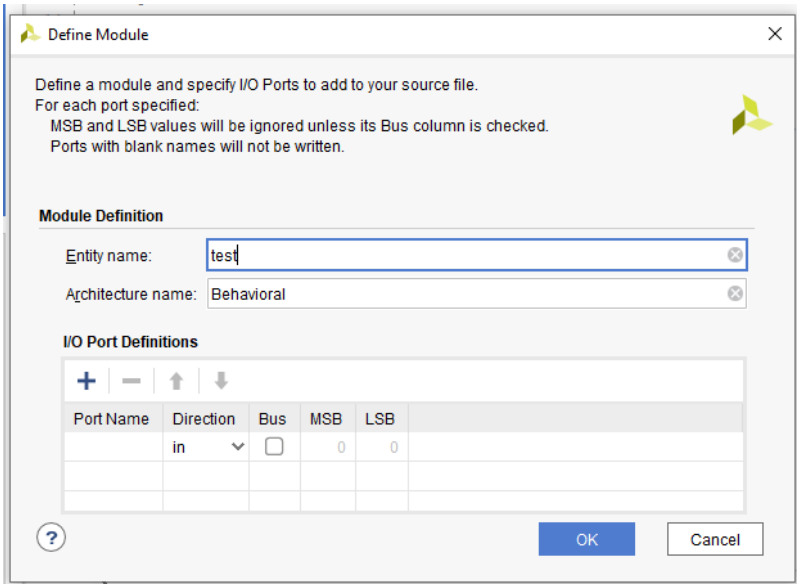
Joonis 5. Algoritmi faili lisamine - "Design source".

c. Avaneb aken (*Joonis 6*), kus saab valida kas luua uus fail või avada olemasolev. Uue lisamiseks vajutada „**Create file**“. Sisestada failinimi ja vajutada „**OK**“. Peale seda muutub aktiivseks nupp „**Finish**“.



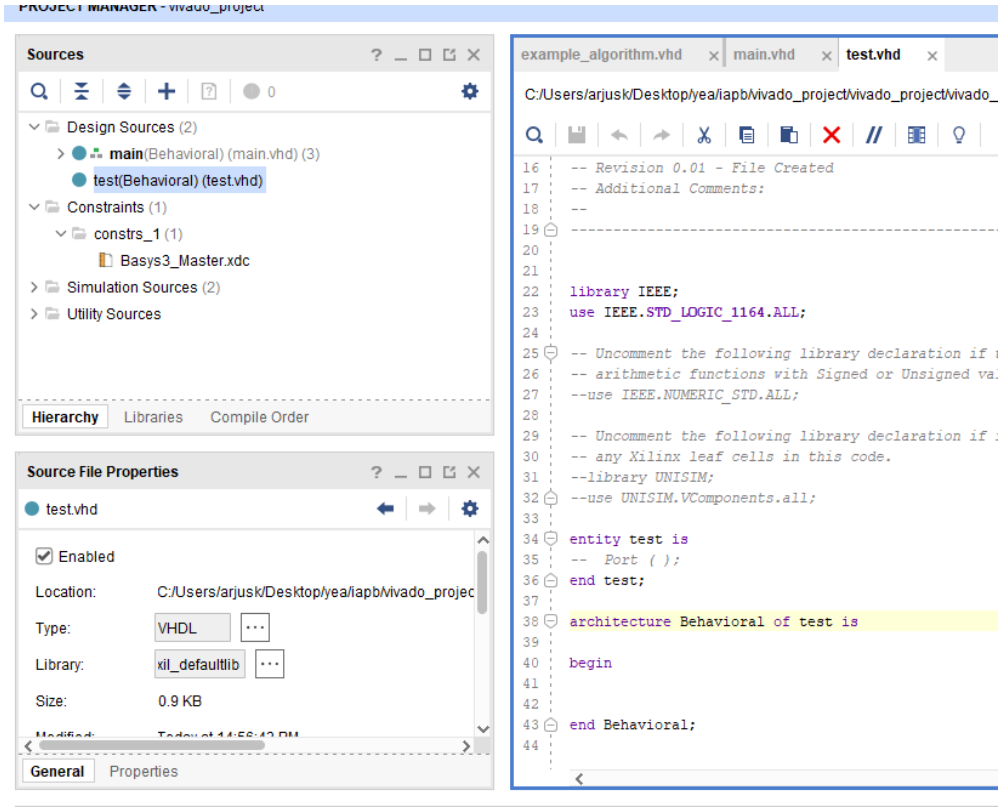
Joonis 6. Faili lisamise kuva.

d. Algoritmi võib ka projekti faili koostada. Selleks peab olema teada, mis nimega on sisendid ja väljundid (kuigi neid saab hiljem ka lisada). *Joonis 7* kujutab sisendite ja väljundite määramise kuva. Kui kõik on valmis vajutada „**OK**“, Signaale saab hiljem ka juurde lisada, muuta või eemaldada (seda siis otse VHDL failis).



Joonis 7. Sisendite ja väljunditele nime andmine uue faili loomisel.

e. Eelnevas punktis sai loodud fail „test.vhd“. Faili võib programmeerida oma algoritmi või kopeerida see varasemalt loodud failist (*Joonis 8*).



Joonis 8. Vivado kuva uuest loodud failist.

Olulised sammud algoritmi koostamisel (lähtudes näidisest „algorithm.vhd“)

Algoritm peab sisaldama protsessi, et saaks algoritmi debugida ehk valida registreid algoritmist, mis kuvatakse 7-segmendi indikaatoril. Muutujad võivad olla teised vastavalt sellele, mida soovitakse ekraanil kuvada. Kuna „**ar**“, „**br**“ ja „**counter**“ on 8-bitised signaalid, siis tuleb lisada 8-bitti nulle muutuja väärtuse ette kuna ekraanile kuvatakse 16-bitiseid signaale (*Joonis 9*).

```
process(sel, ar, br, cr, counter)
begin
        case sel is
                when "00" => dbg <= z8 & counter;
                when "01" => dbg <= z8 & ar;
                when "10" => dbg <= z8 & br;
                when "11" => dbg <= cr;
                when others => dbg <= (others => '0');
        end case;
end process;
```

Joonis 9. Programmikood muutujate viimiseks 8-bitilisest 16-bitiliseks.

- „counter“ loendur, debugimiseks
- „ar“ on esimene number.
- „br“ on teine number.
- „cr“ on vastus.
- „dbg“ on väljund, mida kuvatakse ekraanil
- „sel“ lülitite asendi signaalid

Samuti tuleks jälgida, et „**in**“ ja „**out**“ signaalid oleksid õiget tüüpi. Nimetused võivad olla erinevad. Kasutusel on peamiselt std_logic, unsigned ja std_logic_vector. Neid võib muuta, kuid sel juhul tuleb muuta tüübid ka teistes, seotud failides (*Joonis 10*).

```
entity example_algorithm is
  port (xi, yi : in unsigned(15 downto 0):= (others => '0');
        rst    : in std_logic;
        sel    : in unsigned(1 downto 0);
        xo, dbg: out unsigned(15 downto 0);
        rdy    : out std_logic;
        clk    : in std_logic);
end example_algorithm;
```

Joonis 10. Näidisalgoritmi olemi programmikood.

Selleks, et lisada loodud algoritm „**main.vhd**“ faili, tuleb lisada faili ka vastav komponent.

```
  component example_algorithm is
    port (xi, yi : in unsigned(15 downto 0);
          rst    : in std_logic;
          sel    : in unsigned(1 downto 0);
          xo, dbg: out unsigned(15 downto 0);
          rdy    : out std_logic;
          clk    : in std_logic);
  end component;
```

Joonis 11. Komponendi "example_algorithm" komponendi lisamine.

Seejärel tuleb luua signaalide „**port map**“ ehk vastendada komponendi signaalid algoritmis kasutusel olevate signaalidega (*Joonis 12*).

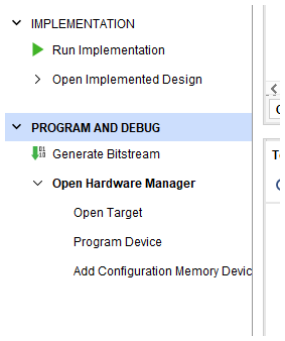
```
algorithm : example_algorithm
port map (
    xi => unsigned(first_number),
    yi => unsigned(second_number),
    rst => btnd,
    sel => sel,
    std_logic_vector(xo) => answer,
    std_logic_vector(dbg) => choice,
    rdy => rdy,
    clk => cclk
);
```

Joonis 12. Signaalide üks-ühele vastendamine.

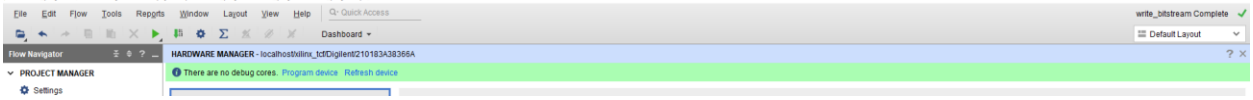
Algoritmi ettevalmistamine Basys 3 plaadile laadimiseks (bitstream)

Algoritmi laadimiseks Basys 3 plaadile on vaja genereerida „**bitstream**“ (Joonis 13). Tegu on Xilinxi poolt loodud kinnise failiformaadiga, mis sisaldab endas FPGA programmeerimiseks vajalikku infot. Basys 3 programmeeriimseks ühendada plaat USB kaabli abil arvutiga ning seejärel lülitada plaadil sisse USB pistiku kõrval olev lüliti. Kui kõik on õigesti hakatakse 7-segmendilisel indikaatoril kuvama numbreid. Juhul kui plaat on ühendatud, kuid 7-segmendiline indikaator sisse ei lülitu kontrollida lüliti kõrval olevat silluse (*jumper*) asendit.

Bitstreami genereerimiseks vajutada vasakul menüüs „**Program and debug**“ ning „**Generate bitstream**“ (Joonis 13). Genereerimise õnnestumist kujutab Joonis 14.



Joonis 13 „Generate Bitstream“ asub vasakul menüüs.



Joonis 14. Edukas bitstreami loomine.

Algoritmi laadimine Basys 3 plaadile

Plaadile laadimiseks vajutada „**Program and Debug**“ menüüs käsule „**Open Hardware Manager**“. Seejärel vajutada menüüs avanenud valikule „**Open Target**“ ning „**Auto Connect**“ (*Joonis 15*). Esimesel laadimisel võib antud protsess aega võtta. Järgnevalt vajutada „**Program Device**“ ja valida pakutud (eeldatavasti ainus) valik. Valiku nimi tähendab ühendatud FPGA kiibi mudelit (**xc7a35t_0**). Kui eelnevalt bitstreami genereerimisel kausta asukohta muudetud pole vajutada nuppu „**Program**“. Kui „**bitstreami**“ asukoht on muutunud, tuleb see avanenud aknas „**Program Device**“ ette anda käsitsi otsides. Viimaseks anda käsk „**Program**“.

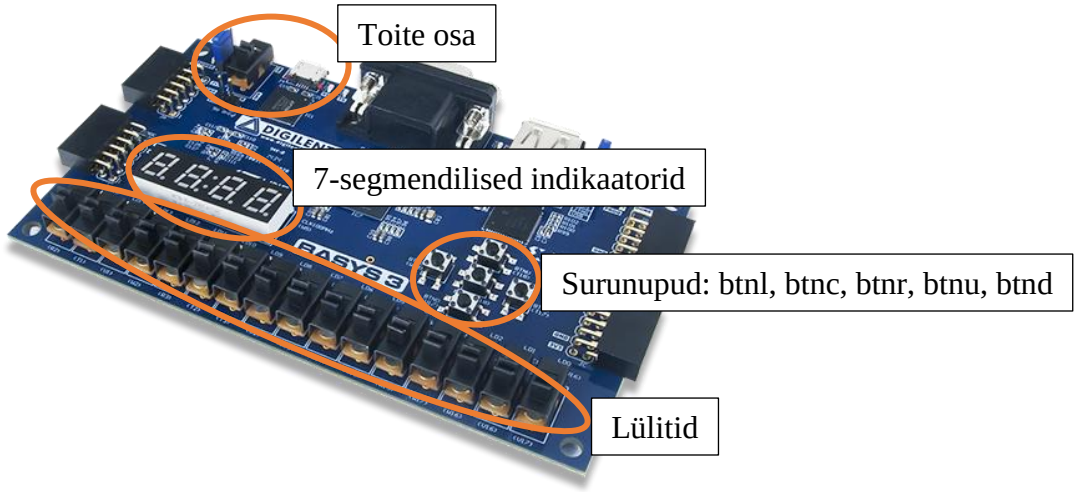
Joonis 15. Plaadi automaatne ühendamine.

Joonis 16. Edukalt ühendatud plaat.

Algoritmi kasutamine plaadil

Operandi sisestamiseks tuleb lülititele panna 2-nd süsteemi arv, mida soovitakse sisestada. Alumine asend tähendab „0“ ning ülemine „1“. Esimese operandi sisestamiseks panna lülitid soovitud asendisse ning vajutada vasakut surunuppu „**btnl**“. Pärast väärtuse sisestamist kuvatakse see 16-nd süsteemis 7-segmendilisel indikaatoril. Teise operandi sisestamiseks panna lülitid soovitud asendisse ning vajutada keskmist surunuppu „**btnc**“. Pärast väärtuse sisestamist kuvatakse see 16-nd süsteemis 7-segmendilisel indikaatoril. Algoritmi käivitamiseks ning tulemuse kuvamiseks tuleb vajutada ülemist nuppu „**btnu**“. Algoritmi sammhaaval käivitamine (debug mode) võimaldab kuvada algoritmi vahetulemusi. Režiimi alustamiseks tuleb hoida all alumist nuppu „**btnd**“ ja vajutada „**btnr**“. Seejärel mõlemad nupud vabastada ning iga parema

nupu „**btnr**“ vajutusega simuleeritakse ühe takti impulsi jooksul läbiviidavat tööd ehk teostatakse järgmine arvutuse samm. Joonis 17.



Joonis 17. Basys 3 FPGA arendusplaat.

Selleks, et kuvada erinevaid registrite väärtusi saab kasutada vasakult kolme esimest lülitit. Lülitite asendi ja kuvatavate registrite kodeeringud on Tabelis 1. Kuvatavete registrite nimed on võetud näidisalgoritmist, mis asub failis „algorithm.vhd“.

Tabel 1 Kuvatavate registrite valik vastavalt kolme lülitile '0' - lüliti väljas, '1' - lüliti sees

Vasakult esimene	Vasakult teine	Vasakult kolmas	Kuvatav register
0	0	0	Esimene number
0	0	1	Teine number
0	1	0	Esimene tulemus
0	1	1	Teine tulemus
1	0	0	Loenduri väärtus algoritmis
1	0	1	Esimene number algoritmis
1	1	0	Teine number algoritmis
1	1	1	Tulemus algoritmis

LED’e kasutatakse selleks, et jälgida, mis olekus on algoritmi juhtautomaat ning mitmete teiste registrite väärtuseid. Tabelis 2 on välja toodud, mis LED, mida näitab.

LED	Kuvatav olek
led(0)	manual registri väärtus
led(1)	algoritmi ready väärtus
led(2)	algoritmi_reset väärtus
led(3)	cclk väärtus
led(10)	state on s_ready olekus
led(11)	state on s_run olekus
led(12)	state on s_reset2 olekus
led(13)	state on s_reset1 olekus
led(14)	state on s_manual olekus
led(15)	state on s_start olekus

Katsetamiseks võib genereerida näidis programmis alguses ühest algoritmist bitstream ning seejärel teisest. Olenevalt algorimtmist kasutuse soovist tuleb välja/sisse kommenteerida vastav algoritmi vastendamise programmikood.

NB! Kuna algoritmis GCD on mitu vaheolekut, siis tuleb umbes kolm korda vajutada paremat nupput takti sisse andmisel, et toimuks arvutus, kuna liigutakse ühest olekust teise. Korrutusalgoritmis liigutakse kohe arvutus olekusse, seepärast toimub seal iga vajutusega vahetulemuste muutus.