Sisukord

1. Projekti üles seadmine 2
2. Oma algoritmi lisamine projekti 3
3. Digilent Basys 3 arendusplaat 5
4. *Debug* loogika 7
5. FPGA programmeerimine 8

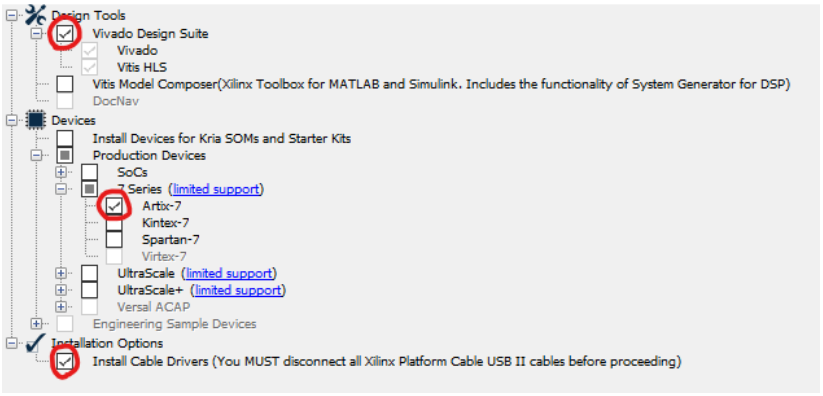
IAS0150 Digitaalsüsteemid –

Arvutusalgoritmi FPGA peal realiseerimine

1. Projekti üles seadmine

Vivado installimine oma arvutis:

- Vivado saab alla laadida: https://www.xilinx.com/support/download.html
- Alla laadida Xilinx Unified Installer 202x.x: Windows Self Extracting Web Installer.(Projekt on testitud versioonidega 2021.2 ja 2022.1)
- Alla laadimiseks tuleb keskkonnas luua tasuta konto
- Installeerimise valikutest tuleb valida vastavalt:
    - Vivado (mitte Vitis)
    - Vivado ML Standard
    - Basys-3 plaati kasutades peavad minimaalselt valitud olema need valikud:



Vivado avamine kooli arvutites

- Arvuti peab olema Linuxi operatsioonisüsteemis
- Ava terminalsobivas kaustas ning anna järgmised käsu
    - cad -> cad -> 4 -> vivado

Vivado projekt

- Projektifail koos kõige vajalikuga on saadaval Github-i repositooriumis: https://github.com/Danel3/IAS0150-FPGA-wrapper
- Peale repositooriumi kloonimist/alla laadimist avada projekti fail: „IAS0150_FPGA_project.xpr“, misjärel peaks projekt avanema Vivado Design Suite-is

2. Oma algoritmi lisamine projekti

Oma algoritmi projekti lisamiseks tuleb kõigepealt valida Vivado aknast: File->Add sources->Add or create design sources->Add Files.

Seejärel tuleb luua Main.vhd faili algoritmi komponent, kus on kõik algoritmi sisendid ja väljundid defineeritud

```
69    --Example gcd algorithm component
70    component gcd
71      port (
72        xi, yi : in unsigned(15 downto 0);
73        clk    : in std_logic;
74        reset  : in std_logic;
75        sel    : in std_logic_vector(1 downto 0);
76        xo,dbg : out unsigned(15 downto 0);
77        rdy    : out std_logic;
78        led    : out std_logic_vector(15 downto 0)
79      );
80    end component;
81
82    -- Your algorith
83  --component [entity name]
84  --port (
85
86
87  --);
88  --end component;
```

Sisendid ja väljundid peab siduma seejärel juhtloogika signaalidega. Juhtloogikas on kõik signaalid *std_logic* tüüpi. Oma algoritmis võib kasutada teisi tüüpe, nt *signed/unsigned*, aga sellisel juhul tuleb signaalide sidumisel tüüpi muuta. Vastavalt vajadusele tuleb muuta ka massiivide pikkusi. **Peale oma algoritmi lisamist kustuta/kommenteeri välja näidisalgoritm.**

```
Multiplication : booth
port map(
  xi => signed(reg_x(9 downto 0)),
  yi => signed(reg_y(9 downto 0)),
  rst => reset,
  clk => clk_algo,
  std_logic_vector(xo) => result(9 downto 0),
  rdy => ready,
  led => led,
  sel => switch(15 downto 14),
  std_logic_vector(dbg) => debug
);
```

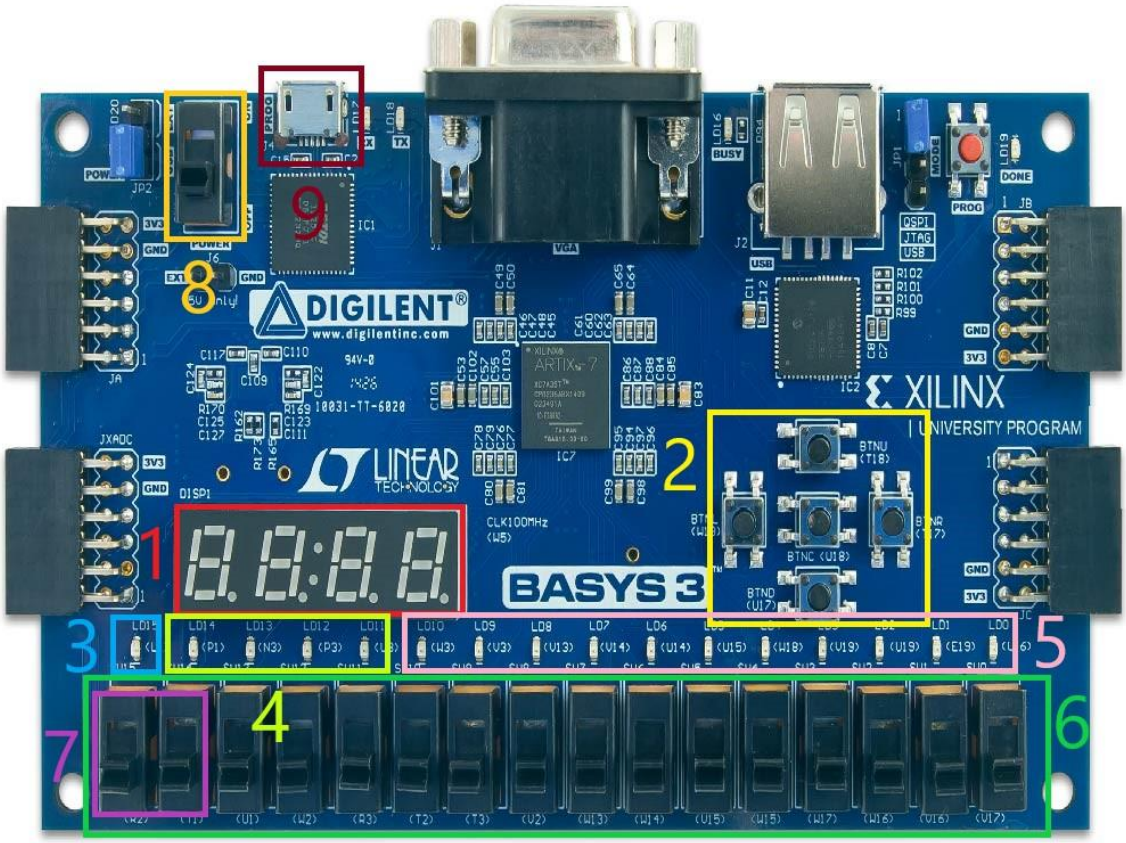
Kindlasti peavad olema kasutusel järgnevad signaalid:

- x ja y – kuni 16 bitised registrid sisendarvude jaoks
- result – kuni 16 bitine register algoritmi vastuse jaoks
- clk_algo – taktsignaal
- reset – reset signaal
    - 1 – sisendid loetakse algoritmi registritesse ja algoritm alustab tööd
    - 0 – Algoritm on töö lõpetanud ja on ootel
- ready – algoritmi töö lõpu signaal
    - 1 – algoritm on töö lõpetanud ja vastus on leitud
    - 0 – algoritm ei ole tööd alustanud, või on pooleli

Abisignaalid (vt. peatükki 4):

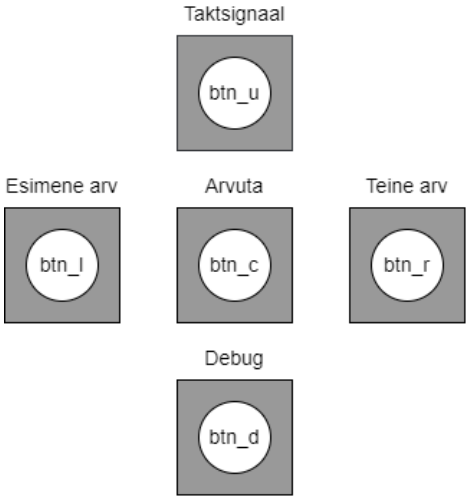
- led – vajalik, et kuvada LED indikaatoritega arvutusalgoritmi olekuid
- sel – 2 bitine massiiv debug lülitite asendi salvestamiseks
- debug – Võimaldab kuvada vaheregistrite väärtusi

3. Digilent Basys 3 arendusplaat



1. 7 segmendi ekraan arvude kuvamiseks. Arve kuvatakse kuueteistkümnendsüsteemis.
2. Nupud arvude sisestamiseks ja algoritmi käivitamiseks.
3. *Debug* režiimi oleku indikaator. Kui LED põleb, siis *Debug* režiim on sisse lülitatud (vt. peatükk 4.)
4. Juhtautomaadi olekute indikaatorid.
5. Arvutusalgoritmi olekute indikaator. Maksimaalselt saab kuvada 11 olekut.
6. 16 lülitit arvude sisestamiseks. Arvud sisestatakse kahendkoodis, kus parempoolseim(SW0) lüliti on madalaim ja vasakpoolseim(SW15) kõrgeim bit. Lüliti alumisele asendile vastab 0 ja ülemisele 1
7. *Debug* lülitid registrite kuvamiseks. Kui *debug* režiim on sisse lülitatud, siis saab kahe vasakpoolseima lülitiga(SW15-14) kuvada arvutusalgoritmi vaheregistrite väärtusi.
8. Plaadi toitelüliti
9. Micro-USB pistik plaadi toite ja programmeerimise jaoks.

Indikaator LEDide asukoht ja nende funktsionaalsus.



- btn_l – vasak nupp, mille vajutusega salvestatakse lülitite hetkeasendi järgi esimese operandi väärtus.
- btn_r - vasak nupp, mille vajutusega salvestatakse lülitite hetkeasendi järgi teise operandi väärtus.
- btn_c – keskmine nupp, millega käivitatakse arvutusalgoritm. Nupu vajutusega edastatakse registrite väärtused arvutusalgoritmile.
- btn_u – ülemine nupp, millega saab arvutusalgoritmi takthaaval lahendada. Esimese vajutusega edastatakse registrite väärtused algoritmile. Iga järgmine vajutus on üks takt, millega algoritmi lahendatakse.
- btn_d – alumine nupp, millega saab debug režiimi sisse/välja lülitada. Debug režiimis saab 7-segmendi ekraanil kuvada registrite vahetulemusi.

4. *Debug* loogika

Basys-3 arendusplaadi peal on võimalik indikaator LEDide abil näidata algoritmi hetkeolekut. Selle jaoks peab igale olekule lisama ühe rea koodi, kus vastav väärtus led massiivis seatakse kõrgeks.

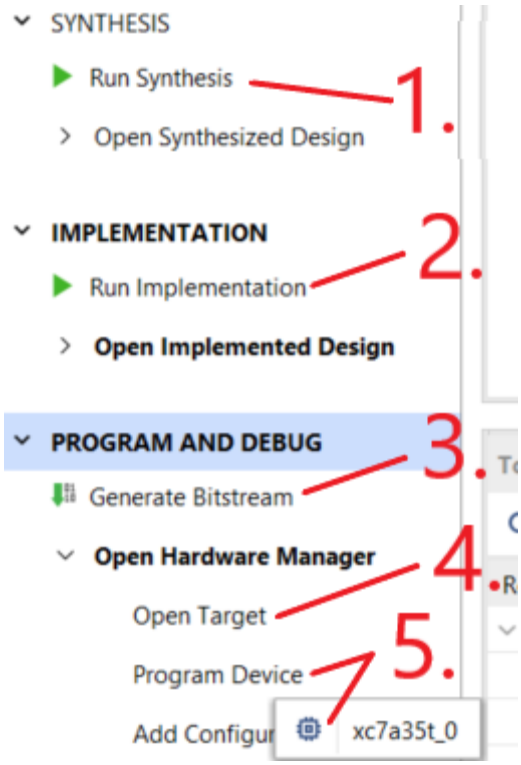
```
when S_wait =>
  led(5 downto 0) <= "000001";
  if  reset='1'  then
    xi_yi_sel <= '1';    ena_x <= '1';    ena_y <= '1';
    next_state <= S_start;
  end if;
-- Loop: ready?
when S_start =>
  led(5 downto 0) <= "000010";
  if  alu_ne='1'  then  next_state <= S_comp;
```

Kahe vasakpoolse lüliti(sw15 ja sw14) abil on võimalik ekraanil kuvada algoritmi eri registrite tulemusi. Selle jaoks tuleb algoritmi lisada üks protsess. Antud näites on „sel“ massiiv lülitite 15 ja 14 positsioon. Vastavalt lüliti positsioonidele antakse dbg massiivile mingi registri hetkeväärtus. *Debug* reziimis kuvatakse „dbg“ väärtus 7-segmendi ekraanil.

```
process(sel, x, y, xo_bf)
begin
  case sel is
      when "00" => dbg <= xo_bf;

      when "01" => dbg <= x;

      when "10" => dbg <= y;

      when others => dbg <= (others => '0');

  end case;
end process;
```

5. FPGA programmeerimine

Vivado vasakus servas olevas menüüs on mitu valiku, millega saab VHDL koodist bitijada genereerida ja selle plaadile laadida.



Esimese kolme punkti jaoks ei pea Basys 3 arendusplaat olema ühendatud. Esimesed 3 punkti võib teha ka korraga vajutades „Generate Bitstream“ nupule. See võib võtta aega rohkem kui minut. Hetkeolekut näeb akna üleval paremas nurgas

Viimase kahe punkti jaoks peab Basys-3 arendusplaat olema ühendatud USB kaabliga arvuti külge ja toitelülitist sisse lülitatud. Esmalt peab valima „Open Target“ ja seejärel valima rippmenüüst „Auto connect“.

Kui ühendus on olemas, siis läheb „Program Device“ nupp aktiivseks ja seda vajutades saab plaati programmeerida. Peale seda on algoritm kasutusvalmis.