AMETIALASEKS KASUTAMISEKS

Küberneetika AS

KRÜPTOKIIP PLD001

TEHNOSPETSIFIKAAT

DO-TD-X-15-0698 Red. 1

70 lk.

Koostas:
Jüri Põldre
Kooskõlastas:
Monika Oit

Tallinn 1998

EESSÕNA

Käesolev tehnospetsifikaat kehtestab digitaalses andmesides võtmevahetuseks ning krüpteerimiseks kasutatavale mikroskeemile PLD001 (edaspidi kiip) esitatavad nõuded ja katsemeetodid nende nõuete kontrolliks. Tehnospetsifikaat on koostatud Tallinna Tehnikaülikooli Arvutitehnika Instituudis ja kooskõlastatud Küberneetika AS-i poolt.

SISUKORD

1. KASUTUSALA DIGITAALSES ANDMESIDES 5

2. NORMATIIVVIITED 7

3. TERMINID, MÄÄRATLUSED, LÜHENDID 8

4. NÕUDED 9

4.1 Kiibi viigud 9

4.2 Füüsilised tingimused 11

4.3 Keskkonna tingimused 12

4.4 Andmevahetus portide kaudu 13

4.5 Käskude täitmine ja käsusüsteem 18

4.6 Kiibi funktsioonid 23

5. KATSEPLAAN 29

5.1 Testitavad funktsioonid 29

5.2 Integreeritud testi alametapid 30

6. KATSETINGIMUSED 31

6.1 Normaaltingimused 31

6.2 Maksimaalsed piirtingimused 31

6.3 Minimaalsed piirtingimused 31

7. KATSEVAHENDID 32

7.1 Liides kiibi sisendite juhtimiseks ja jälgimiseks 32

7.2 Programm liidese juhtimiseks 33

7.3 Kiibi testprogramm 35

7.4 Testimise aparatuur 36

8. KATSEMEETODID JA HINDAMISE KRITEERIUMID 38

8.1 Katseskeem 38

8.2 Liidese juhtprogrammi sisendid 38

8.3 Liidese juhtprogrammi käivitamine ning väljundid 39

8.4 Liidese juhtprogrammi teated 39

8.5 Tulemuste registreerimise viis ja hindamise kriteeriumid 40

LISAD

LISA A Kiibi testprogramm 42
LISA B Siluri isa välismooduli (SVM) juhtprogramm 47
LISA C Testprogrammi ekraaniväljund 50
LISA D Konstant 0 testi tulemused: RAM_C0.DAT 51
LISA E Konstant 1 testi tulemused: RAM_C1.DAT 52
LISA F Malelaua testi tulemused: RAM_CHKB.DAT 53
LISA G Indekseeritud sisendjada testi tulemused: RAM_INDX.DAT 54
LISA H Mälu seis enne modulaararitmeetika teste: WAIT.000 55
LISA J Liitmise ja korrutamise testi tulemused: WAIT.001 56
LISA K Mooduliga korrutamise testi tulemused: WAIT.002 57
LISA L Diskreetne eksponent, testi tulemused: WAIT.003 58
LISA M SVM moodul 59

JOONISTE LOETELU

Joonis 1. Krüptokiibi ühendamine andmesideks 5
Joonis 2. CLCC68 korpus altvaates 9
Joonis 3. Väljundviivitus 12
Joonis 4. Sisendviivitused 12
Joonis 5. Mälu sünkrosignaal 16
Joonis 6. Kiibi arhitektuur 23
Joonis 7. Kiibi töödiagrammid 24
Joonis 8. Kiibi lähtestamine 24
Joonis 9. Kiibi tööreziimide vahetamine 25
Joonis 10. Väliskäsu käivitamine 25
Joonis 11. Registermälu sisend-väljund 26
Joonis 12. Silumisreziimist väljumine 28
Joonis 13. SVM liides 32
Joonis 14. Kiibi katseskeem 38

TABELITE LOETELU

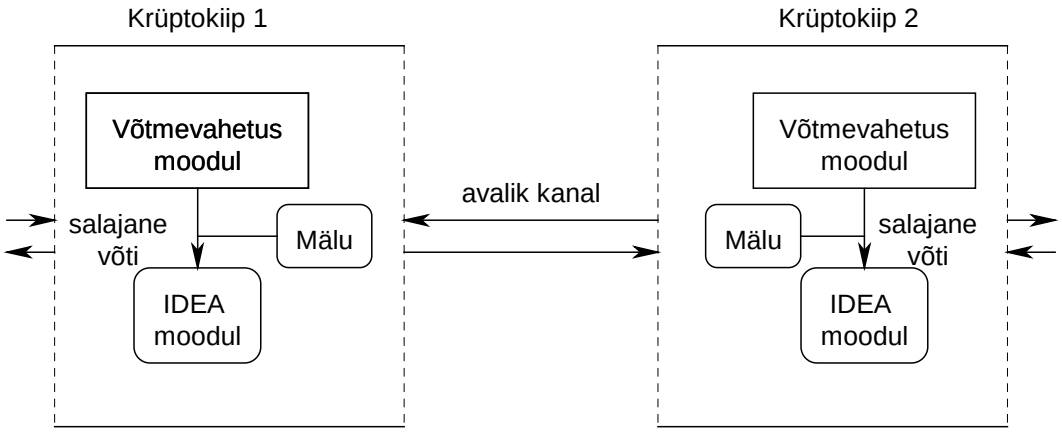
Tabel 1. Kiibi viikude loetelu 9
Tabel 2. Kiibi käsustik 19
Tabel 3. Kiibi registrid 20
Tabel 4. Käskude bitiväljad 20
Tabel 5. Käskude kodeeringud 21
Tabel 6. Testitavad funktsioonid 29
Tabel 7. Integreeritud testi alametapid 30

1. KASUTUSALA DIGITAALSES ANDMESIDES

Kiip on seade, mis on mõeldud digitaalsete andmete krüpteerimiseks ning võtmevahetuseks kahe punkti vahelises sides (*point-point link*) läbi avalike kanalite. Selle saavutamiseks koosneb kiip kahest eraldi sõlmest:

- andmete krüpteerimise moodul
- võtmevahetuse moodul

Andmete krüpteerimis mooduliks on IDEA [APPC, 321-325]. Võtmevahetusmoodul garanteerib IDEA salajase võtme vahetuse läbi avalike kanalite kasutades avaliku võtme krüptograafia algoritmi RSA [APPC, 466-474].



Joonis 1. Krüptokiibi ühendamine andmesideks

Kiibi töö kõige kõrgemal tasemel koosneb neljast operatsioonist:

1) lähtestada kiip (*reset, initialize*),
2) genereeride võti,
3) vahetada genereeritud võti,
4) vahetada andmeid läbi IDEA kanali.

Esimene ja neljas on üheselt määratud. Teine ning kolmas on programeeritavad vastavalt kasutatavale võtmevahetusalgoritmile.
IDEA ning võtmevahetusmoodul kasutavad sama ALU, et hoida kokku kiibi pindala ning seoses sellega vähendada võimsustarvet ning hinda. Antud realisatsioonil ei ole pearõhk mitte võimalikult kiirel võtmavahetusel, vaid kompromissil IDEA ning võtmevahetusmooduli vahel.
Mõlemad sõlmed on omavahel seotud läbi mälu. Mälus hoitakse võtmevahetuseks vajalikke konstante ning IDEA salajasi võtmeid.
Kuna kogu mälu sisu võib antud kiibi realisatsioonis lugeda ning kirjutada väljastpoolt on IDEA algoritmi võimalik testida eraldi võtmevahetusest, kirjutades otse mällu laiendatud IDEA võtme.

Võtmevahetusmoodul (MODEX) realiseerib IDEA võtme vahetust. Assembleri tasemel on realiseeritud käsustik, mis võimaldab võtmevahetusalgoritme võimalikult lihtsalt teostada. Assembleris on tavalistele käskudele (siirded, andmeteisaldus) lisaks spetsiifilised pikkade täisarvudega arvutamiskäsud. Nende käskude abil saab minimaalse programmi pikkusega realiseerida täisarvude modulaararitmeetikale baseeruvaid avaliku võtmega krüptosüsteeme.

Lisaks nendele moodulitele on kiibil veel SV moodul, mis hoolitseb kiibi registermälu ning IDEA teisenduse andmevahetuse eest. IDEA korral töötab SV moodul paralleelselt teisendusega, et saavutada soovitud krüpteerimiskiirust.

2. NORMATIIVVIITED

[APPC] Schneier, Bruce. Applied Cryptography Second Edition: Protocols, algorithms, and source code in C. 1996, John Wiley & Sons, Inc.

[ES210] Europractice, ATMEL 1.0 µm ECPD10 CMOS technology library databook.

[SILUR] Ahto Buldas, Jüri Põldre. Krüptokiibi silur. Programmi dokumentatsioon. DO-TD-X-09-1094.

[SVMDOK] Ahto Buldas, Jüri Põldre. Krüptokiibi emulaator. Projekti dokumentatsioon. DO-TD-X-13-1097.

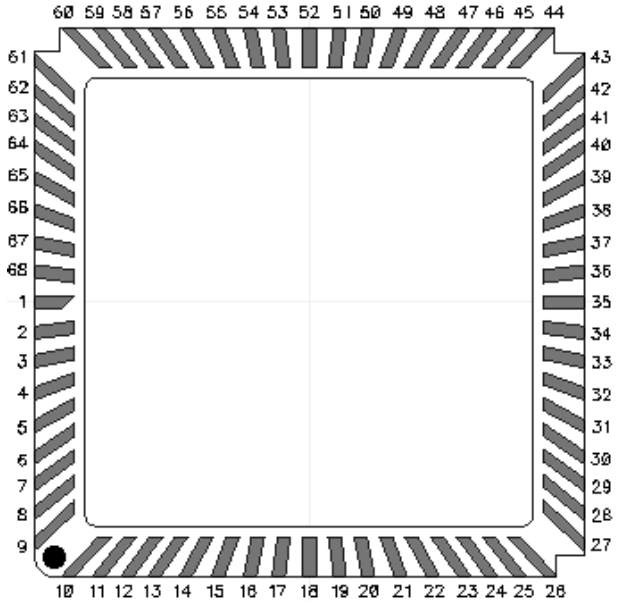
3. TERMINID, MÄÄRATLUSED, LÜHENDID

ALU	Aritmeetika-loogikasõlm
IDEA	64-bitise lähte- ning krüptotekstiga ja 128-bitise võtmega krüptoalgoritm International decryption encryption algoritm
RSA	Avaliku võtme algoritm, kasutatakse siin võtmevahetuseks
Prototüüp	Mikropskeem krüptokiibi katsepartiist (katse eksemplar)
Silur	Krüptokiibi tarkvaraline emulaator [SILUR]
SVM	Krüptokiibi siluri liides viimase ühendamiseks reaalse seadmega [SVMDOK]
SV	Sisend-väljund

4. NÕUDED

4.1 KIIBI VIIGUD

Kiip asub CLCC68 korpuses. Korpuse viikude asetus on joonisel 2.



Joonis 2. CLCC68 korpus altvaates

Kiibi viikude asetus on antud tabelis 1.

Tabel 1. Kiibi viikude loetelu

nr	nimetus	tüüp[1]	funktsioon
1	dataio<2>	ios2k	andmed ja mäluaadress, bitt 2
2	dataio<1>	ios2k	andmed ja mäluaadress, bitt 1
3	dataio<0>	ios2k	andmed ja mäluaadress, bitt 0
4	rom_dat<7>	ios2k	koodimälu aadressiin, bitt 7
5	rom_dat<6>	ios2k	koodimälu aadressiin, bitt 6
6	rom_dat<5>	ios2k	koodimälu aadressiin, bitt 5
7	rom_dat<4>	ios2k	koodimälu aadressiin, bitt 4
8	rom_dat<3>	ios2k	koodimälu aadressiin, bitt 3
9	rom_dat<2>	ios2k	koodimälu aadressiin, bitt 2
10	Ühendamata	-	-
11	Ühendamata	-	-
12	Ühendamata	-	-
13	rom_dat<1>	ios2k	koodimälu aadressiin, bitt 1
14	GND SV	-	mass SV lülidele
15	rom_dat<0>	ios2k	koodimälu aadressiin, bitt 0

[1] Viigu tüübi nimetused tehnoloogia teegis [ES210].

16	ram_me	ops2u	mälu sünkrosignaal	
17	bitio_rdy	ops2u	BITIO valmis	
18	id_in_rdy	ops2u	IDEA sisend valmis	
19	idio_rdy	ops2u	IDEA ja SV valmisolek	
20	main_rdy	ops2u	Kiip valmis reziime vahetama	
21	waiting	ops2u	DEBUG reziim aktiivne	
22	bist_out	ops1u	BISTi väljund	
23	testdata	ops1u	automaatide olekute kontroll	
24	bist_clk	ips8c	BISTi kella valik	
25	do_clk	ips8c	protsesside käivitaja	
26	VDD CORE	-	toide loogikale	
27	Ühendamata	-	-	
28	Ühendamata	-	-	
29	Ühendamata	-	-	
30	GND CORE	-	mass loogikale	
31	dataen	ips8c	andmesiini suund	
32	bitio_mod<1>	ips8c	bitio reziim	bitt 1
33	bitio_mod<0>	ips8c	bitio reziim,	bitt 0
34	io_ram	ips8c	local registri valik io jaoks	
35	id_chnr	ips8c	IDEA kanali valik	
36	commands<4>	ips8c	väliskäsu number,	bitt 4
37	commands<3>	ips8c	väliskäsu number,	bitt 3
38	commands<2>	ips8c	väliskäsu number,	bitt 2
39	commands<1>	ips8c	väliskäsu number,	bitt 1
40	commands<0>	ips8c	väliskäsu number,	bitt 0
41	rd_status	ips8c	koodimälu andmesiini suund	
42	main_mode<1>	ips8c	kiibi tööreziim, bitt 1	
43	main_mode<0>	ips8c	kiibi tööreziim, bitt 0	
45	bist_sel<1>	ips8c	BISTi väljundi valik	
46	rnd_clk	ips8c	juhuarvugeneraatori sisend	
47	run_clk	ips8c	kiibi sünkrosignaal	
48	ext_rom	ips8c	välise koodimälu valija	
49	run	ips8c	DEBUG-t. väljumise lubaja	
50	bist_sel<0>	ips8c	BISTi väljundi valik	
51	bist_test	ips8c	BISTi testreziimi valik	
52	reset	ips4e	asünkroonne lähtestamine	
53	dataio<15>	ios2k	andmed ja mäluaadress, bitt 15	
54	VDD IO	-	toitepinge sisend-väljundile	
55	dataio<14>	ios2k	andmed ja mäluaadress, bitt 14	
56	dataio<13>	ios2k	andmed ja mäluaadress, bitt 13	
57	dataio<12>	ios2k	andmed ja mäluaadress, bitt 12	
58	dataio<11>	ios2k	andmed ja mäluaadress, bitt 11	
59	dataio<10>	ios2k	andmed ja mäluaadress, bitt 10	
60	dataio<9>	ios2k	andmed ja mäluaadress, bitt 9	
61	Ühendamata	-	-	
62	VDD CORE	-	toitepinge loogikale	
63	dataio<8>	ios2k	andmed ja mäluaadress, bitt 8	
64	dataio<7>	ios2k	andmed ja mäluaadress, bitt 7	
65	dataio<6>	ios2k	andmed ja mäluaadress, bitt 6	
66	dataio<5>	ios2k	andmed ja mäluaadress, bitt 5	

67	dataio<4>	ios2k	andmed ja mäluaadress, bitt 4
68	dataio<3>	ios2k	andmed ja mäluaadress, bitt 3

4.2 FÜÜSILISED TINGIMUSED

4.2.1 Toitepinge

	min	max	ühik
Lubatud piirväärtus:	-0.5	5.5	V
Tööpiirkond	4.5	5.5	V

4.2.2 Signaalid

Lubatud piirväärtus:	-0.5	toitepinge + 0.5	V
Loogiline "1" nivoo	70%	100%	protsenti toitepingest
Loogiline "0" nivoo	0	30%	protsenti toitepingest
Sisendvool	-	150	nA

4.2.3 Võimsustarve

Staatiline võimsustarve	-	200	µW
Dünaamiline võimsustarve	-	500	mW

Kui kiipi koormata staatiliste koormustega, siis lisandub võimsustarbele väljundvoolust (p. 4.2.4) tingitud võimsustarve.

4.2.4 Sisend-väljund koormused

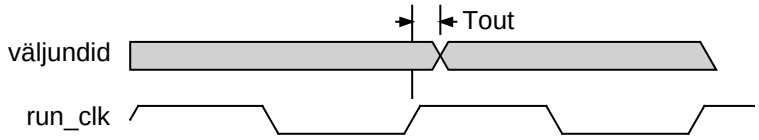
Voolud ja mahtuvused ei tohi ületada viigu lubatud piirparameetreid, mis on vastavalt tüübile järgnevad:

nimi	tüüp	maksimaalne lubatud väljund koormus	maksimaalne lubatud väljund vool	maksimaalne sisend mahtuvus
ios2k	s/v	100 pF	8 mA	3 pF
ips4e	s	-		3 pF
ips8c	s	-		3 pF
ops1u	v	100 pF	4 mA	-
ops2u	v	100 pF	8 mA	-

Need parameetrid on antud keskonna maksimaalsetel piirtingimustel (p. 6.2).

4.2.5 Signaalide viivised

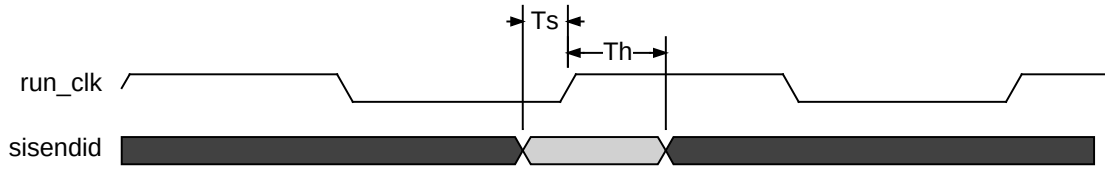
Kiip on sünkroonne ja kõik viivised on antud sünkrosisendi *run_clk* suhtes (Joonis 3).



viik	min	max	ühik	nimetus	kommentaar
kõik väljundid	2	10	ns	Tout	väljundviivitus

Joonis 3. Väljundviivis

Sisendsignaalid peavad olema stabiilsed sünkrosignaali positiivse frondi suhtes (Joonis 4).



sisendi ajahetk	min	max	ühik	nimetus	kommentaar
pre front	5	-	Ns	T_s	setup delay
post front	0	-	Ns	T_h	hold delay

Joonis 4. Sisendviivised

Kiip on staatiliste registritega ning maksimaalselt koormatud viikudega võimaldab tööd sagedustel:

Viik	min	max	ühik	kommentaar
run_clk	0.1	20	MHz	taktsagedus

4.3 KESKKONNA TINGIMUSED

Kiip on ette nähtud tööstuslikuks (industrial) kasutamiseks.

4.3.1 Töötingimused

parameeter	min	max	ühik	kommentaar
Temperatuur	-40	+85	C	väliskeskkonna temperatuur
Niiskus	-	80	%	suhteline niiskus

4.3.2 Hoidetingimused

parameeter	min	max	ühik	kommentaar
Temperatuur	-65	+150	C	väliskeskkonna temperatuur
Niiskus	-	80	%	suhteline niiskus

4.4 ANDMEVAHETUS PORTIDE KAUDU

Järgnevatel graafikutel ei ole eraldi toodud sünkrosignaali *run_clk*. Punktis 4.2.5 antud nõuded viiviste kohta kehtivad kõikide järgnevate diagrammide juures.

4.4.1 Kiip valmis tööreziimi muutuseks

Main_ready, 1 bitt, väljund.

Signaal määrab kiibi valmisoleku reziimide muutmiseks järgnevalt:

Main_rdy	kommentaar
1	Kiip on valmis reziimi muutma või täitma väliskäsku
0	Kiibi tööreziimis on protsess pooleli

4.4.2 Kiibi tööreziimi valik

Main_mode(0) ja main_mode(1), 2 bitti, sisend.

Need viigud kontrollivad kiibi põhilist tööreziimi:

Main_mode(0)	Main_mode(1)	reziim
0	0	IDEA
0	1	Sisend/väljund (SV)
1	0	Sisend/väljund (SV)
1	1	Modulaararitmeetika

4.4.3 Andmevahetussiini laiuse juhtija

Bitio_mod(1) ja Bitio_mod(0), 2 bitti, sisend.

SV reziimis on andmevahetuse järk valitav, kas 16, 8 või 1 bitt korraga. Siini laius on valitav juhtsignaalidega bitio_mod(1) ja bitio_mod(0) enne andmevahetusprotsessi algust järgnevalt:

Bitio_mod(0)	Bitio_mod(1)	siini laius
0	0	16 bitti s/v
0	1	08 bitti s/v
1	0	01 bitti s/v
1	1	01 bitti s/v

Juhul, kui on tegemist vähema, kui 16 bitise laiuse andmevahetusega on andmed siini madalamate järkude peal. Seega toimub 8 bitine SV läbi dataio(7) kuni dataio(0). Järjestikune andmevahetus toimub mööda siini madalamat järku dataio(0).

4.4.4 Koodimälu aadresssiin/SV

Dataio(15) kuni dataio(0), sisend/väljund, 16 bitti.

Modulaaaritmeetika reziimis väljastab kiip siini peale koodimälu aadressid. SV reziimi korral toimub selle siini kaudu andmevahetus kiibi registermälu ning väliskeskkonna vahel. Siini suuna määrab *Dataen* signaal.

4.4.5 SV suund

Dataen, sisend, 1 bitt.

Siini suuna määrab kiibi kõikides tööreziimides *dataen* signaal järgnevalt:

Dataen	suund	kommentaar
1	Kiip->väliskeskkond	Andmete väljastus.
0	Väliskeskkond->kiip	Andmete sisestus.

Andmed on vastavalt kiibi tööreziimile kas käsu aadressid või registermälu sisu.

4.4.6 Koodimälu andmesiini suund

read_stat, sisend, 1 bitt.

Siini suuna määrab kiibi kõikides tööreziimides *read_stat* signaal järgnevalt:

Read_stat	suund	kommentaar
1	Kiip->väliskeskkond	Registrite ning olekute väljastus
0	Väliskeskkond->Kiip	Koodimälu väärtuste sisestus

4.4.7 Koodimälu andmesiin/oleku väljastaja

Rom_dat(7) kuni rom_dat(0), sisend/väljund, 8 bitti.

Modulaararitmeetika reziimi korral loetakse selle siini kaudu välisest koodimälust aritmeetika juhtkäske. SV reziimis on sellest pordist võimalik välja lugeda kiibi

indeksite arvutamise automaadi olekuregistreid. Neid registreid kasutab kiibi emulaator silumisreziimis, et kontrollida juhtautomaatide tööd. Siini suuna määrab *read_stat* signaal.

4.4.8 Väliskäsu number/ Registri number

commands(4) kuni commands(0). Sisend, 5 bitti.

Modulaararitmeetika reziimis antakse väljastpoolt käivitatava väliskäsu number (0 kuni 31), SV reziimis aga väliskeskonnaga andmevahetuses osaleva mäluregistri number (0 kuni 15).

4.4.9 Koodilugeja pinumälu isetesti kontroll

Bist_sel(0), bist_sel(1). Sisend, 2 bitti.
Bist_test. Sisend, 1 bitt.
Bist_clk. Sisen, 1 bitt.
Bist_out. Väljund, 1 bitt.

Pinumälu isetesti aktiveerimiseks tuleb signaalidele omistada järgmised väärtused:

bist_sel(0):=" 1"
bist_sel(1):=" 1"
bist_test:=" 1"

Pärast seda tuleb *Bist_clk* sisendisse anda 128 impulssi sagedusega kuni 10 MHz. *Bist_clk* järgneva 8 impulsi positiivse frondi ajal nihutatakse *bist_out* väljundist välja analüsaatori signatuur madalam bitt kõigepealt. Korras mälu puhul peab signatuur olema " 11011011" .
Kiibi normaalse töö korral peavad sisetesti juhtsignaalid olema järgmiste väärtustega:

bist_sel(0):=" 0"
bist_sel(1):=" 0"
bist_test:=" 0"
bist_clk:=" 0"

NB. Sisetest hävitab pinumälu sisu.

4.4.10 Kiibi juhtautomaatide olekute kontrollsumma väljund

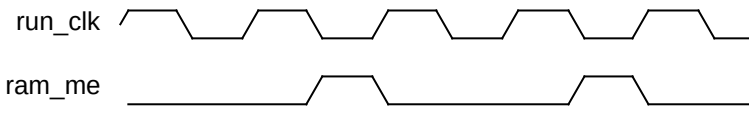
Testdata. väljund, 1 bitt.

Signaal väljastab reaalajas automaadi olekute kontrollsummat. Kasutatakse esialgsel testimisel ning sisemise arhitektuuri verifitseerimisel. Sisemise arhitektuuri testitakse testitava kiibi ning simuleerimistulemuste võrdlemise abil.

4.4.11 Mälu sünkrosignaal

Ram_me, väljund, 1 bitt.

Sisemise mälu sünkrosignaal. Signaal on alati järgneva kujuga:



Joonis 5. Mälu sünkrosignaal

4.4.12 SV automaadi olek

bitio_rdy. Väljund, 1 bitt.
id_in_rdy. Väljund, 1 bitt.
idio_rdy. Väljund, 1 bitt.

Nende signaalide seletust vaata altpoolt sisend-väljund protsessi kirjelduse juurest (p. 4.4.2.4).

4.4.13 Kiibi töö seisatatud läbi silumiskäsu WAIT.

Waiting. Väljund, 1 bitt.

Kiibi modulaararitmeetika reziimis on koodi täitmisel jõutud käsuni `wait`. Käsk seisatab programmi töö ning viib kiibi SV reziimi kuni impulsini *run* sisendil (p. 4.4.14). Signaal näitab kiibi olekut järgnevalt:

Waiting	kommentaar
1	Modulaararitmeetika automaat seisab ning kiip on IO reziimis.
0	Kiibi normaalne töö.

4.4.14 Kiibi töö jätkamine pärast WAIT silumiskäsku.

Run. Väljund, 1 bitt.

Kiibi modulaararitmeetika reziimis silumisreziimist väljumine. Kui *waiting* signaal on aktiivne siis signaali *run* positiivne väärtus üle ühe kellatakti pikkuse viib kiibi välja IO reziimist ja jätkab käsutäitmist `wait` käsule järgnevalt käsult.

4.4.15 Kiibi akumulaatori SV

Io_ram. Sisend, 1 bitt.

Kiibi SV protsessis saab akumulaatorit kirjutada nullindasse registrisse ning lugeda sealt. See signaal valib akumulaatori teisaldamise või normaalse SV vahel järgnevalt:

Io_ram	kommentaar			
1	normaalne IO			
0	akumulaatori teisaldamine			
	a) dataen="0":	AKU	->	R0
	b) dataen="1":	R0	->	AKU

4.4.16 IDEA kanali numbri valik

Id_chnr. Sisend, 1 bitt

Signaal valib IDEA teisenduse korral laiendatud võtmebloki algusaadressi registermälus ning lubab sellega valida krüpteerimise ning dekrüpteerimise vahel järgnevalt:

Id_chnr	kommentaar
0	IDEA võtmed valitakse sisemisest mälust registrist I0
1	IDEA võtmed valitakse sisemisest mälust registrist I1

4.4.17 Välise või sisese koodimälu valik.

Ext_rom, sisend, 1 bitt.

Välise ja sisese koodimälu vahel valija. Kui valitakse väline koodimälu, siis loetakse kood *rom_dat* pordilt ning aadressid antakse *dataio* pordile. Sisemise mälu korral loetakse modulaararitmeetika juhtkoodid sisemisest püsimälust. Väline reziim võimaldab kiibi koodi siluda kasutades selleks välis laetavat koodimälu. Seda signaali loetakse ainult lähtestamistsükli ajal. Kui sellel ajal oli *ext_rom* "1", siis täidetakse aritmeetikakäske läbi välise koodimälu, vastasel korral kasutatakse sisest koodimälu. Sisemises koodimälus ei ole *wait* käsku, ning seetõttu ei ole võimalik siseneda silumisreziimi ning selles reziimis lugeda välja registermälu kogu sisu. Kui on valitud väline käsumälu, siis on võimalik tagasi lülitada sisemisele koodimälule koodi verifitseerimise eesmärgil. Ümberlülitamine toimub signaal *ext_rom* negatiivse nivoo peale. Kui valida kiibi koodi- ning aadressiini väljundsuund signaalidega dataen ja read_stat ning juhtida kiibi sünkrosignaali takthaaval on võimalik täidetavat koodi jälgida. Samuti on võimalik sisemise koodi täitmisest ümber lülitada välise koodi täitmisele viies ext_rom tagasi aktiivsele nivoole. Kuna me teadsime koodi aadressi, siis on võimalik niimoodi sisemise koodi vahele lisada *wait* käske ning lugeda välja sisemine registermälu.
Sellised protseduurid on tarvilikud kiibi koodi verifitseerimiseks võimaliku sisseehitatud salaukse avastamiseks. Raudvara kontrollimiseks tuleb kasutada automaatide olekute kontrollsummat (p. 4.4.10).

Ext_rom	kommentaar
1	Kiip kasutab välist koodimälu
0	Kiip kasutab sisemist koodimälu

4.4.18 Kiibi sünkrosignaal

run_clk. Sisend, 1 bitt.

4.4.19 Asünkroonne lähtestamine.

Reset. Sisend, 1 bitt.

4.4.20 Välise juhuarvugeneraatori sisend

rnd_clk. Sisend, 1 bitt.

Väline asünkroonnne juhuarvugeneraator võimaldab tõsta entroopiat ning suuurendada süsteemi turvalisust. See signaal liidetakse sisemise juhuarvugeneraatori väljundiga läbi 24 bitise LFSR generaatori polünoomiga X(0)⊗X(15)⊗X(23) positsioonil 16. Väline generaator peab olema ühtlase jaotusega valge müra generaator sagedusdiapasoonis 0.1-Fclk.

4.4.21 Protsesside aktiveerija

do_clk. Sisend, 1 bitt.

Kiibi kõikide tegevuste käivitaja. Juhib kõiki tegevusi peale WAIT reziimist väljumise, mille jaoks on sisend *run* (p. 4.4.14).

4.5 KÄSKUDE TÄITMINE JA KÄSUSÜSTEEM

Kiibi aritmeetikaautomaadi käsusüsteem koosneb 27st käsust (Tabel 2). Nende käskudega on võimalik realiseerida modulaararitmeetikale baseeruvaid krüptosüsteeme ning inverteerida IDEA võtit. Käske täidetakse vastavalt allpoolkirjeldatud tabelile. Käsu `WAIT` täitmise tagajärjel minnakse silumisreziimi. Pärast silumisreziimist väljumist täidetakse programmi `WAIT` käsule järgnevalt realt.

Käsusteemi kirjeldamisel kasutatakse järgnevaid tähistusi:

Ra, Rb, Rc:	Registermälu aadressid, kodeeringuid vt. Tabel 3.
NUM13:	13-bitine positiivne täisarvuline konstant.
DP8:	8-bitine relatiivne aadress, - 127 … + 128.
DP13:	13-bitine relatiivne aadress, - 2043 … + 2044.
LB16:	16-bitine absoluutne aadress.
NUM16:	16-bitine positiivne täisarvuline konstant.

Tabel 2. Kiibi käsustik

Nr	käsk	argumendid	kommentaar
			Aritmeetika käsud
0	ADD	Ra, Rb, Rc	Ra := Rb + Rc
1	DEX	Ra, Rb, Rc	Ra := Ra^{Rb} mod Rc
2	DMUL	Ra, Rb, Rc	Ra := Ra × Rb mod Rc
3	DIV	Ra, Rb, Rc	Rb := Ra / Rc
4	MOD	Ra, Rb, Rc	Rb := Ra % Rc
5	MUL	Ra, Rb, Rc	Rc := Ra × Rb
6	ADI	Ra, NUM13	Ra := Ra + NUM
7	SBI	Ra, NUM13	Ra := Ra - NUM
8	SUB	Ra, Rb, Rc	Rc := Ra - Rb
			Juhtimise muutmine
9	JLE	Ra, Rb, DP8	if(Ra ≤ Rb) goto PC+DP8

tabeli 2 järg

Nr	käsk	argumendid	kommentaar
10	JLT	Ra, Rb, DP8	if(Ra < Rb) goto PC+DP8
11	JZ	Ra, DP13	if(Ra == 0) goto PC+DP13
12	JNZ	Ra, DP13	if(Ra != 0) goto PC+DP13
13	JP	Ra, DP13	if(Ra(0)== 0) goto PC+DP13
14	JMP	LB16	goto LB16
15	LOOP	LB16	if(CX > 0) goto LB16; CX--
16	LOOPNZ	LB16	if(CX ≥ 0) goto LB16; CX--
			Andmeteisaldus
17	MOV	Ra, Rb	Ra := Rb
18	MF4	Ra	Ra := 0x10001
19	MVI	Ra, NUM13	Ra := NUM13
20	MVIN	Ra, NUM13	Ra := - NUM13
21	LCX	NUM13	CX := NUM13
22	RND	Ra	Ra := juhuslik arv
			Alamprogrammi käsud
23	CALL	LB16	PUSH PC, goto LBL
24	RET		POP PC
			DEBUG reziimi juhtimine
25	WAIT		peatab programmi kuni run signaali pos. frondini
			Väliskäsu lõpetamine
26	RTI	NUM16	Väliskäsu lõpp koodiga NUM
27	NOP		Tühi käsk

Enne käskude kodeeringut vaatame registermälu adresseerimist. Mälu on jaotatud 16ks 768 bitiseks pikaks registriks R0 kuni R15 või 32ks lühikeseks registriks R0L, R0H,…R15L, R15H. Lühikeste registrite puhul jagatakse pikk register pooleks. Kogu mälu ei ole võimalik adresseerida lühikeste registritega, seda saab teha vaid ülemiste 8 registri korral. Samal ajal ei ole alumised registrid kasutatavad otse assemblertasemel,

ning seega on kogu assemblertaseme mälu adresseeritav nii lühikeste kui pikkade registritega.

Tabel 3. Kiibi registrid

Nr	Pikk register	Lühikesed registrid	kodeeringud pikk	lühikesed		otstarbe
0	Tr	-	00	-	-	Alu register 1
1	T1	-	01	-	-	Alu register 2
2	I0	-	02	-	-	IDEA kanal 0
3	I1	-	03	-	-	IDEA kanal 1
4	Sr	-	04	-	-	Olekuregister
5	N	-	05	-	-	Aalik võti 1
6	M	-	06	-	-	Aalik võti 2
7	D	-	07	-	-	Slajane võti
8	-	P/Q	08	10	11	n=p×q
9	-	Pu/Qu	09	12	13	Uued algarvud
10	-	S/A5L	0A	14	15	$S=p^{-1}\times q$
11	A0	A0L/A0H	0B	16	17	Üldotstarbeline

Tabeli 3 järg

Nr	Pikk register	Lühikesed registrid	kodeeringud pikk	lühikesed		otstarbe
12	A1	A1L/A1H	0C	18	19	Üldotstarbeline
13	A2	A2L/A2H	0D	1A	1B	Üldotstarbeline
14	A3	A3L/A3H	0E	1C	1D	Üldotstarbeline
15	A4	A4L/A4H	0F	1E	1F	Üldotstarbeline

Kõik käsud on kuni kolme baidised. Käskusid on seitse erinevat tüüpi. Tabelis 4 toome iga tüübi kodeeringu:

Tabel 4. Käskude bitiväljad

	Bait 1								Bait 2								Bait 3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
1	-	-	c	c	c	c	c	c																
2	-	a	c	c	c	c	c	c	-	-	-	-	a	a	a	a								
3	b	a	c	c	c	c	c	c	b	b	b	b	a	a	a	a								
4	n	a	c	c	c	c	c	c	n	n	n	n	a	a	a	a	n	n	n	n	n	n	n	n
5	b	a	c	c	c	c	c	c	b	b	b	b	a	a	a	a	-	-	-	c	c	c	c	c
6	-	-	c	c	c	c	c	c	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n
7	b	a	c	c	c	c	c	c	b	b	b	b	a	a	a	a	d	d	d	d	d	d	d	d

Kõikidel käskudel paikneb kood esimese baidi bittides 5 kuni 0 millega on võimalik kodeerida 64 erinevat käsku. Käsukoodi ülemistes bittides on vastavalt tüübile kas registrite (2,3,5,7) või numbrilise konstandi (4) ülemised bitid. Tabelis 4 kasutatakse järgnevaid tähistusi:

c - käsukood
a - registri kood ühe registriga käsu korral
b - registri kood kahe registriga käsu korral

c - registri kood kolme registriga käsu korral
n - 13 bitine positiivne täisarv
d - 8 bitine täisarv

Kõikide eelpoolnimetatud kodeeringute loetakse bitte vasakult paremale, kusjuures vasakul olev bitt on kõige kõrgem järk.

Näide: konstant n on kolmeteistkümne bitine. Tema kodeering on kolmes baidis järgmiselt:

Bait 1, 7 bitt: N(12)
Bait 2, 7-4 bitt: N(11:8)
Bait 3, 7-0 bitt: N(7:0)

Käsutüüpidele vastavad käsud tabelis 4 on järgmise kujuga:

1) Ilma argumendita
   RET, WAIT, NOP
2) Ühe registerargumendiga
   MF4, RND
3) Kahe registerargumendiga
   MOV
4) Ühe register- ning ühe 13-bitise konstantargumendiga (number või suhteline aadress)
   ADI, SBI, JZ, JNZ, JP, MVI, MVIN, LCX
5) Kolme registerargumendiga
   ADD, DEX, DMUL, DIV, MOD, MUL, SUB
6) Ühe 16-bitise konstantargumendiga
   JMP, LOOP, LOOPNZ, CALL, RTI
7) Kahe register- ning ühe 8-bitise konstantargumendiga
   JLE, JLT

Juhul, kui käsukoodi ei eristata täidab kiibi aritmeetikaautomaat vastavad indeks- ja kostandi registrid ja kutsub välja mikrokoodi käsu vastavalt koodimälu alguses olevale teisendustabelile. Selline lähendus võimaldab ülemise taseme käske juurde lisada. Tundmatu käsu pikkus on alati 3 baiti. Seega saab juurde lisada 4 kuni 7 tüübi käske. Kui ignoreerida argumente võib lisada ka 1-3 tüübi käske, kuid sellisel juhul tuleb arvestada koodimälu pikkuse suurenemisega.
Käskude koodid ning tüübid on tabelis 5.

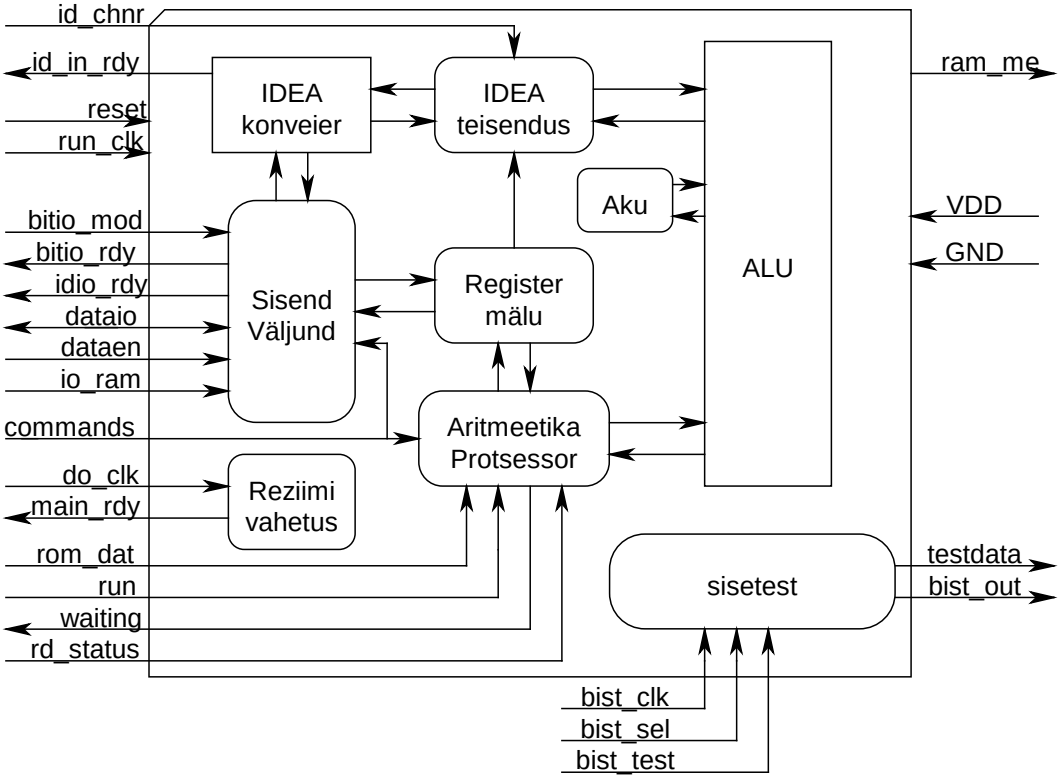
Tabel 5. Käskude kodeeringud

Nr	Käsk		tüüp	kood
0	ADD	Ra, Rb, Rc	5	17
1	DEX	Ra, Rb, Rc	5	18
2	DMUL	Ra, Rb, Rc	5	19
3	DIV	Ra, Rb, Rc	5	20
4	MOD	Ra, Rb, Rc	5	21
5	MUL	Ra, Rb, Rc	5	22
6	ADI	Ra, NUM13	4	23
7	SBI	Ra, NUM13	4	24

8	SUB	Ra, Rb, Rc	5	25
			Juhtimise muutmine	
9	JLE	Ra, Rb, DP8	7	0
10	JLT	Ra, Rb, DP8	7	1
11	JZ	Ra, DP13	4	2
12	JNZ	Ra, DP13	4	3
13	JP	Ra, DP13	4	4
14	JMP	LB16	6	5
15	LOOP	LB16	6	6
16	LOOPNZ	LB16	6	
			Andmeteisaldus	
17	MOV	Ra, Rb	3	7
18	MF4	Ra	2	8
19	MVI	Ra, NUM13	4	9
20	MVIN	Ra, NUM13	4	10
21	LCX	NUM13	6	12
22	RND	Ra	2	13
			Alamprogrammi käsud	
23	CALL	LB16	6	14
24	RET		1	15
			DEBUG reziimi juhtimine	
25	WAIT		1	45
			Väliskäsu lõpetamine	
26	RTI	NUM16	6	31
			Muud käsud	
27	NOP	-	0	46

4.6 KIIBI FUNKTSIOONID

Kiibi täpsustatud sisemine arhitektuur on allpool oleval joonisel.



Joonis 6. Kiibi arhitektuur

Kiibi funktsioonid võib jagada kolme suurde gruppi järgnevalt

1) Sisend-väljund funktsioonid
2) Käsusüsteem
3) IDEA krüpteerimine

Kuna kiibil puudub sisemine koodimälu peame me aritmeetikakäskude täitmiseks kasutama välist koodimälu.

Kiibi SV funktsioone kasutatakse silumis- ja SV reziimis registermälu väljalugemiseks ning kirjutamiseks läbi 1/8/16-bitise siini.

Võtmevahetusautomaat kasutab aritmeetikaprotsessorit, AKUt registermälu ning ALU.

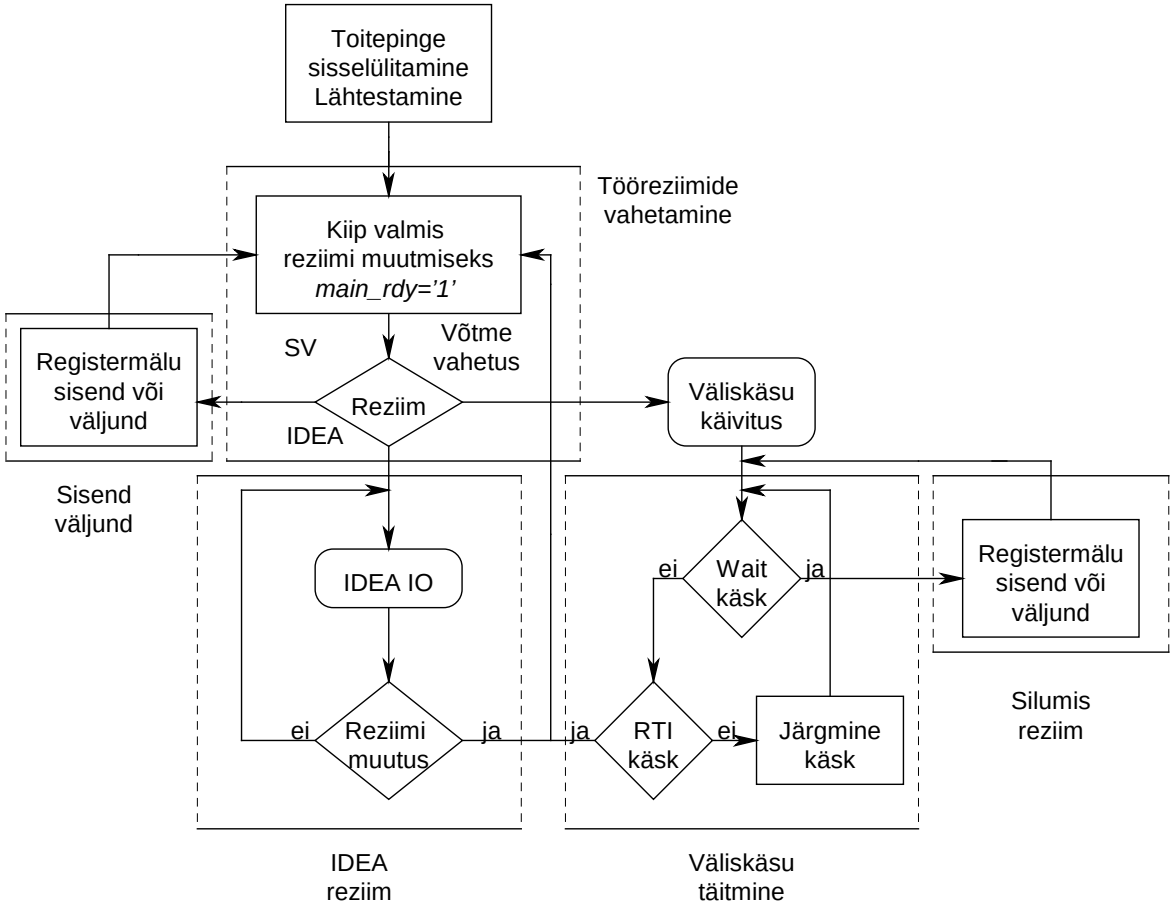
IDEA krüpteerimine kasutab SV automaati, IDEA konveierit ning ALU. Samuti kasutatakse registermälu võtmete lugemiseks.

Sisetest töötab pidevalt ning annab infot kiibi korrasoleku kohta.

Kiibil on erinevat neli erinevat tööreziimi:

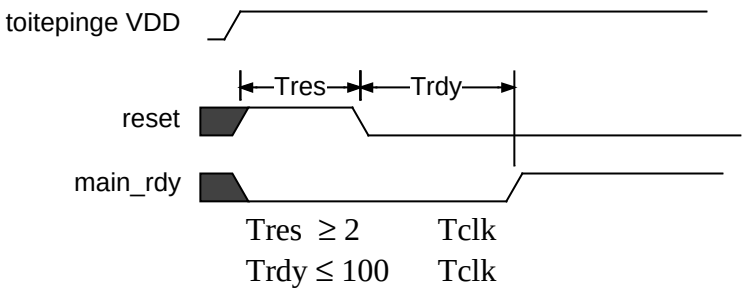
1) Sisend-väljund
2) IDEA reziim
3) Väliskäsu täitmine
4) Silumisreziim

Üleminek nende vahel toimub järgneva diagrammi alusel:



Joonis 7. Kiibi töödiagrammid

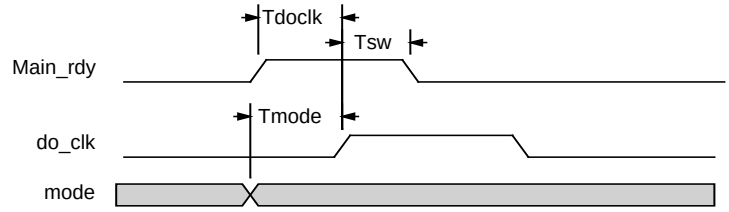
4.6.1 Lähtestamine



Joonis 8. Kiibi lähtestamine

Kiibi lähtestamine algab pärast toitepinge sisselülitamist *reset* signaali aktiveerimisega. *Reset* peab olema aktiivne vähemalt kahe sünkrosignaali run_clk perioodi pikkuselt. Reset signaali negatiivsest frondist alates kuni kiibi valmisolekuni viivitus Trdy on maksimaalselt 100 sünkrosignaali perioodi.

4.6.2 Tööreziimide vahetamine

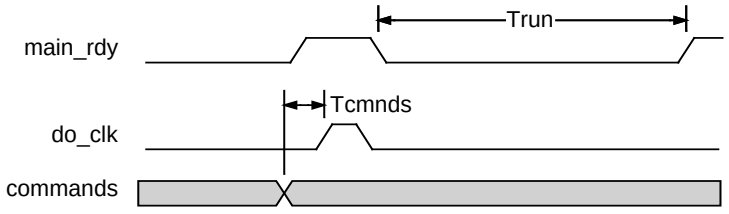


Tdoclk ≥ 1 Tclk
Tmode ≥ 0
Tsw ≤ 20 Tclk

Joonis 9. Kiibi tööreziimide vahetamine

Kiibi tööreziime saab vahetada kui kiibi valmisoleku signaal *main_rdy* on "1". Selles olekus ootap kiip *do_clk* positiivset nivood, mis initsialiseerib reziimide vahetuse. Enne *do_clk* impulssi peab olema õige reziim kiibi viikudel *main_mode*.

4.6.3 Väliskäsu käivitamine



Tcmnds ≥ 0
Trun sõltub väliskäsu programmi pikkusest

Joonis 10. Väliskäsu käivitamine

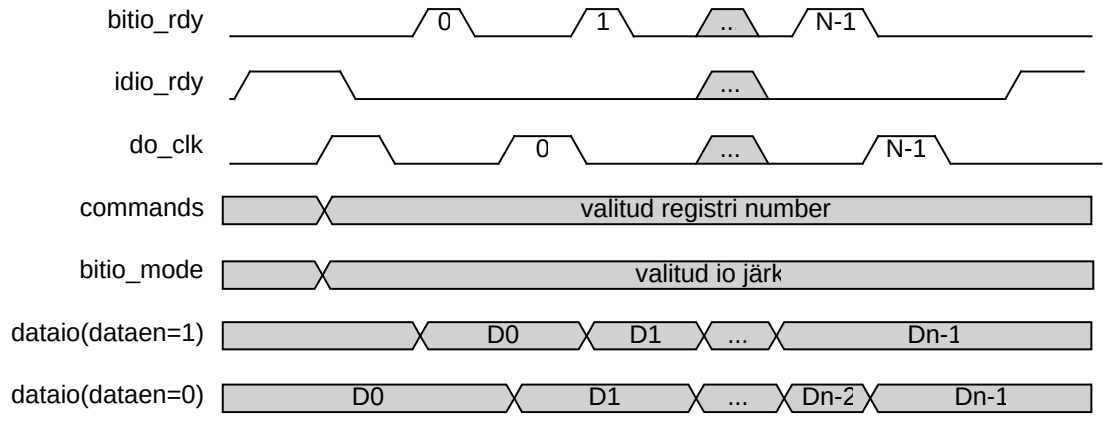
Kui on valitud modulaararitmeetika reziim, siis pärast reziimivahetust käivitab kiip väliskäsu, mille number loetakse siinilt *commands*. Väliskäsu lõppu signaliseerib kiip viies *main_rdy* uuesti aktiivseks.

4.6.4 Registermälu sisend/väljund

Reziim võimaldab kiibi registreid lugeda ja kirjutada väljastpoolt, rikkumata seejuures kiibi aritmeetikaautomaadi olekut. Seda on vaja silumisreziimis registrite sisu jälgimiseks ning SV reziimis võtmevahetusväljade vahetuseks. 768-bitiseid registreid saab lugeda ja kirjutada 16, 8 ja 1 biti kaupa. Vastavalt võtab SV protsess aega N takti. N on erinevate IO reziimide korral järgmine:

bitio_mode	N
00	48
01	96
10	768
11	768

SV protseduur algab idio_rdy signaali kõrge nivooga, millega kiip signaliseerib SV automaadi valmisolekut. Järgneva do_clk impulsiga sisestatakse(dataen=0) või väljastatakse(dataen=1) N-s osa registri M sisust. Registri number M loetakse kiibi väliskäsusiinilt *do_clk* impulsi positiivse frondi ajal. Kui kõik N osa on läbi, siis viimase osa SV järel läheb idio_rdy uuest aktiivseks signaliseerides kiibi järgneva registri SV valmisolekut.



Joonis 11. Registermälu sisend-väljund

4.6.5 IDEA reziim

IDEA reziimis käivitatakse teisendus andmete kirjutamisega teisenduse sisendregistritesse. SV diagramm on sarnane registermälu SVga. Erinevus on andmeregistri pikkuses, mis on 64 bitti. N on seega järgmine:

Bitio_mod	N
00	4
01	8
10	64
01	64

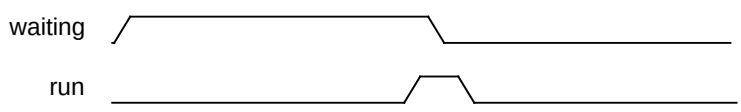
Samuti määrab IDEA korral sisendi IO valmisoleku idio_rdy ning väljundi IO valmisoleku idio_rdy. Kui siseneda IDEA reziimi esimest korda, on id_in_rdy aktiivne, näidates teisenduse sisendkonveieri valmisolekut. Idio_rdy on madal, kuna ühtegi teisendust pole veel toimunud. Edasine andmetöötlus toimub järgnevalt:

1) Vali SV järk siiniga *io_mode*
2) Vii läbi SV protseduur, nagu registermälu puhul, arvestades et:
   - SV automaadi valmidust näitab *id_in_rdy*
   - Viimase andmelõigu sisestamisega läheb signaal *id_in_rdy* aktiivseks, näidates võimalust sisestada järgnevad andmed IDEA kanalisse ootama.
3) Kui on veel andmeid, sisestage need eelpoolmainitud protseduuri jälgides.
4) Kontrolli *idio_rdy*. Juhul, kui see signaal läheb aktiivseks on IDEA kanal lõpetanud andmete töötlemise ning resultaadi võib välja lugeda. Resultaadi väljalugemine toimub samuti kui sisselugemine, va. suund vahetatakse signaaliga *dataen*

Konveieri kasutamine SV juures võimaldab kasutada IDEA kanalit täie võimsusega. Vastasel korral kahaneks krüpteerimiskiirus, kuna IGA teisenduse järel (50 takti) peaks ootama SV järel minimaalselt 4*4*2=36 takti. Nüüd aga võib need operatsioonid teostada paralleelselt teisendusega. Hoolt tuleb kanda andmete korrektse ajalise järgnevuse säilitamise eest. Kuna väljundi ülekirjutamisel ei väljastata veasignaali peab väline liides hoolitsema väljundi väljalugemise eest enne uute andmete poolt ülekirjutamist.
IDEA reziimist väljumiseks tuleb *main_mode* sisendile panna soovitava reziimi number. Selle peale kiip väljub IDEA reziimist ning ootab do_clk impulssi, et siseneda valitud reziimi.

4.6.6 Silumisreziim

võimaldab kiibi väliskäsu programmset seiskamist ning hilisemat käivitamist väljastpoolt. Programm seiskub assemblerikäsu WAIT peale seni kauaks, kuni kiibi *run* sisendisse pole antud positiivset impulssi. Silumisreziimis on kiibi väljund *waiting* ning SV automaat aktiivne.



Joonis 12. Silumisreziimist väljumine

Silumisreziimis on võimalik peale registermälu lugeda ka indeksite arvutamise automaadi sisemiste registrite väärtusi. Selleks tuleb kiibi käsusiinile panna registri indeks vastavalt alljärgnevale tabelile. 8-bitist väärtust on võimalik lugeda välja läbi koodimälu andmesiini *rom_dat*.

Nr.	*commands*	Nimetus	Kommentaar
00	0000	LOWC	Ra
01	0001	LOWC	Rb
02	0010	LOWC	Rs
03	0011	LOWC	Rd
04	0100	LOWC	Bcmpi
05	0101	FLAGS	EZF, IF, F1, F0, LC, LZ, AC, AZ
06	0110	SD	D768 S768
07	0111	IND	ALUS, KOR
08	1000	ER	ER[07:00]
09	1001	ER	ER[15:08]
10	1010	ER	ER[23:16]
11	1011	EXTC	HIVAL, 0 , ESHR_REG
12	11xx	CFT	BP, SP

4.6.7 SV reziim

SV reziimis on võimalik registermälu sisend ja väljund eelpoolkirjeldatud diagrammi alusel. Pärast registri SV tsüklit on kiip valmis järgmiseks reziimivahetuseks ning teatab sellest aktiveerides signaali main_rdy.

5. KATSEPLAAN

Katsetamisel kontrollitakse alljärgnevaid kiibi funktsioone ja andmevahetust:

5.1 TESTITAVAD FUNKTSIOONID

Tabel 6. Testitavad funktsioonid

Nr.	funktsioon	nõuded (p.p)
1	Lähtestamine	4.6.1
2	Tööreziimi vahetus	4.6.2
3	Sisend/Väljund	4.6.4
4	Registermälu	4.6.4
5	Väliskäsu käivitamine	4.6.3, 4.5
6	Käsusüsteem	4.5
7	Silumisreziimi sisenemine	4.6.3, 4.6.6
8	Silumisreziimi sisend-väljund	4.6.7
9	Silumisreziimist väljumine	4.6.3
10	Väliskäsu lõpetamine	4.5
11	IDEA teisenduse sisend	4.6.7
12	IDEA nulli teisendus andmeosas (0 -> 0x10000)	4.6.5
13	IDEA nulli teisendus võtmeosas (0 -> 0x10000)	4.6.5
14	IDEA teisendus juhuslike andmetega	4.6.5
15	IDEA teisenduse väljund	4.6.7

Kõik valmistud kiibid peavad läbima testimise tabel 6 toodud mahus.

Ülaltoodud funktsioone kontrollitakse integreeritud testiga (p. 5.2) kiibi keskonna normaal- ja piirtingimustel. Kiip loetakse korrasolevaks kui ta läbib kogu integreeritud testi.

5.2 INTEGREERITUD TESTI ALAMETAPID

Tabel 7. Integreeritud testi alametapid

number	Alamtesti nimetus integreeritud testis	funktsiooni nr. p. 5.1 (Tabel 6)
1	Kiibi lähtestamine	1
2	Registermälu test	3, 4
3	Käsureziimi sisenemine	2
4	Testprogramm kuni *wait* käsuni	5,6,7
5	Silumisreziimis registrite välja lugemine	8,9
6	Testprogramm kuni järgmise *wait* käsuni	6,7
7	Testprogramm kuni RTI käsuni	9,10
8	Sisenemine SV reziimi	2
9	IDEA kanali registrite nullimine	3,4
10	IDEA reziimi sisenemine	2
11	IDEA teisendus 0 võtmega ja 0 andemtega	11,13,15
12	IDEA teisendus 0 võtmega ja suvaliste andmetega	11,12,15
13	Sisenemine SV reziimi	2
14	IDEA kanali võtmete sisselugemine	2,3,4
15	IDEA reziimi sisenemine	2
16	IDEA teisendus suvalise võtmega ja 0 andmetega	11,12,15
17	IDEA teisendus suvalise võtmega ja suvalise andmetega	11,14,1

6. KATSETINGIMUSED

Katsetuste seeria toimub kiibi keskkonna normaal ja piirtingimustel 10 minuti jooksul.

6.1 NORMAALTINGIMUSED

Mõjufaktor	Väärtus	lubatud kõikumine	Ühik	Kommentaar
VDD	4.9	± 0.1	V	toitepinge
Fclk	20	± 0.1	MHz	töösagedus
Temp	25	± 5	C	keskkonna temperatuur

6.2 MAKSIMAALSED PIIRTINGIMUSED

VDD	4.5	± 0.1	V	toitepinge
Fclk	20	± 0.1	MHz	töösagedus
Temp	80	± 2	C	keskkonna temperatuur

6.3 MINIMAALSED PIIRTINGIMUSED

VDD	5.5	± 0.1	V	toitepimge
Fclk	20	± 0.1	MHz	töösagedus
Temp	-40	± 2	C	keskkonna temperatuur

Märkus 1: Kiibi viigud peavad katsetuste ajal olema koormarud maksimaalse lubatud koormusega (p. 4.2).

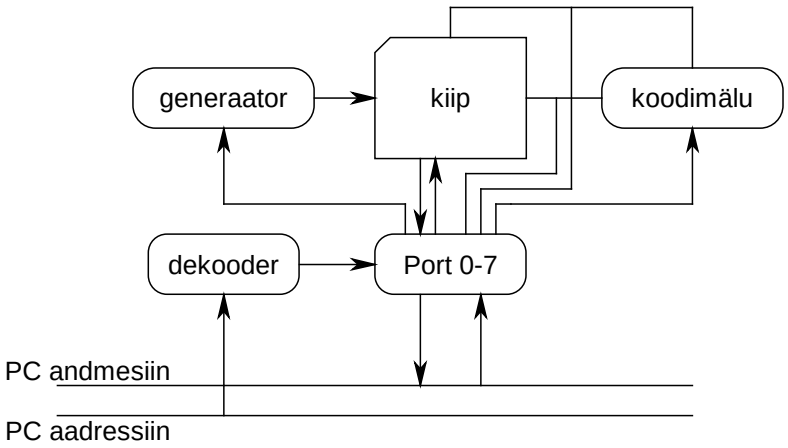
Märkus 2: Kiibi piirtöösageduse leidmiseks tuleb testida kiibi tööd keskkonna piirtingimustel 5 MHz kaupa sagedust suurendadaes kuni integreeritud testi esimese tõrkeni.

Märkus 3. Integreeritud testi tuleb korrata seerias 10 minuti jooksul 1 sek. intervalliga

7. KATSEVAHENDID

7.1 LIIDES KIIBI SISENDITE JUHTIMISEKS JA JÄLGIMISEKS

on PC ISA siini lisamoodul, mis võimaldab kiibi viikusid kontrollida ja juhtida kiibi sünkrogeneraatorit kasutades PC tüüpi arvutit juhtseadmeks ([SVMDOK], LISA M). Selgitame mooduli tööd sellisel määral, kui palju on tarvis testimisprotseduuride teostamiseks.



Joonis 13. SVM liides

Moodul on ehitatud programmeritava loogika baasil, mis võimaldab tema funktsionaalsuse kiiret muutmist. Programmeeritav generaator on PC tüüpi arvutites kasutatav sünkrogeneraator ning koodimälu on PC 32 kilobaidine staatiline mälukiip. Genaraator võimaldab kiibile anda sünkrosignaali sagedusvahemikus 2-100 MHz.
Moodul paigutab kiibi viigud PC SV portide piirkonda seitsmesse porti R0 kuni R7 alates baasist järgmiselt:

baas+	Bittide nr pordis.								
	0	1	2	3	4	5	6	7	suund
0	dataio(0)	dataio(1)	dataio(2)	dataio(3)	dataio(4)	dataio(5)	dataio(6)	dataio(7)	s/v
1	dataio(8)	dataio(9)	dataio(10)	dataio(11)	dataio(12)	dataio(13)	dataio(14)	dataio(15)	s/v
2	romdat(0)	romdat(1)	romdat(2)	romdat(3)	romdat(4)	romdat(5)	romdat(6)	romdat(7)	s/v
3	ioram	id_chnr	cmnds(0)	cmnds(1)	cmnds(2)	cmnds(3)	cmnds(4)	-	s/v
4	ext_rom	run_clk	bist_sel(1)	bist_sel(0)	do_clk	dataen	biomd(0)	biomd(1)	s/v
5	rnd_clk	rd_stat	mmod(1)	mmod(0)	run	reset	bist_tst(1)	bist_tst(0)	s/v
6	testdat	bist_out	waiting	main_rdy	idio_rdy	id_in_rdy	bio_rdy	ram_me	s
7	en_rd	ram_we	ram_oe	clken	clks0	clks1	clks2	clks3	s/v

Realiseeritud moodulis on baas 0x230, mis on ka R0 aadress. R1 on 0x231 jne. kuni R7, mis on 0x237.

Viimases pordis on mälu ja kellageneraatori juhtsignaalid. Ülejäänudest on R0, R1 ja R2 sisendid ühendatud otse kiibi külge. R3-R7 annavad lugedes tagasi sinna kirjutatud väärtuse.
SVM kiibi viikude koormused on maksimaalsed lubatavad koormused (p. 4.2)

7.2 PROGRAMM LIIDESE JUHTIMISEKS

on punktis 7.1 kirjeldatud liidese juhtprogrammi, mis võimaldab sooritada katseplaanis kirjeldatud teste. Programm kasutab [SVMDOK] raames välja töötatud liidese juhtimise alamprogramme. Alamprogrammid kasutavad liidest, et juhtida kiibi tööreziime. Selle saavutamiseks muudavad nad kiibi viikude väärtusi vastavalt p. 4.4.2 toodud diagrammidele kirjutades porti PC SV mälus (p. 7.1).
Kirjeldame alamprogramme, mida kasutatakse kiibi testimise juhtprogrammis.

```
checkregs()
```

Kontrollib liidese registreid ning koodimälu ja initsialiseerib programmi andmestruktuurid. Vea korral teatab sellest terminalile.

```
read_binfile( char *nimi, char *BUF, int *pikkus )
```

Loeb kettafaili nimega *nimi* kogu pikkuses mälusse alates aadressist *BUF*. Mälu peab olema eraldatud enne alamprogrammi väljakutsumist. Salvestab faili pikkuse aadressile *pikkus*. Vea korral teatab sellest terminalile.

```
load_code( int * LO_CODE, int *HI_CODE )
```

Täidab liidese koodimälu mikrokoodi ala aadressilt *LO_CODE* ning assembleri ala aadressilt *HI_CODE*.

```
add_lo_debug()
```

Lisab protseduuri siluri abiprotseduuri käivitamiseks, vajalik registrite väljalugemiseks.

```
mode_io()
```

lülitab kiibi SV reziimi

```
write_regs( int* REGS )
```

Kirjutab kiibi registermälusse massiivi *REGS* sisu.

```
read_regs( int *REGS, int *ACCU )
```

Loeb kiibi registermälu massiivi REGS ja akumulaatori massiivi *ACCU*.

```
save_int_regs( char * nimi, CC * KIIBI_OLEK )
```

Salvestab kiibi oleku, sh. registermälu sisu ning akumulaatori kettafaili nimega *nimi*. Olek loetakse struktuurist, millele viitab *KIIBI_OLEK*.

`waitfor(long bitt, int value )`
Ootab kuni kiibi viigu *bitt* väärtus on *value*

`pulse_run()`
Genereerib impulsi viigul run, tuues sellega kiibi välja silumisreziimist

`enter_idea()`
Pane kiip IDEA reziimi

`idea_encdec_test( int number)`
Sooritab IDEA krüpteerimis-dekrüpteerimistesti, st.
1) sisestab IDEA sisendisse 0 väärtuse
2) krüpteerib selleo 0 kanali võtmega
3) loeb välja tulemuse #0
4) krüpteerib tulemuse 1 kanali võtmega
5) loeb välja tulemuse #1
6) võrdleb lähteandmeid tulemusega #1
7) vea korral teatab sellest terminalile
8) sisestab juhuslikud andmed IDEA sisendisse
9) teeb *number* korda 2-8

Selle protseduuri käivitamisel peab IDEA kanalivõtmed #0 ≡ $(\#1)^{-1}$. Nullise võtme korral on võti ise ka pöördvõti. Nullise võtmega testi nimetatakse involutsioonitestiks.

7.2.1 Registermälu test

Liidese juhtprogrammi siseselt on realiseeritud kiibi registermälu test.

Kiibi registermälu tuleb testida järgneva nelja testiga:

nr.	testi nimetus	tulemused
1.	Konstant 0	RAM_C0.dat
2.	Konstant 1	RAM_C1.dat
3.	Malelaud	RAM_CHKB.dat
4.	Indekseeritud sisendjada	RAM_INDX.dat

Esimese testi tulemusena peab kogu mälu sisu olema "1". Teise testi tulemusena peab mälu sisu olema "0". Kolmanda testi tulemusena peab mälu sisu olema vahelduv bitijada "10101". Neljanda testi tulemusena peavad registrites olema järgmised 16-bitised andmed:

	Ülemine 16-bitine osa	Alumine 16-bitine osa
R0:	002A, 0029,….	0000
R1:	102A, 1029,….	1000
..		

R15: F02A, F029,…. F000

Kõikide testide korral genereeritakse sisendandmed liidese juhtprogrammi siseselt. Liidese juhtprogramm on LISA B.

7.3 KIIBI TESTPROGRAMM

on kiibi käsukeeles kirjutatud programm, mis kontrollib sisemise aritmeetikaautomaadi korrasolekut.

Kiibi aritmeetikasõlme testimiseks peab kontrollima kõiki kiibi käske (p. 4.5):

nimetus	käsu number tabelis 2
1) siirdekäsud	9-16
2) andmeteisalduskäsud	17-22
3) liitmine, lahutamine, korrutamine	0, 5-8
4) jagamine, moodul, mooduliga korrutamine	2-4
5) diskreetne eksponent	1
6) muud käsud	23-27

Testprogramm käivitatakse läbi nullinda väliskäsu. Pärast iga etappi peatatakse programm sisemise `WAIT` käsuga. DEBUG reziimis loetakse registermälu sisu kiibist välja. Testi tulemusfailis saab testi positiivse resultaadi üle otsustada registri A0L väärtuse järgi, mis etapi positiivse tulemuse korral on 0. Samuti peab tsükliloendur CX olekuregistris Sr olema ületäitunud. Testprogramm algab registrite nullimisega ning kasutades aritmeetilisi samasusi kontrollib aritmeetikaautomaadi juhtosa ning ALU tööd kolme erineva testi abil.

Esimene test pärast mälu nullimist kontrollib liitmist ning korrutamist kasutades järgmist samasust:

$$(a+b)(a-b) = a^2 - b^2$$

Kiibi ALU on 8 argumendiga CSA kompressor ning tema väljundis CLA liitja. Valime juhuslikult argumendid a ning b kasutades kiibi sisemistt juhuarvugeneraatorit ning sooritame testi kümme korda.

Teiseks kontrollitakse jagamise/korrutamise/mooduliga korrutamise paari kasutades järgmist samasust:

$$(a * b) \% Z = a*b \bmod Z$$

% tähistab mooduli arvutamist, ning a*b mod Z on mooduliga korrutis. Kiibi mikrokoodis realiseerivad neid tehteid erinevad mikroprogrammid, kuna mooduliga korrutamine peab olema kiirem mooduli arvutamisest. Eelneva testiga on korruti

kontrollitud ning on üle jäänud testida mikrokoodi AR_BSEARCH käsud ja nende käskude poolt kasutatav 7-bitine korruti. Testi tuleb sooritada juhuslike argumenditega testi kümme korda. Juhuslike argumentide valik on siinkohal õigustatud, kuna kiibi sees olev juhuslike arvude generaator annab pärast iga käivitamist sama tulemuse, kui välist juhuarvugeneraatorit ei ole kiibi külge ühendatud. Testida tuleb erinevate lähteandmetega kümme korda.
Viimane test kontrollib modulaareksponenti. See on mikrokoodi kontroll, kuna vajaminevad tehted (DMUL MOD) on siia etappi jõudes eelnevate testide poolt kontrollitud. Eksponendi test kasutab samasust:

$$a^{(e-1)} \bmod e = 1,$$

kus e on algarv. Algarvuks tuleb valida selle testi jaoks F4 ning testi peab kordama juhusliku aluse a korral kümme korda.
Kiibi testprogramm on LISA A.

7.4 TESTIMISE APARATUUR

Testimine teostatakse PC tüüpi arvutil, millele oli lisatud SVM välismoodul koos testitava kiibiga. Voolu ja pinget mõõdetakse numbrilise multimeetriga. Kasutatavate seadmete parameetrid peavad olema järgmised:

7.4.1 Arvuti konfiguratsioon

nimetus	tingimus	väärtus
Protsessor:	vähemalt	i80486
Töösagedus:	vähemalt	66 MHz
Operatiivmälu:	vähemalt	16 MB
Välismälu:	vähemalt	1 MB vaba
OP süsteem:		WIN95 Euroopa versioon

7.4.2 Testliides

SVM ISA siini välismoodul koos testitava kiibiga (p. 7.1, [SVMDOK], LISA M)

7.4.3 Numbriline multimeeter

Pinge mõõtmise piirkond:	10V	± 10 mV
Voolu mõõtmise piirkond:	100mA	± 10 µA

7.4.4 Meandersignaali generaator

Minimaalne sagedus	1± 0.01	MHz
Maximaalne sagedus	50 ± 0.01	MHz
Minimaalne väljundpinge	1± 0.01	V

Maximaalne väljundpinge	10± 0.01	V
Sageduse muutmise aste	1± 0.01	MHz
väljundpinge muutmine	Sujuvalt	

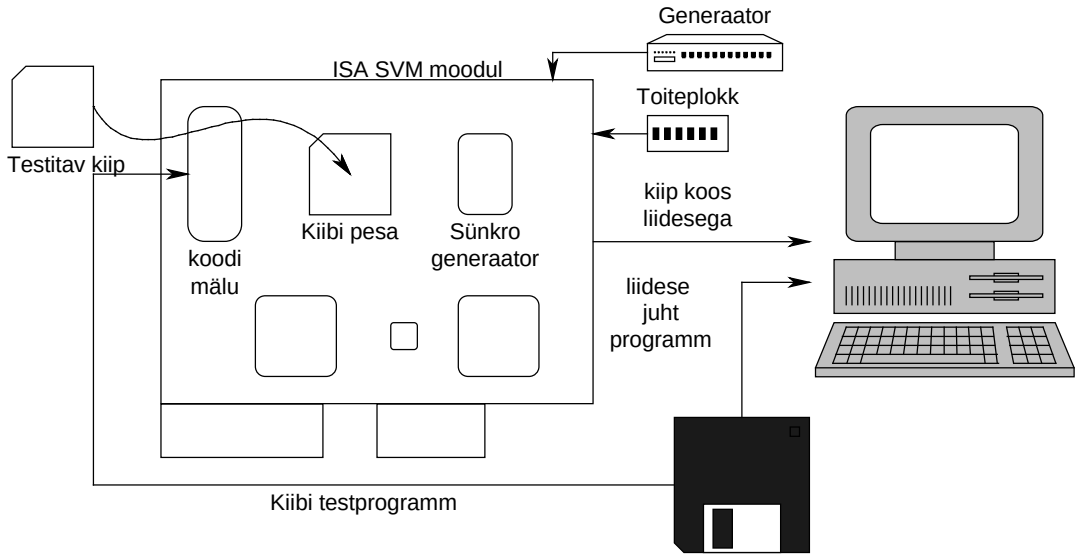
7.4.5 Toiteplokk

Minimaalne väljundpinge	2	V
Maximaalne väljundpinge	10	V
Maximaalne väljundvool vähemalt	1	A
Pinge muutmise aste	0.5	V
Pinge maksimaalne pulsatsioon	± 0.1	V (tipp-tipp)

8. KATSEMEETODID JA HINDAMISE KRITEERIUMID

8.1 KATSESKEEM

Katseskeem koosneb arvutist, mille ISA siinile on lisatud SVM ISA moodul (p. 7.1, [SVMDOK], Lisa M). Mooduli kiibi pesas peab olema testitav kiip. Arvutile peab olema instaleeritud liidese juhtprogramm.



Joonis 14. Kiibi katseskeem

8.2 LIIDESE JUHTPROGRAMMI SISENDID

Testi läbiviimiseks on vaja järgmiseid sisend faile:

A. Jooksvas kataloogis

1) Ctest.exe	Testimise käivitatav programm
2) regs.bin	Registrite sisu pärast kiibi lähtestamist
3) low_deb.img	Mikrokood
4) lo_nop.bin	Mikrokoodi käskude aadressid
5) hi_deb.img	Kiibi testprogramm
6) hi_nop.bin	Kiibi testprogrammi käskude aadressid
7) dkeys.bin	Idea võtmed enc/dec testi jaoks.

B. kataloogis .\KONTROLL

Integreeritud testi kontrollfailid testimisetappide tulemuste hindamiseks (LISA D - LISA L)
Programm Ctest.exe on kompileeritud (ANSI C kompilaator) liidese juhtprogramm (LISA B). Failid regs.bin kuni dkeys.bin genereerib silur [SILUR] kiibi integreeritud

testprogrammist (LISA A). Siluri ASCII kujul väljundfailid hi_deb.img ning lo_deb.img konverteeritakse kahendkujule (*hi_deb.bin, lo_deb.bin*) kasutades programmi kwi2bin. Kwi2bin on järgmise formaadiga

```
kwi2bin sisendfail väljundfail
```

Sisendfail on listingufail siluri väljundformaadis. Väljund salvestatakse *väljundfail* nimega. *Väljundfail* ärajätmisel on väljundi nimi sama mis sisendil, laiendiga bin. Näiteks kwi2bin hi_deb.img kirjutab väljundi faili hi_deb.bin. Kwi2bin on kompileeritud 32 bitise kompilaatoriga BC5.0, mistõttu teda saab kasutada ainult WIN95 keskkonnas.

8.3 LIIDESE JUHTPROGRAMMI KÄIVITAMINE NING VÄLJUNDID

Testimiseks vajalike sisendfailide olemasolul kataloogis tuleb testimise läbiviimiseks käivitada programm CTEST.exe. Programmi töö ajaline kestvus IBM-PC/66 Mhz taktsageduse korral on mitte üle 10e sekundi. Programmi pikemal töötamisel tuleb ta katkestada, kasutades selleks WIN95e süsteemseid vahendeid ning uuesti käivitada. Kui korduvkäivitus ei andnud tulemusi, siis on programmis viga. Kopeerige programmi väljund tekstifaili ning saatke nad koos programmi töökataloogi sisuga projekti täitjatele. Programmi töö normaalsel kulgemisel antakse kontroll testijale tagasi vähema kui kümne sekundi jooksul alates programmi käivitamisest.
Kui mingil põhjusel testimise käigus tuleb testimine katkestada, siis testimise uuesti käivitamiseks on vaja programm CTEST uuesti käivitada. Programm kirjutab tulemuse failid kataloogis RESULTAAT üle.

8.4 LIIDESE JUHTPROGRAMMI TEATED

Programmi CTEST

veateaded ning vea likvideerimise võimalused on järgmised:

```
Faili aaaa.aaa ei saa avada.
```

Kontrollige faili aaaa.aaa olemasolu kettal või väljndfaili korral küllaldase vaba kettaruumi olemasolu.

```
Liidese registrite sv viga
```

LVM ISA mooduli sv viga. Kontrollige mooduli paigutust ISA siini peal. Juhul, kui see ei aita, siis võtke ühendust projekti läbiviijatega.

```
Liidese mälu viga
```

Liidese koodimälu kirjutamis-lugemistest negatiivne. Võtke ühendust projekti täitjatega.

IDEA algoritmi testimise väljundteated

on järgmised:

```
NNNNN IDEA random tests OK.
```

NNNNN IDEA testi sooritatud. Testitakse IDEA krüpteerimis - dekrüpteerimis paari.

```
IDEA test NNNNN failed
```

IDEA test nr. NNNNN on negatiivne. Viitab rikkele kiibi IDEA arvutuses.

Programmi kwi2bin
veateated on järgmised:

```
faili aaaaa.aaa ei saa avada
```

Kontrollida sisendfaili olemasolu ning väljundfaili korral vaba kettaruumi olemasolu.

```
Viga listingufailis real NNNN
```

Sisendfaili rida NNNN ei ole siluri formaadis listing. Kontrollige sisendfaili süntaksit real NNNN. Sisendfaili formaat peab vastama programmi SYNOPSYS simulaatori mälufaili formaadile.

8.5 TULEMUSTE REGISTREERIMISE VIIS JA HINDAMISE KRITEERIUMID

Aritmeetika- ning mälutesti tulemused kirjutatakse liidese juhtprogrammi täitmise korral kataloogi .\RESULTAAT järgnevalt:

1) RAM_C0.dat	Mälu konstant 0 testi väljundfail
2) RAM_C1.dat	Mälu konstant 1 testi väljundfail
3) RAM_CHKB.dat	Mälu malelaua testi väljundfail
4) RAM_INDX.dat	Mälu indekseeritud sisendjada testi väljundfail
5) WAIT.000	MODEX testi sisendetapp- registrite nullimine
6) WAIT.001	MODEX korrutamise testi väljundfail
7) WAIT.002	MODEX modulaarkorrutamise testi väljundfail
8) WAIT.003	MODEX modulaareksponendi testi väljundfail

Neid faile tuleb pärast integreeritud testi sooritamist võrrelda .\KONTROLL kataloogis olevate failidega, mis on ara toodud ka lisas (LISA D - LISA L). Võrdlemiseks tuleb kasutada WIN95 süsteemseid vahendeid või teostada võrdlus visuaalselt.

8.5.1 IDEA test

IDEA testimise tulemused ning teated testimisprotseduuride kulgemise kohta väljastab programm tekstikujul ekraanile, vastavalt p. 8.4 kirjeldatud formaadile. Programmi väljundit terminalile saab vajaduse korral kopeerida pärast testi lõppu tekstifaili WIN95 süsteemsete vahenditega. Juhtprogrammi ekraaniväljund integreeritud testi positiivse tulemuse korral on LISA C.

8.5.2 Mälu test

testi nimetus	Lisa	tulemused
1) Konstant 0	D	RAM_C0.dat
2) Konstant 1	E	RAM_C1.dat
3) Malelaud	F	RAM_CHKB.dat
4) Indekseeritud sisendjada	G	RAM_INDX.dat

Testi tulemuste kontroll viiakse läbi tulemusfailide võrdluse abil. Esimese testi tulemusena peab kogu mälu sisu olema 1 ehk FFFF. Teise testi tulemusena peab mälu sisu olema 0. Kolmanda testi tulemusena 10101 ehk AAAAA. Neljanda testi tulemusena peavad registrites olema järgmised 16-bitised andmed:

	Ülemine 16-bitine osa	Alumine 16-bitine osa
R0:	002A, 0029,….	0000
R1:	102A, 1029,….	1000
..		
R15:	F02A, F029,….	F000

Kõikide testide korral genereeritakse sisendandmed programmi siseselt.

8.5.3 Modulaararitmeetika test

nr. testi nimetus	lisa	Tulemused
1) Mälu nullimine	H	WAIT.000
2) Korrutamine, 10 korda	J	WAIT.001
3) Mooduliga korrutamine, 10 korda	K	WAIT.002
4) Mooduliga eksponent, 10 korda	L	WAIT.003

Testi tulemused hinnatakse väljundfailide võrdlusmeetodil. Esimese testi tulemusfailis peavad olema kõik registrid nullitud. Teise kuni neljanda testi tulemusfailis peab olema register A0L 0 ja staatusregistris CX FFFF. Testid algavad mälu nullimisega ning kasutavad kiibi sisemist juhuarvugeneraatorit.

8.5.4 Integreeritud test

Kiip loetakse korrasolevaks, kui ta läbib positiivselt kõik eelpoolnimetatud testid (8.5.1-8.5.3).

LISA A Kiibi testprogramm

```
;--------------------------------
; Väliskäskude tabel.
; Nullis väliskäsk käivitab testi
;--------------------------------
        JMP     H_test              ; 0
        JMP     H_unknown_cmd_1     ; 1
        JMP     H_unknown_cmd_2     ; 2
        JMP     H_unknown_cmd_3     ; 3
        JMP     H_unknown_cmd_4     ; 4
        JMP     H_unknown_cmd_5     ; 5
        JMP     H_unknown_cmd_6     ; 6
        JMP     H_unknown_cmd_7     ; 7
        JMP     H_unknown_cmd_8     ; 8
        JMP     H_unknown_cmd_9     ; 9
        JMP     H_unknown_cmd_10    ; 10
        JMP     H_unknown_cmd_11    ; 11
        JMP     H_unknown_cmd_12    ; 12
        JMP     H_unknown_cmd_13    ; 13
        JMP     H_unknown_cmd_14    ; 14
        JMP     H_unknown_cmd_15    ; 15
        JMP     H_unknown_cmd_16    ; 16
        JMP     H_unknown_cmd_17    ; 17
        JMP     H_unknown_cmd_18    ; 18
        JMP     H_unknown_cmd_19    ; 19
        JMP     H_unknown_cmd_20    ; 20
        JMP     H_unknown_cmd_21    ; 21
        JMP     H_unknown_cmd_22    ; 22
        JMP     H_unknown_cmd_23    ; 23
        JMP     H_unknown_cmd_24    ; 24
        JMP     H_unknown_cmd_25    ; 25
        JMP     H_unknown_cmd_26    ; 26
        JMP     H_unknown_cmd_27    ; 27
        JMP     H_unknown_cmd_28    ; 28
        JMP     H_unknown_cmd_29    ; 29
        JMP     H_unknown_cmd_30    ; 30
        JMP     H_unknown_cmd_31    ; 31
```

```
H_test:

         CALL     clear_regs              ; puhasta mälu

         WAIT
         CALL     TEST_FM_MUL             ; korrutamise test
                                          ; etapp I
         WAIT
         CALL     TEST_MUL_MOD_DMUL       ; mooduliga korrutamis
                                          ; etapp II
         WAIT
         CALL     TEST_DEX_F4             ; Diskreetne eksponent
                                          ; etapp III
         WAIT
         JMP      H_Done_ok

;----------------------------------------
;  testides kasutatavad registrid:
; a = A0L
; b = A4L
; c = A2L
; d = A3L
; e = A4H
; f = A5L
; g = A0H
; h = A2H
;----------------------------------------
; ETAPP I, korrutamise test
;----------------------------------------
TEST_FM_MUL:
        LCX       10                     ; testime 10 korda
DO_AGTB:
        CALL      FILL_RND_abc           ;
        JLT       A4L, A0L, TEST_FM_AGTB ; kui a>b siis
            ; edasi, muidu
            ; vaheta. Korruti töötab ainult positiivsete
            ; arvudega. See on nii ette nähtud, tegemist
            ; on modulaararitmeetika arvutiga

        MOV       A2L, A4L               ; c := b
        MOV       A4L, A0L               ; b := a
        MOV       A0L, A2L               ; a := c := b
TEST_FM_AGTB:                            ;
        SUB       A0L, A4L, A2L          ; c := a - b
        ADD       A0L, A4L, A3L          ; d := a + b
        MUL       A3L, A2L, A3L          ; d := d * c
        MUL       A0L, A0L, A0L          ; a := a^2
        MUL       A4L, A4L, A4L          ; b := b^2
        SUB       A0L, A4L, A0L          ; a := a - b
        SUB       A0L, A3L, A5L          ; f := a - d
        JNZ       A5L, FM_ERR_AGTB       ; vale, lahkume
        LOOP      DO_AGTB
        JMP       TEST_PASSED
```

```
;--------------------------------------
; ETAPP II, mooduliga korrutamise test
;--------------------------------------
TEST_MUL_MOD_DMUL:
        LCX       10
DO_MUL_MOD_DMUL:
        CALL      FILL_RND_abc
        MUL       A0L, A4L, A4H           ; e := a * b
        MOD       A4H, A4H, A2L           ; e := e % c
        DMUL      A4L, A0L, A2L           ; b := b * a % c
        SUB       A4L, A4H, A3L           ; b := b - e
        JNZ       A3L, FM_ERR_MUL_MOD_DMUL
        LOOP      DO_MUL_MOD_DMUL
        JMP  TEST_PASSED

;--------------------------------------
; ETAPP III, diskreetne eksponent
;--------------------------------------

TEST_DEX_F4:
        LCX       10
DO_DEX_F4:
        MVI       A0L, 0                  ;
        RND       A0L                     ; a := rnd
        MF4       A2L                     ; c := F4
        MVI       A5L, 1                  ;
        SUB       A2L, A5L, A4L           ; b := c - 1
        DEX       A0L, A4L, A2L           ; a := a^b % c
        SUB       A0L, A5L, A0L           ; a := a - 1
        JNZ       A0L, FM_ERR_DEX_F4
        LOOP      DO_DEX_F4
        JMP       TEST_PASSED

;======================================
; vastusekoodid
;======================================

TEST_PASSED:      MVI     A0L, 0
                  RET
FM_ERR_AGTB:
                  MVI     A0L, 1
                  RET
FM_ERR_MUL_MOD_DMUL:
                  MVI     A0L, 2
                  RET
FM_ERR_DEX_F4:
                  MVI     A0L, 3
                  RET
```

```
; väljundkoodid

H_Done_ok:         RTI     0       ; valmis ok
H_unknown_cmd_1:   RTI     1       ; unknown command
H_unknown_cmd_2:   RTI     2       ; unknown command
H_unknown_cmd_3:   RTI     3       ; unknown command
H_unknown_cmd_4:   RTI     4       ; unknown command
H_unknown_cmd_5:   RTI     5       ; unknown command
H_unknown_cmd_6:   RTI     6       ; unknown command
H_unknown_cmd_7:   RTI     7       ; unknown command
H_unknown_cmd_8:   RTI     8       ; unknown command
H_unknown_cmd_9:   RTI     9       ; unknown command
H_unknown_cmd_10:  RTI     10      ; unknown command
H_unknown_cmd_11:  RTI     11      ; unknown command
H_unknown_cmd_12:  RTI     12      ; unknown command
H_unknown_cmd_13:  RTI     13      ; unknown command
H_unknown_cmd_14:  RTI     14      ; unknown command
H_unknown_cmd_15:  RTI     15      ; unknown command
H_unknown_cmd_16:  RTI     16      ; unknown command
H_unknown_cmd_17:  RTI     17      ; unknown command
H_unknown_cmd_18:  RTI     18      ; unknown command
H_unknown_cmd_19:  RTI     19      ; unknown command
H_unknown_cmd_20:  RTI     20      ; unknown command
H_unknown_cmd_21:  RTI     21      ; unknown command
H_unknown_cmd_22:  RTI     22      ; unknown command
H_unknown_cmd_23:  RTI     23      ; unknown command
H_unknown_cmd_24:  RTI     24      ; unknown command
H_unknown_cmd_25:  RTI     25      ; unknown command
H_unknown_cmd_26:  RTI     26      ; unknown command
H_unknown_cmd_27:  RTI     27      ; unknown command
H_unknown_cmd_28:  RTI     28      ; unknown command
H_unknown_cmd_29:  RTI     29      ; unknown command
H_unknown_cmd_30:  RTI     30      ; unknown command
H_unknown_cmd_31:  RTI     31      ; unknown command
H_unknown_cmd_1:   RTI     32      ; unknown command
```

```
;================================================
; abiprotseduurid
;================================================
;------------------------------------------------
; täida pseudojuhusliku generaatori abil
; registrid       A0L, A4L ja A2L.
; nulli registrid A0H, A4H ja A2H
;------------------------------------------------

FILL_RNDA012:
        MVI       A0L, 0
        MVI       A4L, 0
        MVI       A2L, 0
        RND       A0L
        RND       A4L
        RND       A2L
        MVI       A2H, 0
        MVI       A4H, 0
        MVI       A0H, 0
        RET

;------------------------------------------------
; Nulli sisemised registrid
;------------------------------------------------
clear_regs:
        MVI       A0,  0
        MVI       A1,  0
        MVI       A2,  0
        MVI       A3,  0
        MVI       A4,  0
        MVI       A5L, 0
        MVI       U,   0
        MVI       PQ,  0
        MVI       PQu, 0
        MVI       Pu,  0
        MVI       Qu,  0
        MVI       Ul,  0
        MVI       I0,  0
        MVI       I1,  0
        MVI       Sr,  0
        MVI       T1,  0
        MVI       Tr,  0
        MVI       N,   0
        MVI       D,   0
        MVI       M,   0
        RET
```

LISA B Siluri isa välismooduli (SVM) juhtprogramm

```
#include "lvmc_lib.h"

#define WAIT_COUNT 4

void init_debug()
{
   int i;
   for(i=0; i<(WORDS_IN_REG*16); i++)
             CC.REGS[i]=0;          /* nulli m"alu */

   clrscr();                        /* puhasta ekraan */
   checkregs();                     /* kas ikka on liides olemas ? */
   printf("Liidese registrid ok\n");

   read_binfile ("regs.bin", (char *) CC.REGS, &i );
      /* registrid   */

   read_binfile ("low_deb.bin", DINF.lo_code, &DINF.lo_len);
      /* mikrokood  */

   read_binfile("lo_nop.bin", (char *) DINF.lo_nop_places,
                &DINF.lo_cmd_cnt );
      /* mikrokoodi nopi kohad */

   read_binfile ("hi_deb.bin",  DINF.hi_code, &DINF.hi_len);
      /* asssembler     */

   read_binfile ("hi_nop.bin",  (char *) DINF.hi_nop_places,
                &DINF.hi_cmd_cnt );
      /* assembleri nopide kohad */

   DINF.hi_cmd_cnt>>=1;
   DINF.lo_cmd_cnt>>=1;

   /* molemad on karakteri pointerid
      tegelikult taheme integeri omasid,
      iga integer on 2 karakterit */
   /* laeme koodi liidese koodim"allu */

      load_code(DINF.hi_code, DINF.lo_code);
   /* lisame jupikese koodim"allu, mis hakkab debugi iod tegema */
      add_lo_debug();
   printf("\n Debug initialized\n");

}

void test_etapp_1()
{
int i, j;
   mode_io();
   for(i=0; i<WORDS_IN_REG*REGS_IN_CCHIP; i++) CC.TREGS[i]=0;
   write_regs(CC.TREGS);
   mode_io();
   read_regs( CC.REGS, CC.LREG);     /* konstant 0 test */
   printf("%-13s: M„lu test, konstant 0\n", "RAM_C0.dat");

   save_int_regs("RAM_C0.dat", &CC );
   for(i=0; i<WORDS_IN_REG*REGS_IN_CCHIP; i++) CC.TREGS[i]=0xffff;
   mode_io();
   write_regs(CC.TREGS);
   mode_io();
   read_regs( CC.REGS, CC.LREG);     /* konstant 1 test */
   save_int_regs("RAM_C1.dat", &CC );
   printf("%-13s: M„lu test, konstant 1\n", "RAM_C1.dat");
```

```
   for(i=0; i<WORDS_IN_REG*REGS_IN_CCHIP; i++) CC.TREGS[i]=0xAAAA;
   mode_io();
   write_regs(CC.TREGS);
   mode_io();
   read_regs( CC.REGS, CC.LREG);     /* malelaua test */
   save_int_regs("RAM_CHKB.dat", &CC  );
   printf("%-13s: M„lu test, malelaud \n", "RAM_CHKB.dat");

   for(i=0; i<REGS_IN_CCHIP; i++)
      for(j=0;j<WORDS_IN_REG; j++)
        CC.TREGS[i*WORDS_IN_REG+j]=(i<<12)|j;
   mode_io();
   write_regs(CC.TREGS);
   mode_io();
   read_regs( CC.REGS, CC.LREG);     /* indeksi test */
   save_int_regs("RAM_INDX.dat", &CC );
   printf("%-13s:   M„lu   test,   indekseeritud   sisendjada   \n",
"RAM_INDX.dat");
}

/* testimise etapp 2, k"askude test */

void test_etapp_2()
{
   int i;
   char nimeke[30];
   mode_commands();
   pulse_do();
   waitfor(WAIT, 1);
   for(i=0; i<WAIT_COUNT; i++)
      {
      waitfor(WAIT, 1);
      read_regs(CC.REGS, CC.LREG);
/*    loe registrid CCst v"alja          */
      sprintf(nimeke, "WAIT.%-03d", i );
      printf("%-13s: k„skude test \n", nimeke ,i );
/*    tekitame failinime                 */
      save_int_regs( nimeke, &CC);
/*    kirjuta resultaadid v"alja         */
      pulse_run();
      /* kuni j"argmise runini */
      }
}

void test_etapp_3()
{
   int i;
   for(i=0; i<WORDS_IN_REG*REGS_IN_CCHIP; i++) CC.TREGS[i]=0;
   mode_io();
/* io reziim, nullime m"alu       */
   write_regs(CC.TREGS);
/* kirjuta nullid m"allu          */
   enter_idea();
/* sisene IDEA reziimi            */
   printf("IDEA involutsioonitest\n");
   idea_encdec_test(1000);
/* idea involutsioonitest         */
   read_binfile ("idkeys.bin",  (char *)CC.TREGS, &i );
/* loeme idea key sisse           */
   printf("IDEA enc/dec test\n");
   idea_encdec_test(1000);
/* votmega test                   */
}

void run_tests_1to3()
{
   waitfor(MAINRDY, 1);     /* kuni initiga valmis saab       */
   test_etapp_1();          /* Malutest, etapp  I             */
   setcom(0x0);             /* nullis k"ask, testprotseduurid */
   test_etapp_2();          /* k"asutest, etapp II            */
```

```
   test_etapp_3();          /* IDEA test, etapp III           */
}

void main()
{
   int i ,j;
   char nimeke[40], a;
   int com_cnt;
   DEBUG = 0;
   init_debug();            /* loeb koodi liidese koodim"allu  */
   setmhz(10);              /* kella sagedus                   */
   connect_chip();          /* "uhenda kiip k"ulge             */
   SETBIT(CLKSRC);          /* valime generaatori              */
   run_tests_1to3();        /* testi etapid  1 kuni 3          */
}
```

LISA C Testprogrammi ekraaniväljund

```
D:\CHPTST>ctest
Liidese registrid ok

 Debug initialized
soovitud sagedus:  16 MHz, tegelik:  16 MHz
RAM_C0.dat   : Mälu test, konstant 0
RAM_C1.dat   : Mälu test, konstant 1
RAM_CHKB.dat : Mälu test, malelaud
RAM_INDX.dat : Mälu test, indekseeritud sisendjada
WAIT.0       : käskude test
WAIT.1       : käskude test
WAIT.2       : käskude test
WAIT.3       : käskude test
IDEA involutsioonitest
IDEA ZERO DATA: IDEA(2) i=o, 00
IDEA ZERO DATA: IDEA(3) i=o, 00
01000 random IDEA tests OK
IDEA enc/dec test
IDEA ZERO DATA: IDEA(2) i=o, 00
IDEA ZERO DATA: IDEA(3) i=o, 00
01000 random IDEA tests OK

D:\CHPTST>
```

LISA D Konstant 0 testi tulemused: RAM_C0.DAT

```
00: Tr     -
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;
01: T1     -
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;
02: I0     -
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;
03: I1     -
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;
04: Sr     -
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;
05: N      -
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;
06: M      -
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;
07: D      -
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;
08: PQ    P/Q
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;
09: PQu   Pu/Qu
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;
10: Ul    Uu/A5l
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;
11: A0    A0L/A0H
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;
12: A1    A1L/A1H
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;
13: A2    A2L/A2H
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;
14: A3    A3L/A3H
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;
15: A4    A4L/A4H
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;
```

LISA E Konstant 1 testi tulemused: RAM_C1.DAT

```
00: Tr     -
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;      FFFFFFFFFFFFFFFFFFFFFFFF;
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;      FFFFFFFFFFFFFFFFFFFFFFFF;
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;
01: T1     -
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;      FFFFFFFFFFFFFFFFFFFFFFFF;
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;      FFFFFFFFFFFFFFFFFFFFFFFF;
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;
02: I0     -
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;      FFFFFFFFFFFFFFFFFFFFFFFF;
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;      FFFFFFFFFFFFFFFFFFFFFFFF;
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;
03: I1     -
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;      FFFFFFFFFFFFFFFFFFFFFFFF;
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;      FFFFFFFFFFFFFFFFFFFFFFFF;
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;
04: Sr     -
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;      FFFFFFFFFFFFFFFFFFFFFFFF;
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;      FFFFFFFFFFFFFFFFFFFFFFFF;
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;
05: N      -
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;      FFFFFFFFFFFFFFFFFFFFFFFF;
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;      FFFFFFFFFFFFFFFFFFFFFFFF;
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;
06: M      -
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;      FFFFFFFFFFFFFFFFFFFFFFFF;
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;      FFFFFFFFFFFFFFFFFFFFFFFF;
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;
07: D      -
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;      FFFFFFFFFFFFFFFFFFFFFFFF;
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;      FFFFFFFFFFFFFFFFFFFFFFFF;
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;
08: PQ    P/Q
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;      FFFFFFFFFFFFFFFFFFFFFFFF;
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;      FFFFFFFFFFFFFFFFFFFFFFFF;
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;
09: PQu  Pu/Qu
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;      FFFFFFFFFFFFFFFFFFFFFFFF;
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;      FFFFFFFFFFFFFFFFFFFFFFFF;
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;
10: Ul   Uu/A5l
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;      FFFFFFFFFFFFFFFFFFFFFFFF;
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;      FFFFFFFFFFFFFFFFFFFFFFFF;
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;
11: A0   A0L/A0H
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;      FFFFFFFFFFFFFFFFFFFFFFFF;
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;      FFFFFFFFFFFFFFFFFFFFFFFF;
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;
12: A1   A1L/A1H
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;      FFFFFFFFFFFFFFFFFFFFFFFF;
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;      FFFFFFFFFFFFFFFFFFFFFFFF;
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;
13: A2   A2L/A2H
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;      FFFFFFFFFFFFFFFFFFFFFFFF;
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;      FFFFFFFFFFFFFFFFFFFFFFFF;
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;
14: A3   A3L/A3H
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;      FFFFFFFFFFFFFFFFFFFFFFFF;
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;      FFFFFFFFFFFFFFFFFFFFFFFF;
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;
15: A4   A4L/A4H
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;      FFFFFFFFFFFFFFFFFFFFFFFF;
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;      FFFFFFFFFFFFFFFFFFFFFFFF;
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;
```

LISA F Malelaua testi tulemused: RAM_CHKB.DAT

```
00: Tr     -
AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;
AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;
AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;
01: T1     -
AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;
AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;
AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;
02: I0     -
AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;
AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;
AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;
03: I1     -
AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;
AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;
AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;
04: Sr     -
AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;
AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;
AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;
05: N      -
AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;
AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;
AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;
06: M      -
AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;
AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;
AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;
07: D      -
AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;
AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;
AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;
08: PQ     P/Q
AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;
AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;
AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;
09: PQu   Pu/Qu
AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;
AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;
AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;
10: Ul    Uu/A5l
AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;
AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;
AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;
11: A0    A0L/A0H
AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;
AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;
AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;
12: A1    A1L/A1H
AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;
AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;
AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;
13: A2    A2L/A2H
AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;
AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;
AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;
14: A3    A3L/A3H
AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;
AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;
AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;
15: A4    A4L/A4H
AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;
AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;
AAAAAAAAAAAAAAAAAAAAAAAA;     AAAAAAAAAAAAAAAAAAAAAAAA;
```

LISA G Indekseeritud sisendjada testi tulemused: RAM_INDX.DAT

```
00: Tr     -
0005000400030002000100000;     000B000A0009000800070006;      00110010000F000E000D000C;
0017001600150014001300120;     001D001C001B001A00190018;      0023002200210020001F001E;
0029002800270026002500240;     002F002E002D002C002B002A;
01: T1     -
1005100410031002100110000;     100B100A1009100810071006;      10111010100F100E100D100C;
1017101610151014101310120;     101D101C101B101A10191018;      1023102210211020101F101E;
1029102810271026102510240;     102F102E102D102C102B102A;
02: I0     -
2005200420032002200120000;     200B200A2009200820072006;      20112010200F200E200D200C;
2017201620152014201320120;     201D201C201B201A20192018;      2023202220212020201F201E;
2029202820272026202520240;     202F202E202D202C202B202A;
03: I1     -
3005300430033002300130000;     300B300A3009300830073006;      30113010300F300E300D300C;
3017301630153014301330120;     301D301C301B301A30193018;      3023302230213020301F301E;
3029302830273026302530240;     302F302E302D302C302B302A;
04: Sr     -
4005400440034002400140000;     400B400A4009400840074006;      40114010400F400E400D400C;
4017401640154014401340120;     401D401C401B401A40194018;      4023402240214020401F401E;
4029402840274026402540240;     402F402E402D402C402B402A;
05: N      -
5005500450035002500150000;     500B500A5009500850075006;      50115010500F500E500D500C;
5017501650155014501350120;     501D501C501B501A50195018;      5023502250215020501F501E;
5029502850275026502550240;     502F502E502D502C502B502A;
06: M      -
6005600460036002600160000;     600B600A6009600860076006;      60116010600F600E600D600C;
6017601660156014601360120;     601D601C601B601A60196018;      6023602260216020601F601E;
6029602860276026602560240;     602F602E602D602C602B602A;
07: D      -
7005700470037002700170000;     700B700A7009700870077006;      70117010700F700E700D700C;
7017701670157014701370120;     701D701C701B701A70197018;      7023702270217020701F701E;
7029702870277026702570240;     702F702E702D702C702B702A;
08: PQ    P/Q
8005800480038002800180000;     800B800A8009800880078006;      80118010800F800E800D800C;
8017801680158014801380120;     801D801C801B801A80198018;      8023802280218020801F801E;
8029802880278026802580240;     802F802E802D802C802B802A;
09: PQu   Pu/Qu
9005900490039002900190000;     900B900A9009900890079006;      90119010900F900E900D900C;
9017901690159014901390120;     901D901C901B901A90199018;      9023902290219020901F901E;
9029902890279026902590240;     902F902E902D902C902B902A;
10: Ul    Uu/A5l
A005A004A003A002A001A000;      A00BA00AA009A008A007A006;      A011A010A00FA00EA00DA00C;
A017A016A015A014A013A012;      A01DA01CA01BA01AA019A018;      A023A022A021A020A01FA01E;
A029A028A027A026A025A024;      A02FA02EA02DA02CA02BA02A;
11: A0    A0L/A0H
B005B004B003B002B001B000;      B00BB00AB009B008B007B006;      B011B010B00FB00EB00DB00C;
B017B016B015B014B013B012;      B01DB01CB01BB01AB019B018;      B023B022B021B020B01FB01E;
B029B028B027B026B025B024;      B02FB02EB02DB02CB02BB02A;
12: A1    A1L/A1H
C005C004C003C002C001C000;      C00BC00AC009C008C007C006;      C011C010C00FC00EC00DC00C;
C017C016C015C014C013C012;      C01DC01CC01BC01AC019C018;      C023C022C021C020C01FC01E;
C029C028C027C026C025C024;      C02FC02EC02DC02CC02BC02A;
13: A2    A2L/A2H
D005D004D003D002D001D000;      D00BD00AD009D008D007D006;      D011D010D00FD00ED00DD00C;
D017D016D015D014D013D012;      D01DD01CD01BD01AD019D018;      D023D022D021D020D01FD01E;
D029D028D027D026D025D024;      D02FD02ED02DD02CD02BD02A;
14: A3    A3L/A3H
E005E004E003E002E001E000;      E00BE00AE009E008E007E006;      E011E010E00FE00EE00DE00C;
E017E016E015E014E013E012;      E01DE01CE01BE01AE019E018;      E023E022E021E020E01FE01E;
E029E028E027E026E025E024;      E02FE02EE02DE02CE02BE02A;
15: A4    A4L/A4H
F005F004F003F002F001F000;      F00BF00AF009F008F007F006;      F011F010F00FF00EF00DF00C;
F017F016F015F014F013F012;      F01DF01CF01BF01AF019F018;      F023F022F021F020F01FF01E;
F029F028F027F026F025F024;      F02FF02EF02DF02CF02BF02A;
```

LISA H Mälu seis enne modulaararitmeetika teste: WAIT.000

```
00: Tr     -
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;
01: T1     -
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;
02: I0     -
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;
03: I1     -
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;
04: Sr     -
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;
05: N      -
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;
06: M      -
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;
07: D      -
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;
08: PQ    P/Q
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;
09: PQu  Pu/Qu
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;
10: Ul   Uu/A5l
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;
11: A0   A0L/A0H
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;
12: A1   A1L/A1H
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;
13: A2   A2L/A2H
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;
14: A3   A3L/A3H
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;
15: A4   A4L/A4H
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;     00000000000000000000000000;
00000000000000000000000000;     00000000000000000000000000;
```

LISA J Liitmise ja korrutamise testi tulemused: WAIT.001

```
00: Tr     -
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;
FFFFFFFFFFFFFFFFFFFFFFFF;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;
01: T1     -
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;
02: I0     -
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;
03: I1     -
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;
04: Sr     -
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
0000FFFF0000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;
05: N      -
612EA7ACB73B9AB68DD07178;     FFFFFF7C870DBCB4E63396D1;     FFFFFFFFFFFFFFFFFFFFFFFF;
FFFFFFFFFFFFFFFFFFFFFFFF;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;
06: M      -
FAC88B97951A2B715235E7B0;     68401658FE65F68E81703C23;     02894ACD73D7322F0858EE23;
FFFFFFFFFFFF6EBFBA8EB2B4;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;
07: D      -
F2DB00163D275ED0CD33EDB1;     D05C9DA119CFE65AD2280C01;     E54F0867E3357AFA2892DC64;
0000000000000A79F80818B98;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;
08: PQ    P/Q
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;
FFFFFFFFFFFFFFFFFFFFFFFF;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;
09: PQu  Pu/Qu
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;
10: Ul   Uu/A5l
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;
11: A0   A0L/A0H
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;
12: A1   A1L/A1H
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
0000FFFF0000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;
13: A2   A2L/A2H
612EA7ACB73B9AB68DD07178;     FFFFFF7C870DBCB4E63396D1;     FFFFFFFFFFFFFFFFFFFFFFFF;
FFFFFFFFFFFFFFFFFFFFFFFF;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;
14: A3   A3L/A3H
FAC88B97951A2B715235E7B0;     68401658FE65F68E81703C23;     02894ACD73D7322F0858EE23;
FFFFFFFFFFFF6EBFBA8EB2B4;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;
15: A4   A4L/A4H
F2DB00163D275ED0CD33EDB1;     D05C9DA119CFE65AD2280C01;     E54F0867E3357AFA2892DC64;
0000000000000A79F80818B98;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;
```

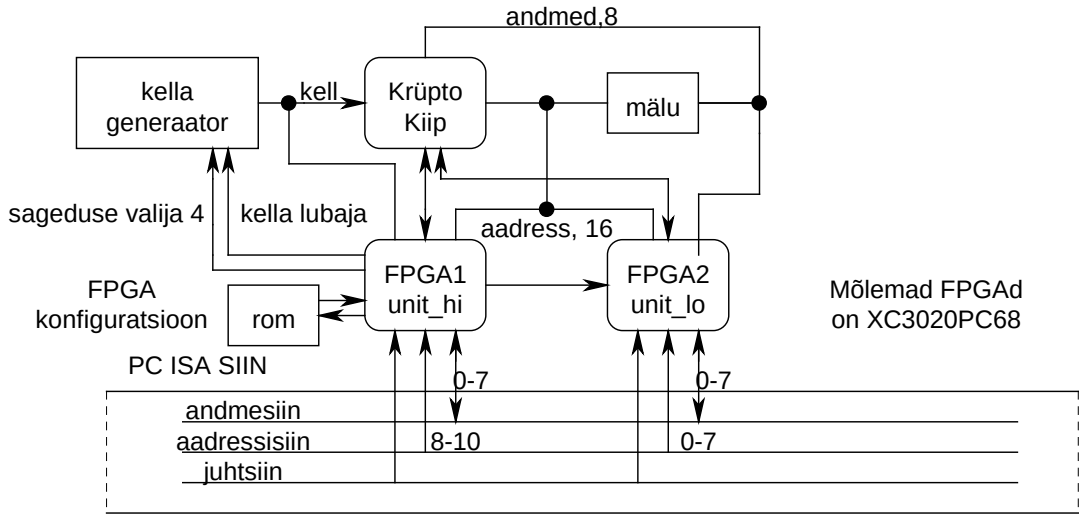
LISA K Mooduliga korrutamise testi tulemused: WAIT.002

```
00: Tr     -
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;
FFFFFFFFFFFFFFFFFFFFFFFF;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;
01: T1     -
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;
02: I0     -
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;
03: I1     -
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;
04: Sr     -
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
0000FFFF0000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;
05: N      -
FEBF057A5555C9AE38DC7F37;     000000041910B03A3DDC1FC3D;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;
06: M      -
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;
07: D      -
4A784B4E627ACD0DC7A8639E;     0000003619C498CB584D6E2C;     000000000000000000000000;
000000000000000000000000;     4A784B4E627ACD0DC7A8639E;     0000003619C498CB584D6E2C;
000000000000000000000000;     000000000000000000000000;
08: PQ    P/Q
FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFFF;
FFFFFFFFFFFFFFFFFFFFFFFF;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;
09: PQu  Pu/Qu
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;
10: Ul   Uu/A5l
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;
11: A0   A0L/A0H
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;
12: A1   A1L/A1H
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
0000FFFF0000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;
13: A2   A2L/A2H
FEBF057A5555C9AE38DC7F37;     000000041910B03A3DDC1FC3D;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;
14: A3   A3L/A3H
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;
15: A4   A4L/A4H
4A784B4E627ACD0DC7A8639E;     0000003619C498CB584D6E2C;     000000000000000000000000;
000000000000000000000000;     4A784B4E627ACD0DC7A8639E;     0000003619C498CB584D6E2C;
000000000000000000000000;     000000000000000000000000;
```

LISA L Diskreetne eksponent, testi tulemused: WAIT.003

```
00: Tr     -
FFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFF;
FFFFFFFFFFFFFFFFFFFFFFF;     000000000000000000000001;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;
01: T1     -
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;
02: I0     -
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000001;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;
03: I1     -
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;
04: Sr     -
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
0000FFFF0000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;
05: N      -
000000000000000000010001;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;
06: M      -
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;
07: D      -
000000000000000000010000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     4A784B4E627ACD0DC7A8639E;     0000003619C498CB584D6E2C;
000000000000000000000000;     000000000000000000000000;
08: PQ    P/Q
FFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFF;     FFFFFFFFFFFFFFFFFFFFFFF;
FFFFFFFFFFFFFFFFFFFFFFF;     000000000000000000000001;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;
09: PQu  Pu/Qu
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;
10: Ul   Uu/A5l
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000001;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;
11: A0   A0L/A0H
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;
12: A1   A1L/A1H
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
0000FFFF0000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;
13: A2   A2L/A2H
000000000000000000010001;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;
14: A3   A3L/A3H
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     000000000000000000000000;
15: A4   A4L/A4H
000000000000000000010000;     000000000000000000000000;     000000000000000000000000;
000000000000000000000000;     4A784B4E627ACD0DC7A8639E;     0000003619C498CB584D6E2C;
000000000000000000000000;     000000000000000000000000;
```

LISA M SVM moodul



Mõlemad FPGAd on	XC3020PC64-125
Staatiline mälu	IS61C256-20
Kellageneraator	AV9107-03CN14
FPGA konf. mälu	1736A

Joonis 15. SVM ISA moodul

Kiibi väljundid on koormatud FPGA ja mälu sisenditega. FPGA sisendmahtuvus on 10 pF, mälu sisendmahtuvus on 12 pF. Koormusele on lisatud mahtuvused, et saavutada soovitud 100 pF maksimaalset koormust.
Liidese täpne elektriline skeem (VHDL kirjeldus) on [SVMDOK]. Liides on realiseeritud makettplaadil.

Liidese elektriline skeem, VHDL kirjeldus, viikude paigutus ning süntesaatori juhtprogrammi on järgnevatel lehekülgedel:

SVM ISA MOODUL. VHDL KIRJELDUS 60

UNIT_HI viikude paigutus (XILINX 3020 on CC68 korpuses) 62

UNIT_LO viikude paigutus (XILINX 3020 on CC68 korpuses) 64

UNIT_HI VHDL kirjeldus 65

UNIT_LO VHDL kirjeldus 68

Sünteesi juhtprogramm 71

Unit_hi elektriline skeem pärast sünteesi 72

Unit_lo elektriline skeem pärast sünteesi 72

SVM_ISA elektriline skeem 72

SVM ISA MOODUL. VHDL KIRJELDUS

```
LIBRARY IEEE;
use IEEE.std_logic_1164.all;
use WORK.svm_add_comp.all;

entity svm_top is
port (  data_isa: inout std_logic_vector(15 downto  0);
       adr_isa: in std_logic_vector(15 downto 0);
            iord: in std_logic;
            iowr: in std_logic;
       isa_reset: in std_logic;
           clock: in std_logic
);
end svm_top;

architecture aaa of svm_top is
signal      sel_lo: std_logic;
signal      sel_hi: std_logic;
signal  ena_romdat: std_logic;
signal CHIP_DATA: std_logic_vector(15 downto 0);   -- io, aadr/data
<>CC
signal  COMMANDS: std_logic_vector(4  downto 0);  -- commands,      -
>CC
signal  ROMDAT: std_logic_vector(7  downto 0);  -- romdat        <>CC
signal  ID_CHNR: std_logic;             -- idea kanali nr           -
>CC
signal  IO_RAM: std_logic;            -- io voi ram              ->CC
signal  BIST_TEST: std_logic_vector(1 downto 0); -- bist test       -
>CC
signal  RESET: std_logic;           -- reset                ->CC
signal  RUN: std_logic;           -- valjumine debugist      ->CC
signal  MAIN_MODE: std_logic_vector(1 downto 0); -- pohimood        -
>CC
signal  RD_STATUS: std_logic;            -- loe staatust            -
>CC
signal  RUN_CLK: std_logic;           -- jooksukell              ->CC
signal  RAM_OE: std_logic;            -- RAM output enable      ->RAM
signal  RAM_WE: std_logic;            -- RAM write enable       ->RAM
signal  BITIO_MOD: std_logic_vector(1 downto 0); -- BITIO_MODE
->CC
signal  DATAEN: std_logic;            -- DATAEN            ->CC
signal  DO_CLK: std_logic;            -- tegevuste aktiveerija   ->CC
signal  BIST_SEL: std_logic_vector(1 downto 0); -- BISTI SELEKTOR  -
>CC
signal  RND_CLK: std_Logic;             -- SUVA GENERAATORI KELL    -
>CC
signal  EXT_ROM: std_logic;           -- EXT_ROM                ->CC
signal  RAM_ME: std_logic;            -- RAM_ME                 <-CC
signal  BITIO_RDY: std_logic;         -- BITIO_RDY              <-CC
signal  ID_IN_RDY: std_logic;            -- ID_IN_RDY           <-CC
signal  IDIO_RDY: std_logic;          -- IDIO_RDY               <-CC
signal  MAIN_RDY: std_logic;          -- MAIN_RDY               <-CC
signal  WAITING: std_logic;           -- WAITING                  <-
CC
signal  BIST_OUT: std_logic;            -- BIST_OUT               <-
CC
signal      TESTDATA:  std_logic;                   -- state hashing
<-CC
signal        CLKENA: std_logic;               -- kella genaraatori
tristate ->CLKGEN
signal            CLKSEL:  std_logic_vector(3  downto  0);  --  kella
generaatori sagedus
                                          --> CLKGEN
signal      RAMADR: std_logic_vector(14 downto 0); -- malu aadress
```

```
signal   LOGIC_ONE: std_logic;                        -- lihtsalt yks

signal   BIST_CLK:  std_Logic;
begin

LOGIC_ONE<='1';
-- et testida, yhendame kokku asjakesi

KELLAGENE: CLKGEN
port map (
SELDIV=>CLKSEL,
REFCLK=>clock,
HICLK=>RUN_CLK,
OE=>CLKENA
);



unhi: unit_hi
port map(
   CLKSEL=>CLKSEL,
   aadr_hi=>adr_isa(11 downto 8),
   aadr_lo=>adr_isa(2 downto  0),
   data_hi=>data_isa(15 downto 8),
     clock=>clock,
       res=>isa_reset,
      iowr=>iowr,
      iord=>iord,
    CLKENA=>CLKENA,
    sel_lo=>sel_lo,
    sel_hi=>sel_hi,
ena_romdat=>ena_romdat,
    DATAEN=>DATAEN,
    CHD_HI=>CHIP_DATA(15 downto 8),
     CMNDS=>COMMANDS,
   ID_CHNR=>ID_CHNR,
    IO_RAM=>IO_RAM,
BIST_TEST0=>BIST_TEST(0),
BIST_TEST1=>BIST_TEST(1),
     RESET=>RESET,
       RUN=>RUN,
MAIN_MODE0=>MAIN_MODE(0),
MAIN_MODE1=>MAIN_MODE(1),
 RD_STATUS=>RD_STATUS,
   RUN_CLK=>RUN_CLK,
    RAM_OE=>RAM_OE,
    RAM_WE=>RAM_WE
);

unlo: unit_lo
port map (
   aadr_lo=>adr_isa(7 downto 0),
   data_lo=>data_isa(7 downto 0),
     clock=>clock,
       res=>isa_reset,
      iowr=>iowr,
      iord=>iord,
    sel_hi=>sel_hi,
    sel_lo=>sel_lo,
ena_romdat=>ena_romdat,
    CHD_LO=>CHIP_DATA(7 downto 0),
    ROMDAT=>ROMDAT,
BITIO_MOD0=>BITIO_MOD(0),
BITIO_MOD1=>BITIO_MOD(1),
    DATAEN=>DATAEN,
    DO_CLK=>DO_CLK,
 BIST_SEL0=>BIST_SEL(0),
 BIST_SEL1=>BIST_SEL(1),
   RND_CLK=>RND_CLK,
   EXT_ROM=>EXT_ROM,
```

```
    RAM_ME=>RAM_ME,
 BITIO_RDY=>BITIO_RDY,
 ID_IN_RDY=>ID_IN_RDY,
  IDIO_RDY=>IDIO_RDY,
  MAIN_RDY=>MAIN_RDY,
   WAITING=>WAITING,
  BIST_OUT=>BIST_OUT,
  TESTDATA=>TESTDATA
);

RAMADR(13 downto 0)<=CHIP_DATA(13 downto 0);
RAMADR(14)<=CHIP_DATA(15);

STATRAM: RAM
port map(
DATAIO=>ROMDAT,
ADR=>RAMADR,
WE=>RAM_WE,
OE=>RAM_OE,
CS=>LOGIC_ONE
);


PLD001: CORE
port map(
DO_CLK=>DO_CLK,
DATAIO=>CHIP_DATA,
DATAEN=>DATAEN,
BITIO_MOD=>BITIO_MOD,
BITIO_READY=>BITIO_RDY,
IO_RAM=>IO_RAM,
ID_IN_RDY=>ID_IN_RDY,
IDIO_RDY=>IDIO_RDY,
ID_CHNR=>ID_CHNR,
COMMANDS=>COMMANDS,
RD_STATUS=>RD_STATUS,
MAIN_MODE=>MAIN_MODE,
MAIN_READY=>MAIN_RDY,
RND_CLK=>RND_CLK,
RUN_CLK=>RUN_CLK,
ROM_DATA=>ROMDAT,
EXT_ROM=>EXT_ROM,
RESET=>RESET,
WAITING=>WAITING,
RUN=>RUN,
BIST_CLK=>BIST_TEST(1),
BIST_TEST=>BIST_TEST(0),
BIST_SEL=>BIST_SEL,
BIST_OUT=>BIST_OUT,
TESTDATA=>TESTDATA,
RAM_ME=>RAM_ME
);
end aaa;
```

UNIT_HI viikude paigutus (XILINX 3020 on CC68 korpuses)

```
/* tehnoloogia väljund*/
set_attribute current_design "part" -t string "3020PC68-125"
/* io viikude asukoht */
/* isa busi andmed  */
set_attribute find(port, "data_hi<0>"     ) "pad_location" -t string "P53"
set_attribute find(port, "data_hi<1>"     ) "pad_location" -t string "P51"
set_attribute find(port, "data_hi<2>"     ) "pad_location" -t string "P50"
set_attribute find(port, "data_hi<3>"     ) "pad_location" -t string "P49"
```

```
set_attribute find(port, "data_hi<4>"     ) "pad_location" -t string "P48"
set_attribute find(port, "data_hi<5>"     ) "pad_location" -t string "P47"
set_attribute find(port, "data_hi<6>"     ) "pad_location" -t string "P46"
set_attribute find(port, "data_hi<7>"     ) "pad_location" -t string "P43"
/* isa  aadress */
set_attribute find(port, "aadr_lo<0>"    ) "pad_location" -t string "P42"
set_attribute find(port, "aadr_lo<1>"    ) "pad_location" -t string "P41"
set_attribute find(port, "aadr_lo<2>"    ) "pad_location" -t string "P40"
set_attribute find(port, "aadr_hi<0>"    ) "pad_location" -t string "P39"
set_attribute find(port, "aadr_hi<1>"    ) "pad_location" -t string "P38"
set_attribute find(port, "aadr_hi<2>"    ) "pad_location" -t string "P37"
set_attribute find(port, "aadr_hi<3>"    ) "pad_location" -t string "P36"

set_attribute find(port, "CLKSEL<0>"     ) "pad_location" -t string "P33"
set_attribute find(port, "CLKSEL<1>"     ) "pad_location" -t string "P32"
set_attribute find(port, "CLKSEL<2>"     ) "pad_location" -t string "P31"
set_attribute find(port, "CLKSEL<3>"     ) "pad_location" -t string "P29"
/* 3000 seerias need kindla koha peal */
set_attribute find(port, "clock"         ) "pad_location" -t string "P11"
/* viimased isa siini peale minevad asjad */
set_attribute find(port, "iord"          ) "pad_location" -t string "P24"
set_attribute find(port, "iowr"          ) "pad_location" -t string "P23"
/* sisemised  */
set_attribute find(port, "ena_romdat"    ) "pad_location" -t string "P22"
set_attribute find(port, "sel_hi"        ) "pad_location" -t string "P21"
set_attribute find(port, "DATAEN"        ) "pad_location" -t string "P20"
set_attribute find(port, "sel_lo"        ) "pad_location" -t string "P19"

/*  kiibi andmed */
set_attribute find(port, "chd_hi<7>"     ) "pad_location" -t string "P54"
set_attribute find(port, "chd_hi<6>"     ) "pad_location" -t string "P55"
set_attribute find(port, "chd_hi<5>"     ) "pad_location" -t string "P56"
set_attribute find(port, "chd_hi<4>"     ) "pad_location" -t string "P57"
set_attribute find(port, "chd_hi<3>"     ) "pad_location" -t string "P61"
set_attribute find(port, "chd_hi<2>"     ) "pad_location" -t string "P62"
set_attribute find(port, "chd_hi<1>"     ) "pad_location" -t string "P63"
set_attribute find(port, "chd_hi<0>"     ) "pad_location" -t string "P64"
/* kasud  */
set_attribute find(port, "CMNDS<0>"      ) "pad_location" -t string "P65"
set_attribute find(port, "CMNDS<1>"      ) "pad_location" -t string "P66"
set_attribute find(port, "CMNDS<2>"      ) "pad_location" -t string "P67"
set_attribute find(port, "CMNDS<3>"      ) "pad_location" -t string "P2"
set_attribute find(port, "CMNDS<4>"      ) "pad_location" -t string "P3"
/* valjundid */
set_attribute find(port, "RESET"         ) "pad_location" -t string "P4"
set_attribute find(port, "BIST_TEST0"    ) "pad_location" -t string "P5"
set_attribute find(port, "BIST_TEST1"    ) "pad_location" -t string "P6"
set_attribute find(port, "RUN"           ) "pad_location" -t string "P7"
set_attribute find(port, "RUN_CLK"       ) "pad_location" -t string "P8"
```

```
set_attribute find(port, "MAIN_MODE0"      ) "pad_location" -t string "P9"
set_attribute find(port, "MAIN_MODE1"      ) "pad_location" -t string "P12"
set_attribute find(port, "RD_STATUS"       ) "pad_location" -t string "P13"
set_attribute find(port, "ID_CHNR"         ) "pad_location" -t string "P14"
set_attribute find(port, "IO_RAM"          ) "pad_location" -t string "P15"

/* rami juhtimine */
set_attribute find(port, "RAM_OE"          ) "pad_location" -t string "P16"
set_attribute find(port, "RAM_WE"          ) "pad_location" -t string "P17"
/* iocs16 */
set_attribute find(port, "CLKENA"          ) "pad_location" -t string "P68"
```

UNIT_LO viikude paigutus (XILINX 3020 on CC68 korpuses)

```
/* tehnoloogia väljund*/
set_attribute current_design "part" -t string "3020PC68-125"
/* io viikude asukoht */
/* aadress */
set_attribute find(port, "aadr_lo<0>"    ) "pad_location" -t string "P43"
set_attribute find(port, "aadr_lo<1>"    ) "pad_location" -t string "P42"
set_attribute find(port, "aadr_lo<2>"    ) "pad_location" -t string "P41"
set_attribute find(port, "aadr_lo<3>"    ) "pad_location" -t string "P40"
set_attribute find(port, "aadr_lo<4>"    ) "pad_location" -t string "P39"
set_attribute find(port, "aadr_lo<5>"    ) "pad_location" -t string "P38"
set_attribute find(port, "aadr_lo<6>"    ) "pad_location" -t string "P37"
set_attribute find(port, "aadr_lo<7>"    ) "pad_location" -t string "P36"
/* isa siini andmed  */
set_attribute find(port, "data_lo<0>"    ) "pad_location" -t string "P33"
set_attribute find(port, "data_lo<1>"    ) "pad_location" -t string "P32"
set_attribute find(port, "data_lo<2>"    ) "pad_location" -t string "P31"
set_attribute find(port, "data_lo<3>"    ) "pad_location" -t string "P29"
set_attribute find(port, "data_lo<4>"    ) "pad_location" -t string "P24"
set_attribute find(port, "data_lo<5>"    ) "pad_location" -t string "P23"
set_attribute find(port, "data_lo<6>"    ) "pad_location" -t string "P22"
set_attribute find(port, "data_lo<7>"    ) "pad_location" -t string "P21"
/*  kiibi andmed */
set_attribute find(port, "chd_lo<7>"     ) "pad_location" -t string "P46"
set_attribute find(port, "chd_lo<6>"     ) "pad_location" -t string "P47"
set_attribute find(port, "chd_lo<5>"     ) "pad_location" -t string "P48"
set_attribute find(port, "chd_lo<4>"     ) "pad_location" -t string "P49"
set_attribute find(port, "chd_lo<3>"     ) "pad_location" -t string "P50"
set_attribute find(port, "chd_lo<2>"     ) "pad_location" -t string "P51"
set_attribute find(port, "chd_lo<1>"     ) "pad_location" -t string "P53"
set_attribute find(port, "chd_lo<0>"     ) "pad_location" -t string "P54"
```

```
/* romi data  */
set_attribute find(port, "romdat<7>"      ) "pad_location" -t string "P55"
set_attribute find(port, "romdat<6>"      ) "pad_location" -t string "P56"
set_attribute find(port, "romdat<5>"      ) "pad_location" -t string "P57"
set_attribute find(port, "romdat<4>"      ) "pad_location" -t string "P61"
set_attribute find(port, "romdat<3>"      ) "pad_location" -t string "P62"
set_attribute find(port, "romdat<2>"      ) "pad_location" -t string "P63"
set_attribute find(port, "romdat<1>"      ) "pad_location" -t string "P64"
set_attribute find(port, "romdat<0>"      ) "pad_location" -t string "P65"

/* 300 seerias peavad need kaks olema kindlas kohas */
set_attribute find(port, "clock"          ) "pad_location" -t string "P11"

/* sisendid */
set_attribute find(port, "RAM_ME"         ) "pad_location" -t string "P66"
set_attribute find(port, "BITIO_RDY"      ) "pad_location" -t string "P67"
set_attribute find(port, "ID_IN_RDY"      ) "pad_location" -t string "P68"
set_attribute find(port, "IDIO_RDY"       ) "pad_location" -t string "P2"
set_attribute find(port, "MAIN_RDY"       ) "pad_location" -t string "P3"
set_attribute find(port, "WAITING"        ) "pad_location" -t string "P4"

set_attribute find(port, "BIST_OUT"       ) "pad_location" -t string "P5"
set_attribute find(port, "TESTDATA"       ) "pad_location" -t string "P6"
set_attribute find(port, "BIST_SEL0"      ) "pad_location" -t string "P7"
set_attribute find(port, "DO_CLK"         ) "pad_location" -t string "P8"
set_attribute find(port, "DATAEN"         ) "pad_location" -t string "P9"
set_attribute find(port, "BITIO_MOD1"     ) "pad_location" -t string "P12"
set_attribute find(port, "BITIO_MOD0"     ) "pad_location" -t string "P13"

set_attribute find(port, "ena_romdat"     ) "pad_location" -t string "P14"
set_attribute find(port, "iord"           ) "pad_location" -t string "P15"
set_attribute find(port, "iowr"           ) "pad_location" -t string "P16"
set_attribute find(port, "sel_hi"         ) "pad_location" -t string "P30"
/* valjundid */
set_attribute find(port, "BIST_SEL1"      ) "pad_location" -t string "P17"
set_attribute find(port, "RND_CLK"        ) "pad_location" -t string "P19"
set_attribute find(port, "EXT_ROM"        ) "pad_location" -t string "P20"
set_attribute find(port, "sel_lo"         ) "pad_location" -t string "P28"
```

UNIT_HI VHDL kirjeldus

```
-- unit high
-- madalate andmete ja madalate aadressite osa
LIBRARY IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use WORK.svm_isa_pack.all;
```

```
entity unit_hi is
port (
    CLKSEL: out    std_logic_vector(3 downto 0); -- clock chip control
   aadr_hi: in     std_logic_vector(3 downto 0); -- aadressi ylemine osa
   aadr_lo: in     std_logic_vector(2 downto 0); -- aadressi alumine osa
   data_hi: inout  std_logic_vector(7 downto 0); -- data low
     clock: in     std_logic;   -- kell
       res: in     std_logic;   -- reset
      iowr: in     std_logic;   -- io write
      iord: in     std_logic;   -- io read
    CLKENA: out    std_logic;   -- selects external clock generator
    sel_lo: in     std_logic;   -- select hi, ylemise aadressi selektor
    sel_hi: out    std_logic;   -- registrite selektor
 ena_romdat: out    std_logic;                    -- R7 2
    DATAEN: in     std_logic;
    chd_hi: inout  std_logic_vector(7 downto 0); -- R1
     CMNDS: out    std_logic_vector(4 downto 0); -- R3 7..2
   ID_CHNR: out    std_logic;                          -- R3 1
    IO_RAM: out    std_logic;                    -- R3 0
 BIST_TEST0: out    std_logic;                -- R5 7
 BIST_TEST1: out    std_logic;                          -- R5 6
     RESET: out    std_logic;                    -- R5 5
       RUN: out    std_logic;                    -- R5 4
 MAIN_MODE0: out    std_logic;                          -- R5 3
 MAIN_MODE1: out    std_logic;                          -- R5 2
  RD_STATUS: out    std_logic;                          -- R5 1
   RUN_CLK: out    std_logic;                          -- R5 0
    RAM_OE: out    std_logic;                          -- R7 1
    RAM_WE: out    std_logic                           -- R7 0
);
end unit_hi;

architecture BEHA of unit_hi is
signal R1_reg, R1_rd: std_logic_vector(7 downto 0);   -- registerid
signal R3_reg: std_Logic_vector(6 downto 0);
signal R3_rd: std_logic_vector(7 downto 0);   --
signal R5_reg, R5_rd: std_logic_vector(7 downto 0);   --
signal R7_rd:  std_logic_vector(7 downto 0);   -- valjundiga
signal R7_reg: std_logic_vector(7 downto 0);
signal sel_r1, sel_r3, sel_r5, sel_r7: std_logic;            -- registrite dekoodrid
signal RUN_CLK_reg:  std_logic;
signal RUN_CLK_div:    std_Logic;
signal data_hi_out, data_hi_in: std_logic_vector(7 downto 0); -- main data
kommutaator
signal CNTR, CNT_IN:  unsigned(7 downto 0);
signal presc: std_logic_vector(7 downto 0);
begin
-- data sisendi tristate
data_hi_in<=data_hi;
```

```
tri_din: process(  data_hi_out, iord,
                sel_r1, sel_r3, sel_r5, sel_r7)
begin
if( (iord=IORD_ACTIVE) and
   ( sel_r1='1' or  sel_r3='1' or sel_r5='1' or sel_r7='1' ) ) then
      data_hi<=data_hi_out;
 else  data_hi<=(others=>'Z');
 end if;
end process;

-- data tristate
R1_rd<=chd_hi;
tri_dat: process(  R1_reg, DATAEN )
begin
if( DATAEN = '0' ) then chd_hi<=R1_reg;
                    else chd_hi<=(others=>'Z');
                end if;
end process;
-- valjundid 1
( CMNDS(4), CMNDS(3), CMNDS(2), CMNDS(1),
  CMNDS(0), ID_CHNR, IO_RAM  ) <= R3_reg;
r3_rd(6 downto 0)<=r3_reg;
r3_rd(7)<='0';
-- valjundid 2
( BIST_TEST0, BIST_TEST1, RESET,     RUN,
  MAIN_MODE0, MAIN_MODE1, RD_STATUS, RUN_CLK_reg ) <=R5_reg;
r5_rd<=r5_reg;
-- valjundid 3
CLKSEL<=R7_reg(7 downto 4);
RAM_OE<=R7_reg(2);
RAM_WE<=R7_reg(1);
ena_romdat<= R7_reg(0);
r7_rd<=R7_reg;

CLKENA<= R7_reg(3);

RUN_CLK<=  RUN_CLK_REG when R7_reg(3) = '0' else
         'Z';



-- salvestaja
REGS: process( res, clock )
begin
if(res = RES_ACTIVE ) then
R1_reg<=(others => '0');
R3_reg<=(others => '0');
R5_reg<=(others => '0');
R7_reg<=(others => '0');
```

```
presc<=(others=>'0');
elsif(clock'EVENT and clock=ACT_FRONT ) then
 if(  iowr = IOWR_ACTIVE ) then
   if( sel_r1 = '1' ) then R1_reg<=data_hi_in; end if;
   if( sel_r3 = '1' ) then R3_reg<=data_hi_in(6 downto 0); end if;
   if( sel_r5 = '1' ) then R5_reg<=data_hi_in; end if;
   if( sel_r7 = '1' ) then R7_reg<=data_hi_in; end if;
  end if;
end if;
end process;
-- lugeja

data_hi_out<= R1_rd when sel_r1='1' else
            R3_rd when sel_r3='1' else
            R5_rd when sel_r5='1' else
            R7_rd;

-- aadressi kalkulaator
BASECALC: process( aadr_hi, aadr_lo, sel_lo )
begin
sel_r1<='0';
sel_r3<='0';
sel_r5<='0';
sel_r7<='0';
sel_hi<='0';
if(aadr_hi(3 downto 0)=BASE_11to8 ) then
sel_hi<='1';
 if(sel_lo='1') then
  if(aadr_lo="001") then sel_r1<='1'; end if;
  if(aadr_lo="011") then sel_r3<='1'; end if;
  if(aadr_lo="101") then sel_r5<='1'; end if;
  if(aadr_lo="111") then sel_r7<='1'; end if;
 end if;
end if;
end process;

end BEHA;
```

UNIT_LO VHDL kirjeldus

```
-- unit low
-- madalate andmete ja madalate aadressite osa
LIBRARY IEEE;
use IEEE.std_logic_1164.all;
use WORK.svm_isa_pack.all;
entity unit_lo is
port (
   aadr_lo: in     std_logic_vector(7 downto 0); -- alumised bitid datat
   data_lo: inout  std_logic_vector(7 downto 0); -- data low
```

```
      clock: in     std_logic;   -- kell
        res: in     std_logic;   -- reset
       iowr: in     std_logic;   -- io write
       iord: in     std_logic;   -- io read
     sel_hi: in     std_logic;   -- select hi, ylemise aadressi selektor
     sel_lo: out    std_logic;   -- registrite selektor
 ena_romdat: in     std_logic;
     chd_lo: inout  std_logic_vector(7 downto 0); -- R0
     romdat: inout  std_logic_vector(7 downto 0); -- R2
 BITIO_MOD0: out    std_logic;   -- R4 7
 BITIO_MOD1: out    std_logic;   -- R4 6
     DATAEN: out    std_logic;   -- R4 5
     DO_CLK: out    std_logic;   -- R4 4
  BIST_SEL0: out    std_logic;   -- R4 3
  BIST_SEL1: out    std_logic;   -- R4 2
    RND_CLK: out    std_logic;   -- R4 1
    EXT_ROM: out    std_logic;   -- R4 0
     RAM_ME: in     std_logic;   -- R6 7
  BITIO_RDY: in     std_logic;   -- R6 6
  ID_IN_RDY: in     std_logic;   -- R6 5
   IDIO_RDY: in     std_logic;   -- R6 4
   MAIN_RDY: in     std_logic;   -- R6 3
    WAITING: in     std_logic;   -- R6 2
   BIST_OUT: in     std_logic;   -- R6 1
   TESTDATA: in     std_logic    -- R6 0
);
end unit_lo;

architecture BEHA of unit_lo is
signal R0_reg, R0_rd: std_logic_vector(7 downto 0);   -- registerid
signal R2_reg, R2_rd: std_logic_vector(7 downto 0);   --
signal R4_reg, R4_rd: std_logic_vector(7 downto 0);   --
signal R6_reg, R6_rd:         std_logic_vector(7 downto 0);   -- valjundiga
signal sel_r0, sel_r2, sel_r4, sel_r6: std_logic;             -- registrite dekoodrid
signal data_lo_out, data_lo_in: std_logic_vector(7 downto 0); -- main data
kommutaator
signal DATAEN_i: std_logic;
begin
-- data sisendi tristate
data_lo_in<=data_lo;
tri_din: process(  data_lo_out, iord,
                sel_r0, sel_r2, sel_r4, sel_r6 )
begin
if( iord=IORD_ACTIVE and ( sel_r0='1' or  sel_r2='1' or
                        sel_r4='1' or sel_r6='1'    )
   ) then data_lo<=data_lo_out;
 else  data_lo<=(others=>'Z');
 end if;
end process;
```

```
DATAEN<=DATAEN_i;
-- data tristate
R0_rd<=chd_lo;
tri_dat: process(  R0_reg, DATAEN_i )
begin
if( DATAEN_i = '0' ) then chd_lo<=R0_reg;
                     else chd_lo<=(others=>'Z');
                end if;
end process;
-- romdata tristate
R2_rd<=romdat;
tri_ROMD: process(  R2_reg, ena_romdat )
begin
if( ena_romdat = '1' )  then romdat<=R2_reg;
                        else romdat<=(others=>'Z');
                    end if;
end process;
r4_rd<=r4_reg;
-- kiibi valjundite juhtija
( BITIO_MOD0, BITIO_MOD1, DATAEN_i, DO_CLK,
  BIST_SEL0,  BIST_SEL1,  RND_CLK,   EXT_ROM ) <= R4_reg;

-- kiibi sisendite lugeja
R6_rd <=(RAM_ME,     BITIO_RDY,  ID_IN_RDY, IDIO_RDY,
       MAIN_RDY,   WAITING,    BIST_OUT,  TESTDATA);

-- salvestaja

REGS: process( res, clock )
begin
if(res = RES_ACTIVE ) then
R0_reg<=(others => '0');
R2_reg<=(others => '0');
R4_reg<=(others => '0');
elsif(clock'EVENT and clock=ACT_FRONT ) then
 if( iowr = IOWR_ACTIVE ) then
   if( sel_r0 = '1' ) then R0_reg<=data_lo_in; end if;
   if( sel_r2 = '1' ) then R2_reg<=data_lo_in; end if;
   if( sel_r4 = '1' ) then R4_reg<=data_lo_in; end if;
 end if;
end if;
end process;
-- lugeja
data_lo_out<= R0_rd when sel_r0='1' else
            R2_rd when sel_r2='1' else
            R4_rd when sel_r4='1' else
            R6_rd;
```

```
-- aadressi kalkulaator
BASECALC: process( aadr_lo, sel_hi )
begin
sel_r0<='0';
sel_r2<='0';
sel_r4<='0';
sel_r6<='0';
sel_lo<='0';
if(aadr_lo(7 downto 3)=BASE_7to3 ) then
sel_lo<='1';
 if(sel_hi='1') then
  if(aadr_lo(2 downto 0)="000") then sel_r0<='1'; end if;
  if(aadr_lo(2 downto 0)="010") then sel_r2<='1'; end if;
  if(aadr_lo(2 downto 0)="100") then sel_r4<='1'; end if;
  if(aadr_lo(2 downto 0)="110") then sel_r6<='1'; end if;
 end if;
end if;
end process;

end BEHA;
```

Sünteesi juhtprogramm

```
/* see synteseerib siluja valismooduli
   isa siini peale kaiva variandi. lihtsalt muidu kiirus ei
   tule valja. See synteesib kogu asja. ei ole mingit erilist
   hierarhiat vaja, kuna kogu asi on nii lihtne */
DESGS = { unit_hi }
KELL = "clock"
RESET = "res"
foreach ( DES, DESGS )
{
current_design  DES
create_clock -period 100 KELL
ungroup -all -flatten
set_input_delay  10 all_inputs()
set_output_delay 10 all_outputs()
/* nyyd paneme pordid kylge */
IOPADS = find(port, "*")
IOPADS = IOPADS - find( port, RESET )
set_port_is_pad IOPADS
ungroup -all -flatten
set_pad_type -vih 3.33 -vil 1.05 all_inputs()
set_pad_type -voh 3.33 -vol 1.05 all_outputs()
set_pad_type -slewrate NONE IOPADS
insert_pads
/* kompileerime asja kokku */
compile -map_effort high
/* kuhu on padid ja milline jupp */
```

```
/* 3000 on gsr hardwired ja negatiivselt aktiivne
   lihtsalt koristame ara meie reseti neti */
replace_fpga
remove_net  all_connected( find(port, RESET ) )
remove_port find(port, RESET )
include "p2_" + DES
write -f edif -hier -o DES + ".sedif"
write -f xnf -hier -o DES + ".sxnf"
}
```

Järgneval kolmel lehel on:

Unit_hi elektriline skeem pärast sünteesi
Unit_lo elektriline skeem pärast sünteesi
SVM_ISA elektriline skeem