Vaasan Yliopisto

Teknillinen Tiedekunta

Automaatiotekniikka

Olli Ketola

8-Bittisen Tietokoneen Toteutus DE2 FPGA Kehitysalustalla

Digitaalipiirien Mallinnus

Vaasassa 08.06.2018

Jarmo Alander

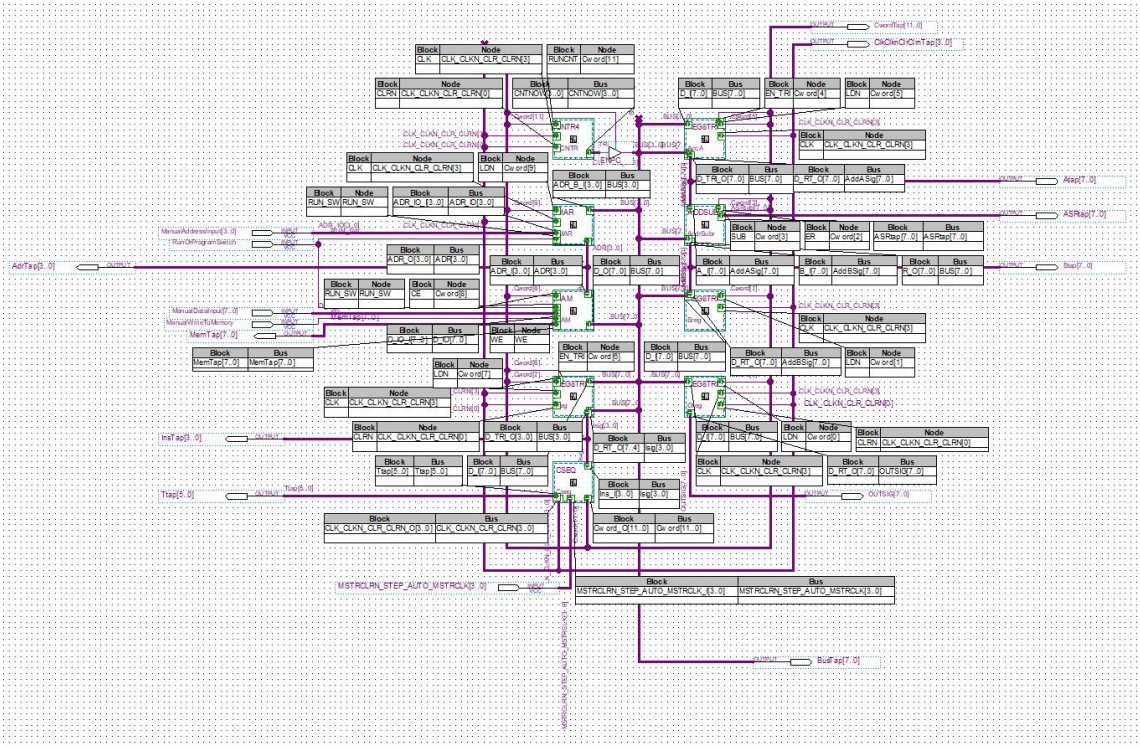
SISÄLLYSLUETTELO

1 JOHDANTO

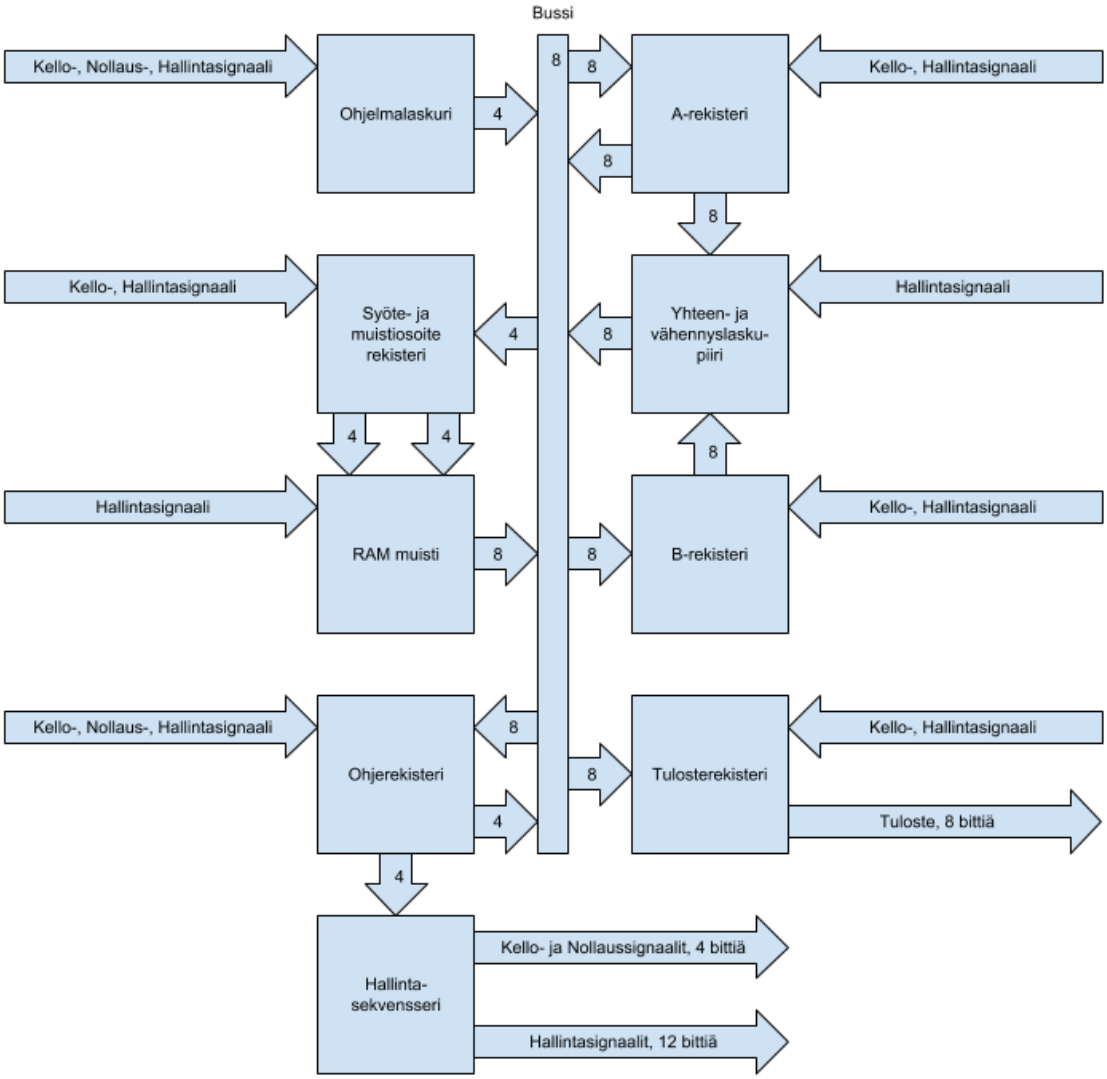
Tämä raportti käsittelee yksinkertaisen 8-bittisen tietokoneen toteutusta FPGA kehitysalustan avulla. Toteutettu tietokone on versio Paul Malvinon ja Jerald A Brownin kirjassa Digital Computer Electronics (1999) esiintyvästä SAP-1 tietokoneesta. Kirjassa esiintyvän TTL-piireihin perustuvan esimerkkitietokoneen piirikaavioita on sovellettu vastaavan toiminnallisuuden toteuttamiseksi Vaasan Yliopistolta lainatulle Altera DE2 FPGA kehitysalustalle. Kyseinen toiminnallisuus on toteutettu piirikaavioina Altera Quartus II 13.0sp1 Web Edition sovelluksen avulla.

2 TIETOKONEEN TOTEUTUS

Kuva 1 esitää tietokoneen pääpiirin toteutusta Quartus sovelluksen piirikaavio editorilla. Kuva 2 on yksinkertaistettu esitys vastaavan tietokoneen arkkitehtuurista. Se koostuu yhdestä jaetusta bussista, viidestä rekisteristä, ohjelmalaskurista, hallintasekvensseristä, yhteen- ja vähennyslaskupiiristä, sekä signaaleista kelloille, resetoinnille ja hallinnalle. (Malvino & Brown 1999: 141)



Kuva 1. SAP-1 tietokoneen pääpiiri Quartuksella toteutettuna, SAP1.bdf (Malvino & Brown 1999: 141, 166-171)



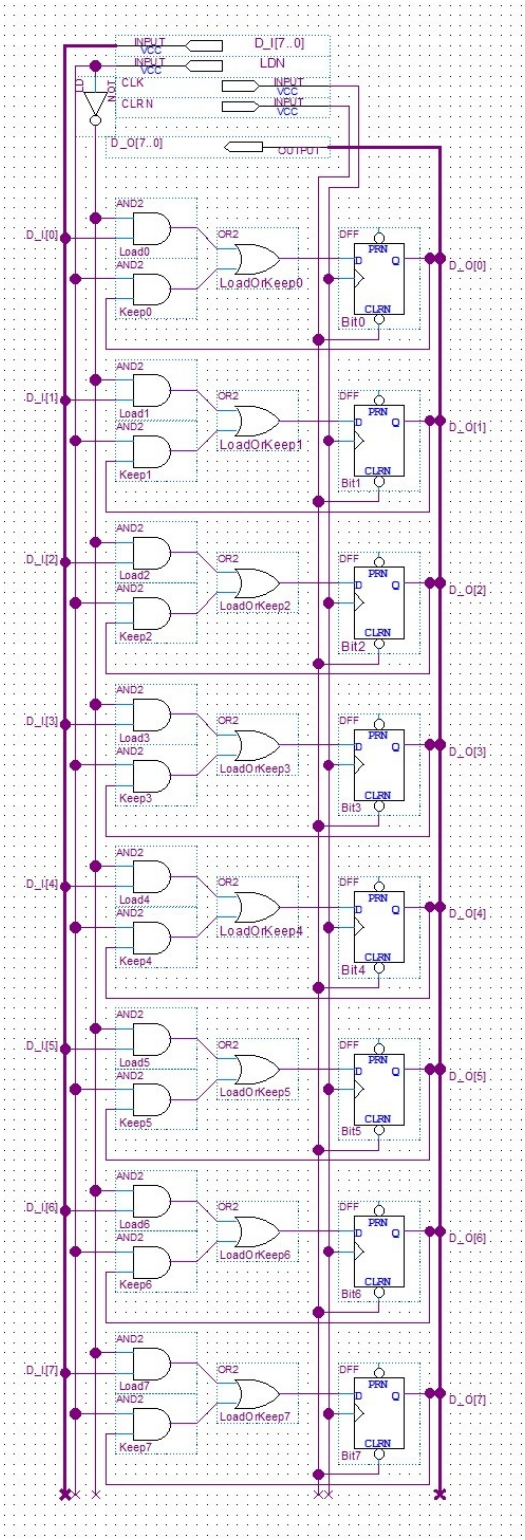
Kuva 2. SAP-1 tietokoneen yksinkertaistettu arkkitehtuuri. (Malvino & Brown 1999: 141)

2.1 Rekisterit

SAP-1 tietokone sisältää viisi eri 8 bittistä rekisteriä: A-, B-, muistiosoite-, ohje- ja tulosterekisterit. Muistiosoiterekisteristä vain alemmat 4 bittiä ovat käytössä. Kaikki rekisterit perustuvat pohjimmiltaan kuvan 3 esittämän REG8.bdf piirikaavion instansseihin.

D_I[7..0] Signaali on rekisteriin luettavan datan signaali. LDN on käänteinen lataussignaali, jonka matalalle pudotessa D_I[7..0] signaalin arvo luetaan rekisteriin. CLK on rekisterin kellosignaali. Ja CLRN on rekisterin käänteinen nollaus signaali, jonka matalalle pudotessa rekisterin sisältö nollataan.

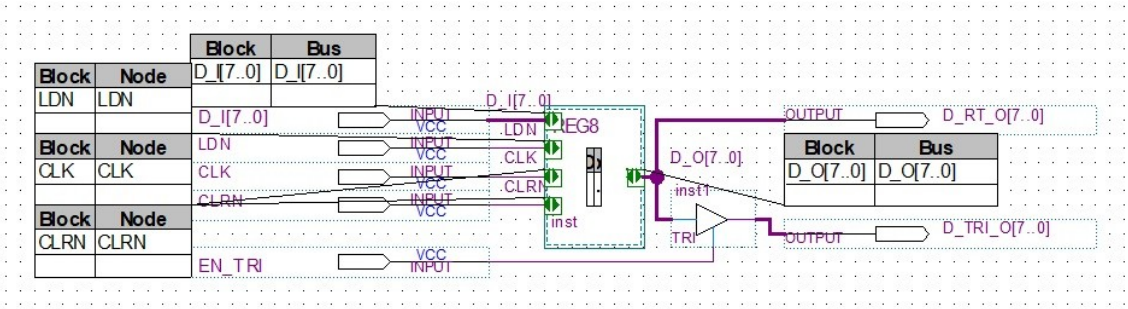
Rekisterin sisältämä data on tallennettu kahdeksaan kellotettuun D-kytkimeen. Jotka kellon käydessä ja LDN signaalista riippuen, joko tallentavat omaa bittiään vastaavan sisääntulevan datan, tai säilyttävät sen hetkisen datansa kopioimalla oman sisältönsä takaisin itseensä. CLK ja CLRN signaalit on kytketty suoraan jokaisen D-kytkimen CLK ja CLRN sisääntuloihin. Kuvan 3 vasemman reunan AND ja OR logiikkapiiri rykelmät laskevat LDN signaalin perustella ollaanko rekisteriin lukemassa uutta dataa. (Malvino & Brown 1999: 106-107)



Kuva 3. 8 bittinen rekisteri, REG8.bdf (Malvino & Brown 1999: 107)

Muistiosoiterekisteriä lukuun ottamatta, kaikki rekisterit käyttävät kuvan 4 mukaisen REG8TRI.bdf piirin instansseja. Kyseinen piiri koostuu vain aikaisemman REG8.bdf

piirin instanssista, jonka ulostulosignaali on jaettu kahteen signaaliin, joista D_RT_O[7..0] on rekisterin reaaliaikainen sisältö, ja D_TRI_O[7..0] on rekisterin sisältö korkean impedanssin kolmitilaulostulo kytkimen takana. Kyseistä kolmitilaulostuloa hallitsemaan on lisätty uusi EN_TRI sisääntulo. Tässä piirissä D_RT_O on tarkoitettu rekisterin sisällön näyttämiseen ulostulo LED-valoissa, ja D_TRI_O on tarkoitettu rekisterin sisällön bussiin kytkemiseen.

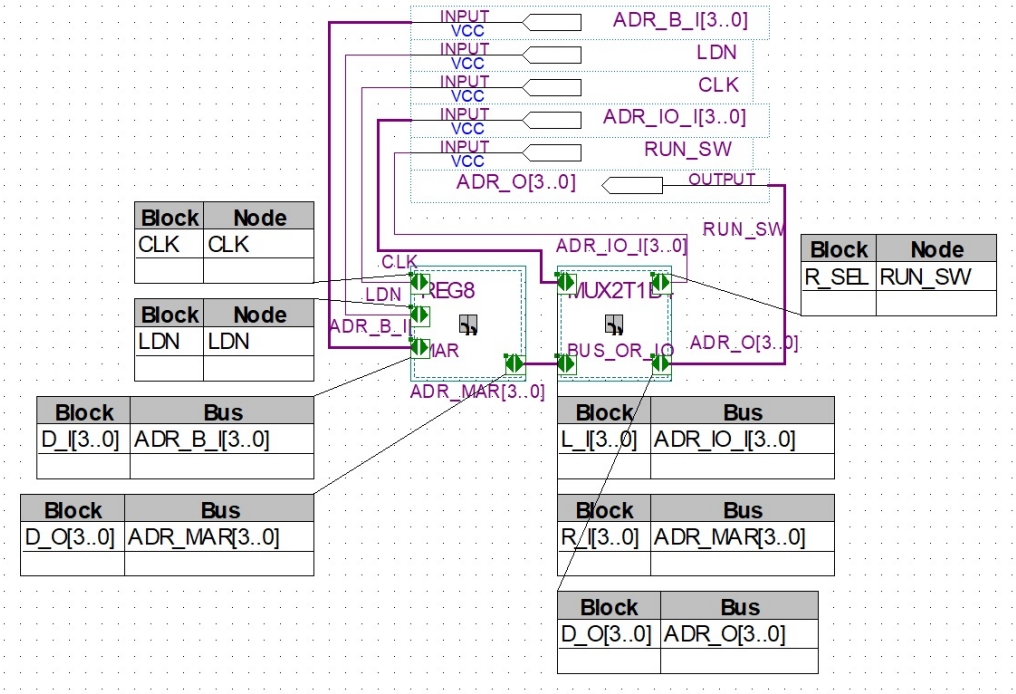


Kuva 4. 8 bittinen rekisteri korkean impedanssin kolmitilaulostulolla, REG8TRI.bdf

2.1.1 Syöte ja muistiosoiterekisteri

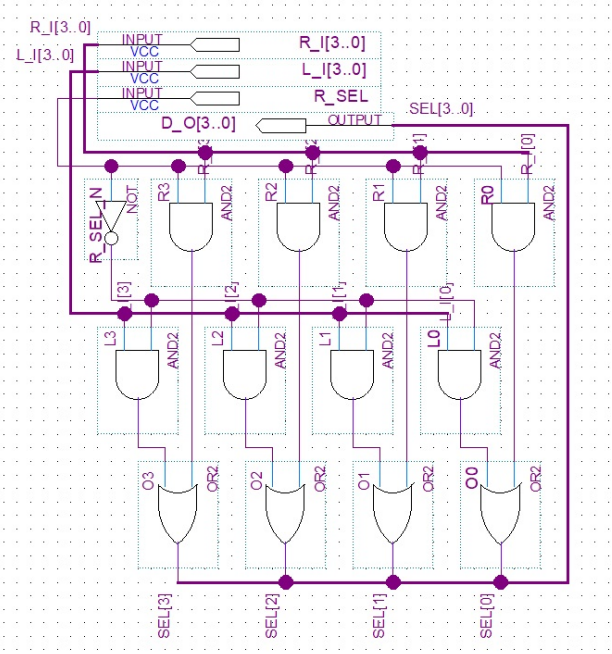
Kuva 5 esittää tietokoneen syötteen ja muistiosoiterekisterin piiriä. Tämä piiri sisältää REG8.bdf muistiosoiterekisteripiirin lisäksi myös muistiosoitteen manuaalisen valinnan logiikan. Kyseinen logiikka koostuu multiplexeristä, joka valitsee, luetaanko aktiivinen muistiosoite muistiosoiterekisteristä, vaiko manuaalisesta muistiosoite syötteestä. Kyseisellä muistiosoitteella valitaan, mitä tietokoneen muistipiirin 16 muistiosoitteesta ollaan lukemassa tai kirjoittamassa. (Malvino & Brown 1999: 140, 153-155)

ADR_B_I[3..0] signaali on bussin 4 alempaa bittiä. LDN on muistiosoiterekisterin käänteinen lataussignaali, joka kertoo, koska uusi muistiosoite tulisi lukea bussista. CLK on tietokoneen ajokellon signaali. ADR_IO_I[3..0] on manuaalisen muistiosoitteen syöte. Ja RUN_SW on tietokoneen sen hetkinen moodi, jonka perusteella multiplexer valitsee, kumpaa muistiosoitetta käytetään. Ajomoodissa käytetään muistiosoiterekisteriä ja ohjelmointimoodissa manuaalisen muistiosoitteen syötettä. (Malvino & Brown 1999: 153)



Kuva 5. Syötteen ja muistiosoiterekisterin piiri, IMAR.bdf (Malvino & Brown 1999: 154)

Kuva 6 esittää IMAR.bdf piirissä käytetyn multiplexerin piirikaavion. Piiri yksinkertaisesti valitsee R_SEL hallintasignaalin avulla, kumpi R_I[3..0] ja L_I[3..0] sisääntulo signaaleista ohjataan D_O[3..0] ulostulosignaaliin. (Malvino & Brown 1999: 58-60)



Kuva 6. 4 bitin kahdesta yhteen multiplexeri, MUX2T1B4.bdf (Malvino & Brown 1999: 60)

2.1.2 Ohjerekisteri

Ohjerekisteriä käytetään tallentamaan tietokoneen prosessointia ohjaavia ohjekoodeja. Sen ylempää neljää bittiä käytetään tallentamaan varsinainen ohjekoodi ja se lähetetään aina suoraan hallintasekvensserille. Jos kyseessä on ohje joka manipuloi muistia, niin sen alempaa neljää bittiä käytetään tallentamaan muistiosoite, jota kyseisellä ohjeella ollaan manipuloimassa. Kyseinen osoite lähetetään sitten tarvittaessa bussiin kulmiulostulon kautta. (Malvino & Brown 1999: 141, 153, 155)

2.1.3 A-rekisteri

A-rekisteriä käytetään tallentamaan väliaikaisia tuloksia tietokoneen ajon aikana. Sen tallentama data lähetetään jatkuvasti suoraan yhteen- ja vähennyslaskupiirille, ja myös tarvittaessa bussiin kolmiulostulon kautta sen EN_TRI hallintasignaalin tilasta riippuen. (Malvino & Brown 1999: 142, 156, 158)

2.1.4 B-rekisteri

B-rekisteriä käytetään yhteen- ja vähennyslasku operaatioissa. Sen sisältämä data lähetetään jatkuvasti yhteen- ja vähennyslaskupiirille, jolloin se automaattisesti lasketaan yhteen A-rekisterin sisällön kanssa. B-rekisteriin luetaan uutta dataa bussista sen LDN hallintasignaalin tilasta riippuen. (Malvino & Brown 1999: 142, 157-158)

2.1.5 Tulosterekisteri

Tulosterekisteriä käytetään tallentamaan tietokoneen suorittaman laskutoimituksen lopullinen tulos. Kun tietokoneelle annetaan OUT ohje, niin A-rekisterin sisältö kopioidaan tulosterekisteriin, josta se ohjautuu tietokoneen ulostuloon. Tämän projektin tietokoneen toteutuksessa tulosterekisterin sisältö ohjataan kehitysalustan kahdeksan 7-segmenttinäytön alaoikeisiin pystysegmentteihin, jolloin tuloksen voi lukea binäärimuodoosa sen perusteella, mitkä kyseisistä segmenteistä palavat. (Malvino & Brown 1999: 142, 157-158)

2.2 Ohjelmalaskuri

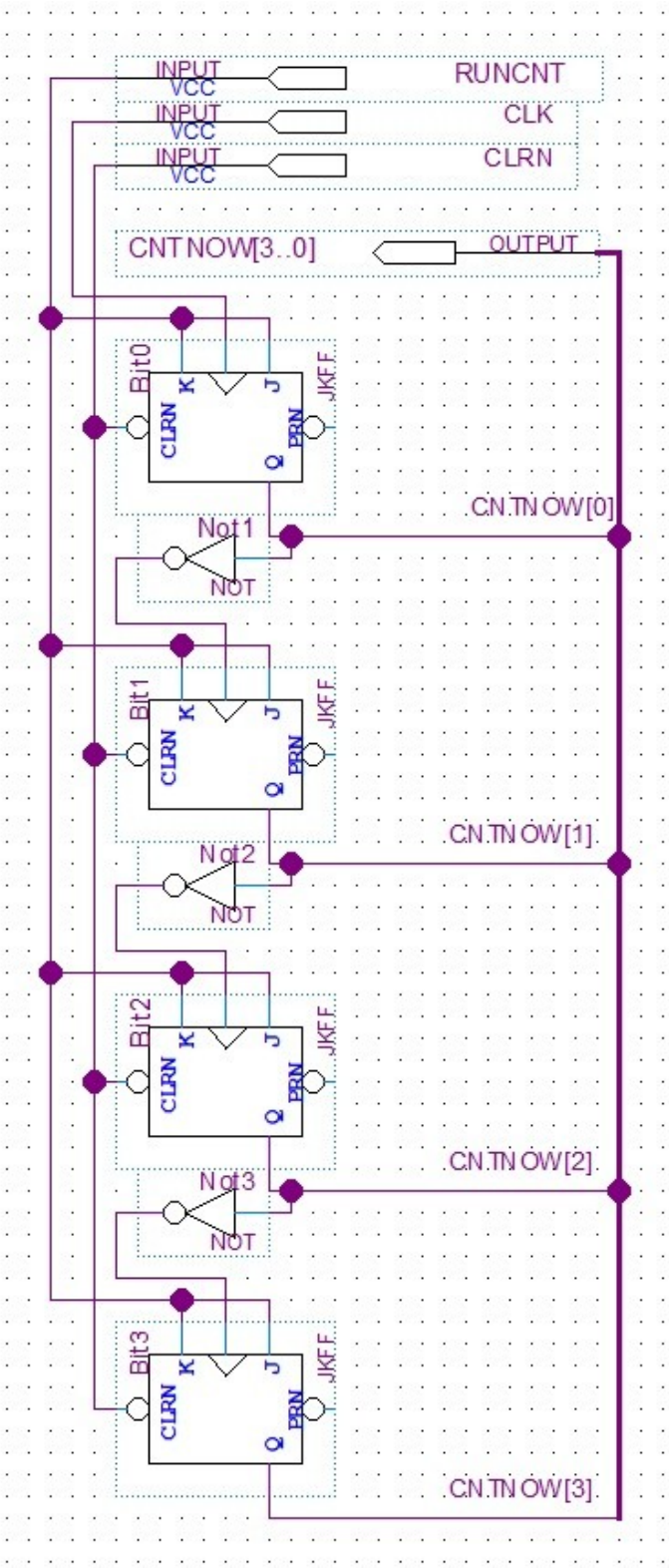
Ohjelmalaskuria käytetään pitämään kirjaa siitä, missä vaiheessa tietokoneen ajama ohjelma on. Se toteuttaa kyseisen tehtävän laskemalla 0000:sta 1111:teen askeltamalla tallentamaansa lukua yhdellä ylöspäin kerran ohjesyklissä. Kun tietokoneen ajama ohjelma ohjelmoidaan muistin alkuun, niin ohjelman ajon aikana voidaan ohjelman milläkin hetkellä tarvitseman ohjekoodin muistiosoite lukea ohjelmalaskurista. (Malvino & Brown 1999: 110-113,140, 153-154)

Ohjelman ajon alkaessa ohjelmalaskuri nollataan, jolloin ohjelmalaskurin arvoksi asetetaan 0000, mikä on ensimmäisen ohjekoodin muistiosoite. Jokaisen ohjekoodin suorituksen yhteydessä ohjelma laskuria askelletaan yhdellä ylöspäin, jolloin ensimmäisen ohjeen suorituksen jälkeen ohjelmalaskurin arvo 0001, eli ohjelman toisen ohjekoodin muistiosoite. Toisen ohjekoodin ajon jälkeen sen arvo on 0010, eli kolmannen ohjekoodin muistiosoite, jne. (Malvino & Brown 1999: 140, 153)

Kuva 7 esittää toteuten ohjelmalaskurin piiriä. RUNCNT signaali askeltaa laskuria yhdellä, CLRN signaali nollaa laskurin ja CLK signaali on tietokoneen ajokellon signaali. CNTNOW[3..0] signaali sisältää ohjelmalaskurin sen hetkisen arvon. Kyseisen signaalin kolmiulostulo kytkin, joka hallitsee koska ohjelmalaskurin arvo luetaan bussiin, on tämän piirin ulkopuolella. Sen voi nähdä kuvan 1 SAP1.bdf piirikaavion vasemmassa yläkulmassa.

Jokaisen bitin JK-kytkimen ulostulo on kytketty käänteisenä seuraavan bitin kelloon. Käänteisenä sen vuoksi että, Malvinon ja Brownin kirjan esimerkki piiri käytti käänteisellä kellolla juoksevia TTL piirejä. Tässä tapauksessa myös CLK signaalin olisi tullut olla käänteinen, mutta toteutettu kone toimii vaikka CLK signaali on unohdettu kääntää tässä toteutuksessa. CLK signaalin juostessa bitti 0:n arvo vaihtelee puolta tahtia CLK signaaliin verrattuna, bitti 1 puolta tahtia bittiin 0 verrattuna, bitti 2 puolta tahtia bittiin 1 verrattuna, ja bitti 3 puolta tahtia bittiin 2 verrattuna. Tällöin laskuri käy lävitse kaikki siihen mahtuvat binääriluvut nollasta alkaen numerojärjestyksessä: 0000, 0001, 0010,

0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110 ja 1111. (Malvino & Brown 1999: 110-113)



Kuva 7. Ohjelmalaskurin piiri, CNTR4.bdf (Malvino & Brown 1999: 113, 154)

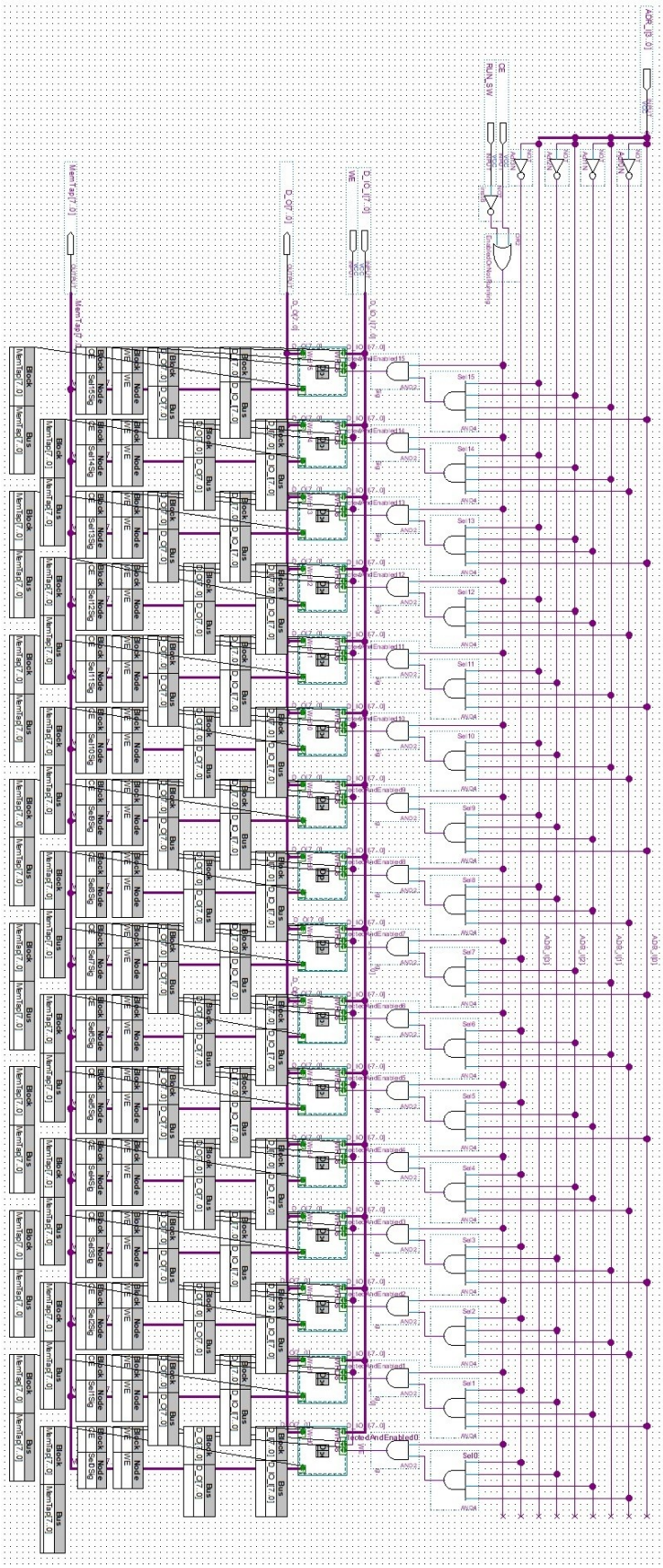
2.3 Muisti

Kuva 8 esittää tietokoneen muistipiiriä. Se koostuu kuudestatoista kuvan 9 mukaisesta muistisanan piiristä, joista kuvan 8 oikealla laidalla oleva dekooderi valitsee muistiosoiterekisterin sisällön perusteella yhden aktiiviseksi. (Eater 2016)

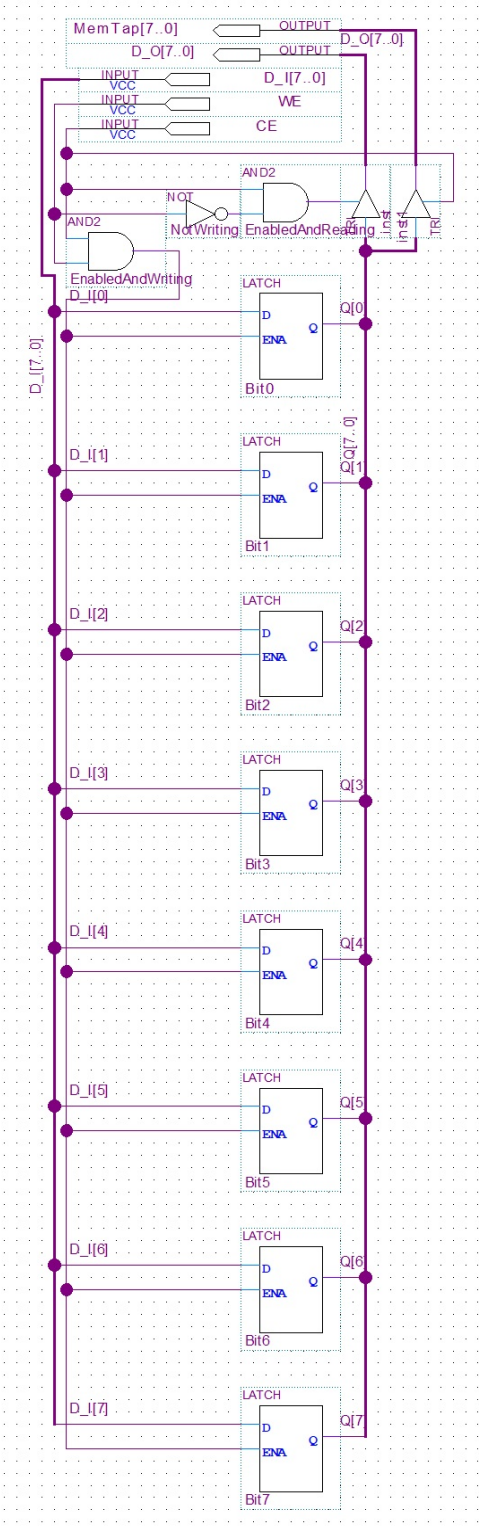
ADR_I[3..0] on muistiosoiterekisteristä saatu muistiosoite. CE signaali on hallintasignaali joka aktivoi muistiosoiterekisterin määrittelemän muistisanan. RUN_SW on hallintasignaali, joka kertoo, onko tietokone ajo- vai ohjelmointimoodissa. WE on kirjoituksen aktivoinnin hallintasignaali, jonka tilan perusteella sisään tulevan datan signaali, D_IO_I[7..0], mahdollisesti kirjoitetaan valittuun ja aktivoituun muistisanaan. D_O[7..0] signaali sisältää valitun ja aktivoidun muistisanan sisällön, jos siihen ei olla kirjoittamassa dataa. Ja MemTap[7..0] on muistin tilan seurantaan tarkoitettu signaali, joka vastaa D_O[7..0] signaalia, mutta ei välitä ollaanko valittuun ja aktivoituun muistisanaan sillä hetkellä kirjoittamassa mitään.

Kuva 9 esittää 8 bitin muistisanan piiriä. Se on käytännössä kellottamattomista D-kytkimistä rakennettu 8 bitin rekisteri. (Eater 2016)

D_I[7..0] on sisääntulevan datan signaali. WE on kirjoittamisen aktivoinnin signaali. CE on piirin aktivoinnin signaali. Jos WE on kirjoitustilassa ja CE on aktivoitu, niin D_I[7..0] signaalin sisältö kirjoitetaan muistisanaan. D_O[7..0] signaali sisältää muistisanan sisällön, jos sana on aktivoitu ja WE on lukutilassa. Ja MemTap[7..0] signaali sisältää muistisanan sisällön jos se on aktivoitu, mutta ei välitä WE signaalin tilasta.



Kuva 8. 16 tavun RAM muistin piiri, RAM.bdf (Eater 2016)



Kuva 9. 8 bitin RAM muistisanan piiri, WRD8.bdf (Eater 2016)

2.4 Hallintasekvensseri

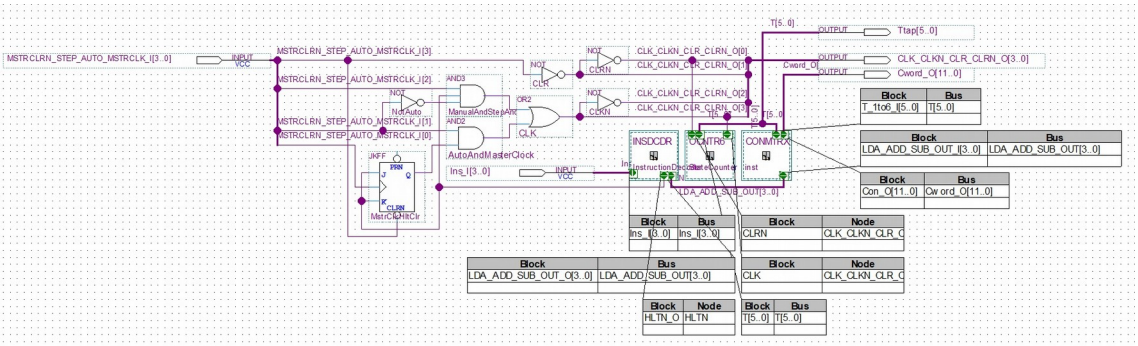
Kuva 10 esittää tietokoneen hallintasekvensserin pääpiiriä. Se koostuu kolmesta alapiiristä, sekä ajonhallinnan logiikasta. Kyseiset alapiirit ovat ohjedekooderi, joka muuttaa ohjerekisterin arvon ohjesignaaliksi. Rinkilaskuri, joka pitää kirjaa kellosyklin sen hetkisestä tilasta. Sekä hallintamatriisi, joka generoi hallintasanan signaalin edellisten kahden piirin generoiman ohjesignaalin ja kellosyklin tilan perusteella. (Malvino & Brown 1999: 141-142, 158-161)

Ajonhallinnan logiikka puolestaan huolehtii kello- ja nollaus signaalien generoimisesta mestarikellon (MSTRCLK), mestarinollauksen komplementin (MSTRCLRN), sekä automaattisen kellon valintasignaalin (AUTO), ja manuaalisen kellon askellussignaalin (STEP) perusteella. (Malvino & Brown 1999: 158-161)

Hallintasekvesserin sisääntulot ovat siis MSTRCLRN_STEP_AUTO_MSTRCLK[3..0], joka on edellä mainittujen signaalien yhdistelmä, sekä INS_I[3..0], joka on ohjerekisterin ylemmän neljän bitin, eli sillä hetkellä suoritettavan ohjeen arvo. Sen ulostuloina ovat CLK_CLKN_CLR_CLRN[3..0], tietokoneen ajokellon ja nollauksen signaalit ja niiden komplementit. Cword[11..0], eli tietokoneen hallintasana, joka hallitsee datan siirtoa tietokoneen komponenttien välillä. Sekä Ttap[5..0], jonka avulla kellosyklin sen hetkisen tilan voi esittää kehitysalustan LED-valoissa. (Malvino & Brown 1999: 158-161)

Tietokoneen CLR ja CLRN nollaus ja nollauksen komplementti signaalit perustuvat suoraan MSTRCLRN signaalin tilaan. Tietokoneen ajokellon CLK ja CLKN signaalit perustuvat STEP, AUTO, MSTRCLK ja ohjedekooderin tuottamaan HLTN signaaliin. Kun AUTO ja HLTN signaalit ovat korkeita, niin tietokoneen ajokello käy MSTRCLK signaalia puolta hitaampaa tahtia. Kun AUTO on matalalla ja HLTN korkealla, niin tietokoneen kellosignaalin tila on sama kuin STEP signaalin tila. Ja kun HLTN on matalalla, niin tietokoneen ajokello ei käy ollenkaan. (Malvino & Brown 1999: 158-161)

Piirin MstrClkHltClr niminen JK-kytkin huolehtii siitä, että tietokoneen automaattinen kello tottelee ohjelman ajon lopettavaa HLT ohjetta. Kun ohjedekooderiin syötetään HLT komennon ohjekoodi 1111. Niin sen HLTN ulostulo tipahtaa alas ja kytkee siten MstrClkHltClr JK-kytkimen J ja K sisääntulot, ja siten koko kytkimen, pois päältä. Tällöin tietokoneen mestarikellon käynti ei enää kytke kyseistä JK-kytkintä päälle ja pois, ja tietokoneen ajokellon signaali CLK ei enää saa säännöllistä mestarikelloon perustuvaa syötettä. HLTN signaalin alas tipahtaminen estää myös STEP signaalin läpipääsyn CLK signaaliin. (Malvino & Brown 1999: 158-161)

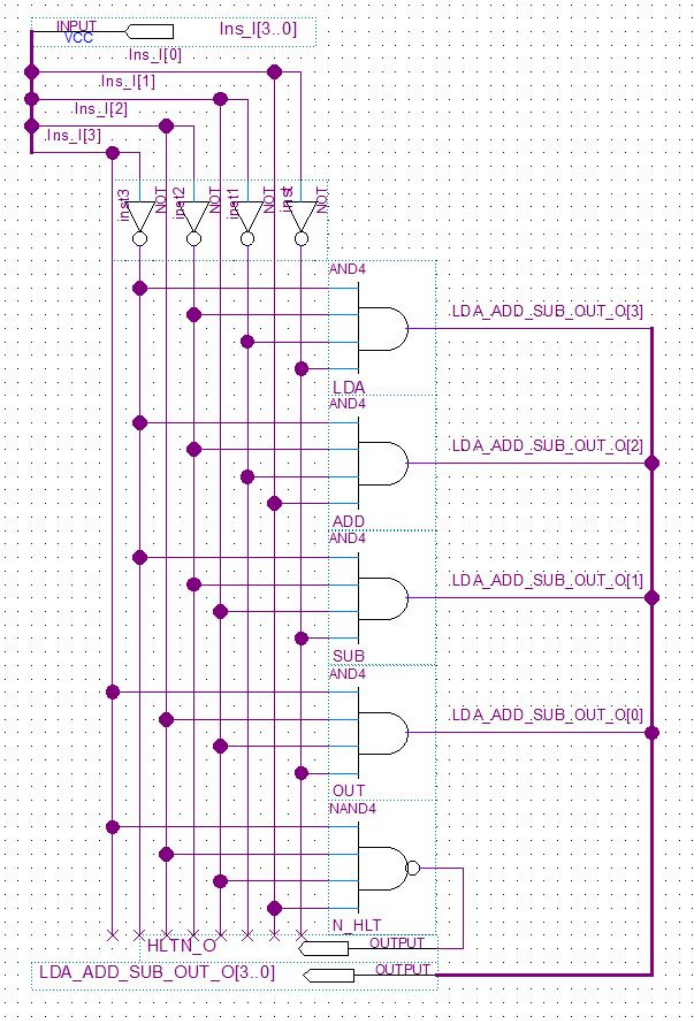


Kuva 10. Hallintasekvensserin pääpiiri, CSEQ.bdf (Malvino & Brown 1999: 158-161)

Taulukko 1 esittelee tietokoneen ohjejoukon, kuva 11 esittää ohjedekooderin piiriä, joka valitsee ohjerekisterin neljässä ylemmässä bitissä olevan ohjekoodin perusteella, minkä ohjeen signaali lähetetään eteenpäin. LDA, ADD, SUB ja OUT ohjeiden signaalit lähetetään hallintamatriisille. HLT ohjeen signaali käännetään komplementiksi ja lähetetään suoraan aikaisemmin esitellylle kellon hallinnan logiikalle HLTN signaalina. (Malvino & Brown 1999: 144-151, 158-161)

Taulukko 1. Tietokoneen ohjejoukko (Malvino & Brown 1999: 144-151)

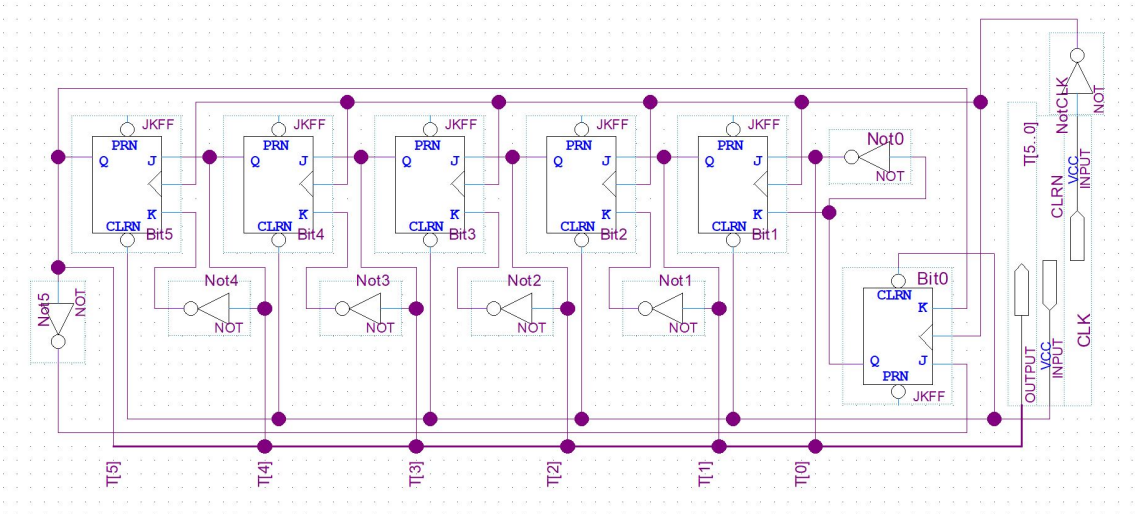
Ohje	Ohjekoodi	Toiminto
LDA ????	0000 ????	Siirtää ohjeen jälkeen annetun ???? muistiosoitteen sisällön A-rekisteriin.
ADD ????	0001 ????	Siirtää ohjeen jälkeen annetun ???? muistiosoitteen sisällön B-rekisteriin, ja lisää B-rekisterin arvon A-rekisterin arvoon.
SUB ????	0010 ????	Siirtää ohjeen jälkeen annetun ???? muistiosoitteen sisällön B-rekisteriin, ja vähentää B-rekisterin arvon A-rekisterin arvosta.
OUT XXXX	1110 XXXX	Siirtää A-rekisterin arvon tulosterekisteriin.
HLT XXXX	1111 XXXX	Pysäyttää tietokoneen ajon.



Kuva 11. Ohjedekooderin piiri, INSDCDR.bdf (Malvino & Brown 1999: 158-161)

Kuva 12 esittää kellosyklin rinkilaskurin piiriä. Se pitää kirjaa siitä, mikä tietokoneen kellosyklin kuudesta eri tilasta on menossa. CLRN signaalin saadessaan, se saa alkuarvokseen 000001. Jokaisen bitin JK-kytkimen ulostulo Q ja sen komplementti, on kytketty seuraavan bitin sisääntulohin J ja K, bittiä 0 lukuun ottamatta, joka on käänteinen muiden bittien JK-kytkimiin verrattuna. Tämä siitä syystä, että Malvinon ja Brownin kirja käytti rinkilaskurin rakentamiseen fyysisiä JK-kytkimiä, joissa ei ollut preset signaalia. Joten laskurin nollaaminen oikeaan lukuun 000001 vaati 0 bitin kääntämistä päinvastaiseksi. Tämä piiri toimisi samalla tavalla, jos bitti 0 käännettäisiin vastaamaan muita bittejä, ja sen CLRN signaali ohjattaisiin CLRN sisääntulon sijasta sen PRN sisääntuloon. (Malvino & Brown 1999: 114-116, 158-161)

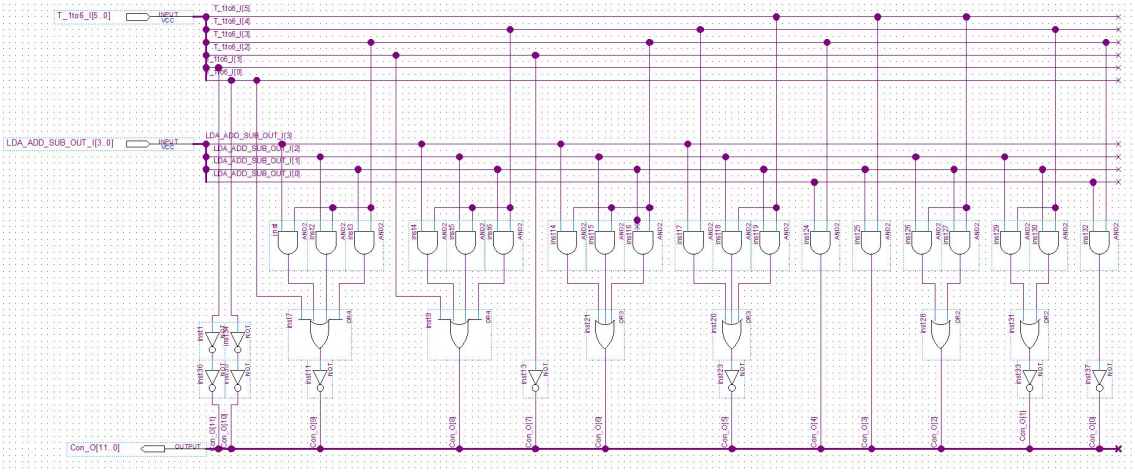
CLK kellosignaalin käydessä, laskurin bitit asettavat itseään seuraavan bitin arvon omaksi arvokseen. Tällöin laskuri saa vuorotellen arvot 000001, 000010, 000100, 001000, 010000 ja 100000, jonka jälkeen se palaa takaisin arvoon 000001. Laskurin sen hetkisen arvon voi lukea signaalista T[5..0]. (Malvino & Brown 1999: 114-116)



Kuva 12. Kellosyklin tilan rinkilaskurin piiri, OCNTR.bdf (Malvino & Brown 1999: 114-116, 158-161)

Kuvan 13 esittämän hallintamatriisin tehtävänä on orkestroida kellosyklin tilan ja sillä hetkellä ajettavan ohjeen perusteella datan siirtymistä tietokoneen muistin, laskureiden ja eri rekisterien välillä. Piiri generoi sisääntulevien T1_to_6_I[5..0] ja LDA_ADD_SUB_OUT_I[3..0] signaalien perusteella Con_O[11..0] hallintasana signaalin, joka ohjataan edelleen hallintasekvensserin Cword_O[11..0] ulostulo signaaliin.

Taulukko 2 esittää, mitkä datansiirrot tapahtuvat minkäkin hallintasanan bitin perusteella. Miinusmerkki, -, bitin toiminnon kuvauksen edessä tarkoittaa että kyseessä on komplementti, eli käänteinen hallintasignaali. Toteutettu matriisi poikkeaa hieman kirjan matriisista, koska osa hallintasignaaleista on tässä toteutuksessa käänteisiä kirjan hallintasignaaleihin verraatuna. (Malvino & Brown 1999: 141-153, 158-161)



Kuva 13. Hallintamatriisin piiri, CONMTRX.bdf (Malvino & Brown 1999: 158-161)

Taulukko 2. Komentosanan signaalien toiminnot. (Malvino & Brown 1999: 141-153, 160-161)

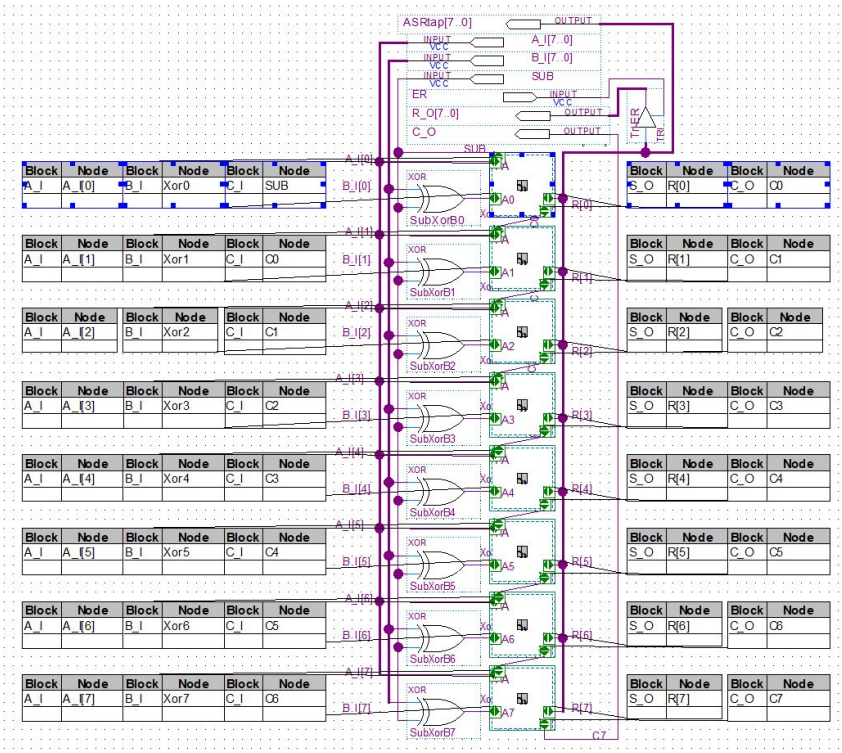
Komentosana	Signaalin toiminto
11	Askella ohjelmalaskuria
10	Siirrä ohjelmalaskurin arvo bussiin
9	-Siirrä bussin arvo muistiosoite rekisteriin
8	Siirrä muistin arvo bussiin
7	-Siirrä bussin arvo komentorekisteriin
6	Siirrä komentorekisterin arvo bussiin
5	-Siirrä bussin arvo A-rekisteriin
4	Siirrä A-rekisterin arvo bussiin
3	Aktivoi vähennyslaskumoodi
2	Siirrä yhteenlaskupiirin arvo bussiin
1	-Siirrä bussin arvo B-rekisteriin
0	-Siirrä bussin arvo tulosterekisteriin

2.5 Yhteen- ja vähennyslaskupiiri

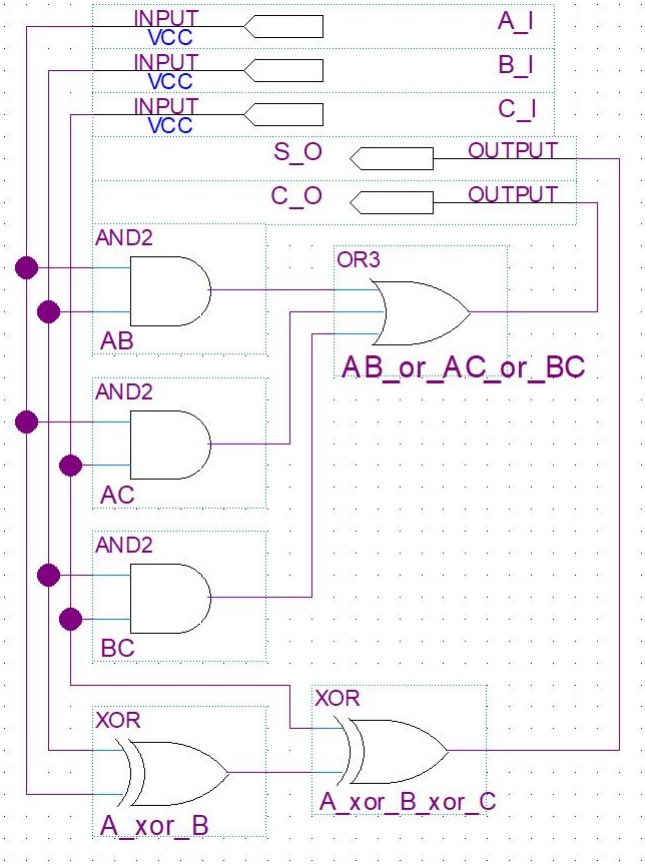
Kuva 14 esittää tietokoneen yhteen- ja vähennyslaskupiiriä. A_I[7..0] on A-rekisterin sisältö, B_I[7..0] on B rekisterin sisältö, SUB on vähennyslaskumoodin hallintasignaali, ER on laskun tuloksen kolmitilaulostulon hallintasignaali, R_O[7..0] on laskun tuloksen arvo, joka lähetetään bussiin ER signaalin hallitseman kolmitilaulostulon lävitse ja ASRtap[7..0] on laskun tuloksen reaaliaikainen signaali, joka ohjataan kehitysalustan ledvaloille koneen tilan seuraamista varten. C_O on ylijääneen muisti, eli carry-bitin arvo, jota ei tietokoneen tässä toteutuksessa käytetä mihinkään. (Malvino & Brown 1999: 81-87, 156, 158)

Piiri koostuu kahdeksasta kuvan 15 mukaisesta full-adder täysyhteenlaskupiiristä, joiden C_O carry-ulos, ja C_I carry-sisääntulot otkytketty toisiinsa. Yhteenlaskettavien A- ja B-rekisterien arvojen eri bitit on kytketty niitä vastaavien full-adder piirien A_I ja B_I sisääntuloihin. Ja yhteen- tai vähennyslaskun tulos rakennetaan ketjuttamalla full-adder piirien S_O summa ulostulot oikeaan järjestykseen. Vähennyslasku suoritetaan muuttamalla B_I[7..0] itsensä kahden komplementiksi. Tämä tapahtuu kääntämällä B_I[7..0] päinvastaiseksi laskemalla sen eri bittien ja SUB signaalin väliset XOR tulot, ja lisäämällä A_I[7..0]:n ja käänteisen B_I[7..0]:n yhteenlaskun tulokseen yksi, mikä tapahtuu ohjaamalla SUB signaalin arvo ensimmäisen full-adder piirin C_I carry-sisääntuloon. (Malvino & Brown 1999: 81-87, 156, 158)

Kuvan 15 full-adder piiri suorittaa kahden eri yhden bitin luvun yhteenlaskun, ottaen huomioon mahdollisen edellisestä bitistä saadun tai seuraavaan bittiin lähetettävän muistinumeron. Taulukko 3 esittää full-adder piirin totuustaulua. (Malvino & Brown 1999: 81)



Kuva 14. 8 bittinen kahden komplementin yhteen- vähennyslaskupiiri, ADDSUB8.bdf (Malvino & Brown 1999: 81-87, 156, 158)



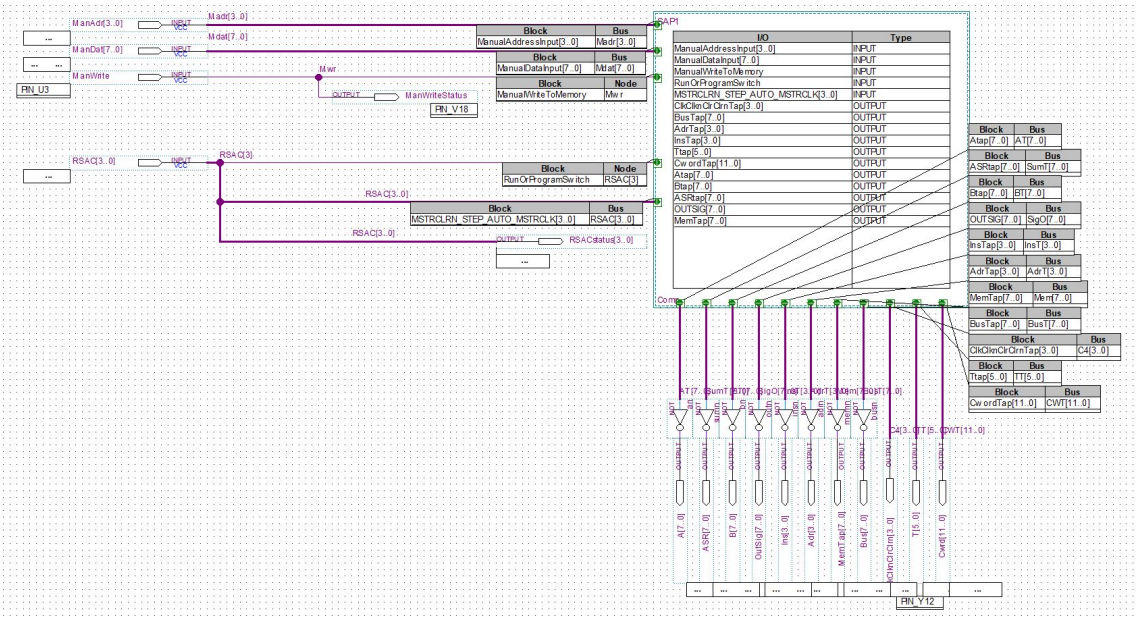
Kuva 15. Yhden bitin full adder laskupiiri, FA.bdf (Malvino & Brown 1999: 81)

Taulukko 3. Full-adder piirin totuustaulu. (Malvino & Brown 1999: 81)

A_I	**+B_I**	**+C_I**	**=C_O S_O**
0	0	0	0 0
0	0	1	0 1
0	1	0	0 1
0	1	1	1 0
1	0	0	0 1
1	0	1	1 0
1	1	0	1 0
1	1	1	1 1

2.6 Fyysisten syötteiden ja ulostulojen konfiguraatio

Kuva 16 esitää Quartus projektin ylimmän tason SAPSOCKET.bdf piiriä, joka sisältää vain instanssin tietokoneen aikaisemmin esitellystä SAP1.bdf pääpiiristä, sekä sen kytkennät kehitysalustan sisään ja ulostuloihin. Taulukko 4 kertoo mitä eri sisään- ja ulostulosignaalit tekevät. Kyseistä piiriä käytetään yksinkertaistamaan SAP1.bdf piirin kytkentöjen hallintaa. On helpompi määritellä piirin eri kytkentöjen yhteydet fyysiseen kehitysalustaan, kun vain sisään ja ulostulosignaalit ovat näkyvissä.



Kuva 16. Sisään ja ulostulojen signaalit, SAPSOCKET.bdf

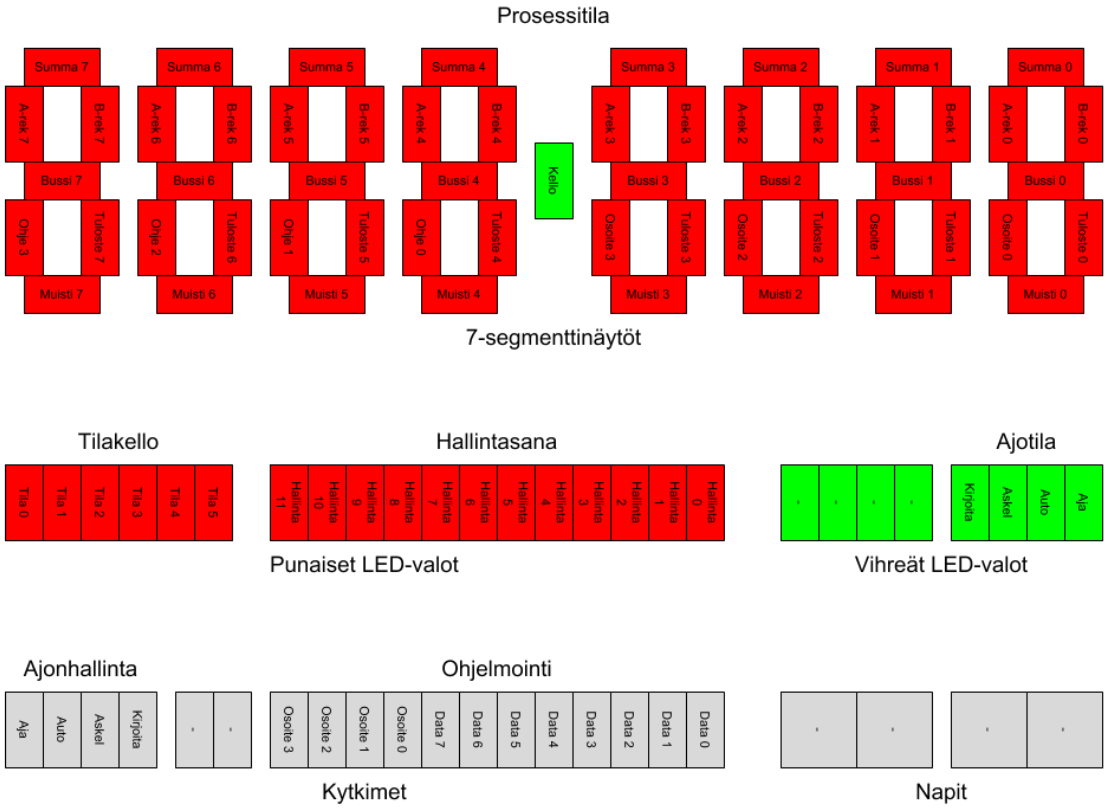
Taulukko 4. Tietokoneen sisään ja ulostulosignaalien nimet ja tarkoitukset

Signaali	Suunta	Tarkoitus
ManAdr[3..0]	Sisääntulo	Manuaalinen muistiosoitteen syöttö.
ManDat[7..0]	Sisääntulo	Manuaalinen datan syöttö.
ManWrite	Sisääntulo	Manuaalinen datan kirjoittaminen muistiosoitteeseen.
RSAC[3..0]	Sisääntulo	Ajonhallinnan signaalit. Lyhennelmä sanoista Run, Step, Auto, Clock. Run (Aja) signaali valitsee ajo ja ohjelmointitilojen välillä. Step (Askel) signaalilla askelletaan manuaalista kellosignaalia. Auto signaali määrittelee käytetäänkö manuaalista vai automaattista kelloa. Ja Clock (Kello) signaali on tietokoneen mestarikello, jota käytetään automaattisen kellosignaalin generointiin.
ManWriteStatus	Ulostulo	Manuaalisen kirjoitus-signaalin merkkivalon signaali.
RSACstatus[3..0]	Ulostulo	Ajonhallinnan signaalien merkkivalojen signaalit.
A[7..0]	Ulostulo	A-rekisterin sisällön signaali.
ASR[7..0]	Ulostulo	Yhteenlaskupiirin sisällön signaali.
B[7..0]	Ulostulo	B-rekisterin sisällön signaali.
OutSig[7..0]	Ulostulo	Tulosterekisterin sisällön signaali.
Ins[3..0]	Ulostulo	Ohjerekisterin sisällön alemmat neljä bittiä, jotka sisältävät sen hetkisen ohjekoodin.
Adr[3..0]	Ulostulo	Osoiterekisterin sisällön signaali.
MemTap[7..0]	Ulostulo	Osoiterekisterin määrittelemän muistisanan sisältö.
Bus[7..0]	Ulostulo	Bussin sisältö.
ClkClknClrClrn[3..0]	Ulostulo	Koneen ajokellon ja resetoinnin signaalit. Sekä niiden käänteis-signaalit. Hallintasekvensseri generoi ne RSAC signaalien perusteella. Vain Clk ja Clrn signaalit, eli ajokellon signaali ja resetoinnin käänteis-signaali, ovat käytössä.
T[5..0]	Ulostulo	Koneen kellosyklin tilakellon signaali.
Cwrd[11..0]	Ulostulo	Koneen komentosanan signaali.

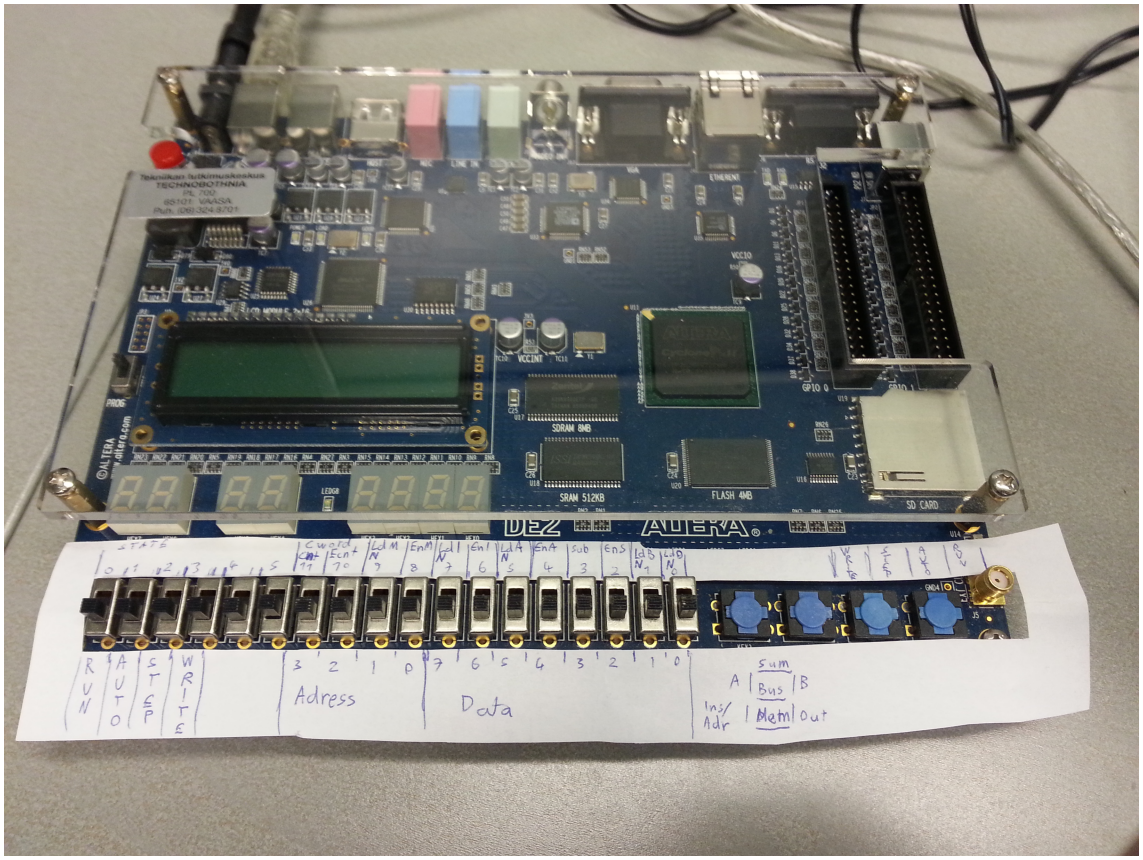
Tietokoneen muistin ohjelmoinnin (ManAdr[3..0], ManDat[7..0], ManWrite), sekä ajonhallinnan (RSAC[3..1]) sisääntulojen signaalit sidottiin DE2 kehitysalustan kytkimiin. Painonappeja vältettiin, koska osa niistä reagoi huonosti. Luultavasti aikaisemman

käytön aiheuttaman kulutuksen vuoksi. Tilakello (T[5..0]) ja hallintasana (Cwrd[11..0]) sidottiin kehitysalustan punaisiin LED-valoihin. Kellon (ClkClknClrClrn[3]) ja ajotilan (ManWriteStatus, RSACstatus[3..1]) signaalit sidottiin kehitysalustan vihreisiin LED-valoihin. Ja tietokoneen bussin (Bus[7..0]), muistin (MemTap[7..0]), laskupiirin (ASR[7..0]), sekä rekisterien (A[7..0], B[7..0], OutSig[7..0], Ins[3..0], Adr[3..0]) tilat sidottiin 7-segmenttinäyttöihin. Mestarikellon (RSAC[0]) signaali sidottiin kehitysalustan sisäiseen kelloon.

7-segmenttinäyttöjä kohdeltiin sarjana tavallisia LED-valoja, koska kehitysalustan ulostulot eivät olisi riittäneet kaikkien tietokoneen signaalien esittämiseen. Muuten ongelman kiertämiseksi olisi pitänyt ohjelmoida koneeseen jonkinlainen näytönhallintapiiri, joka olisi näyttänyt vuorotellen koneen bussin, muistin, sekä eri rekisterien arvoja, sekä dekooderi 8-bittisen kahden komplementtiluvun esittämiseen 7-segmenttinäytöllä. Tähän vaadittu työ olisi lisännyt testattavien piirien määrää, ja siten hidastanut ja hankaloittanut koneen testaamista kokonaisuutena. Kuvat 17 ja 18 esittävät miten tietokoneen eri signaalit on sidottu kehitysalustan sisään ja ulostuloihin, sekä niiden lukemisen helpottamiseen tarkoitetun nimilapun.



Kuva 17. Tietokoneen sisään- ja ulostulosignaalien, sekä kehitysalustan kytkimien ja valojen välille määritellyt yhteydet.



Kuva 18. Kehitysalusta, sekä kytkinten ja LED valojen lukemisen helpottamiseen tarkoitettu nimilappu.

2.6.1 Ajonhallinnan kytkimet

Kehitysalustan kytkimet 17-14 sidottiin ajonhallinnan kytkimiksi. Kytkin 17 vaihtaa koneen tilaa ohjelmointi- ja ajotilan välillä. Kytkin 16 valitsee, ajetaanko tietokoneen kelloa manuaalisesti, vai automaattisesti. Kytkin 15 askeltaa manuaalista kelloa. Ja kytkin 14 kirjoittaa datakytkimillä määritetyn sanan muistiosoite kytkimien osoittamaan muistiosoitteeseen, jos kone on ohjelmointitilassa.

2.6.2 Muistiosoite- ja datakytkimet

Kytkimiä 11-0 käytetään koneen muistin ohjelmointiin. Kytkimet 11-8 määrittelevät muistiosoitteen jota ollaan ohjelmoimassa. Kytkimillä 7-0 määritellään data, jota kyseiseen muistiosoitteeseen ollaan kirjoittamassa.

2.6.3 Ajotilan LED-valot

Kehitysalustan vihreistä LED valoista numerot 0-3, sekä 8 on sidottu tietokoneen ajotilan, sekä kellon seuraamiseen. LED-0 kertoo, onko kone ajo-, vai ohjelmointitilassa. Valaistu LED tarkoittaa ajotilaa. LED-1 kertoo, käyttääkö kone manuaalista vai automaattista kelloa. Valaistu LED osoittaa automaattisen kellon olevan aktiivisena. LED-2 on sidottu manuaalisen kellon tilaan. Se vastaa manuaalisen kellon kytkimen (kytkin 15) tilaa. LED-3 kertoo, onko kirjoituskytkin (kytkin 14) aktiivisena vai ei. Valaistu LED osoittaa että kirjoitussignaali on aktiivinen ja kone on kirjoittamassa datakytkinten tilaa muistiin, sillä edellytyksellä, että kone on ohjelmointi tilassa. LED-8 on sidottu koneen kellosignaaliin. Se vilkkuu koneen käynnin tahdissa.

2.6.4 Tilakellon ja komentosanan LED-valot

Kehitysalustan punaisista LED-valoista numerot 17-12 on sidottu tietokoneen tilakellon seuraamiseen. LED-17 vastaa tilakellon bittiä 0, eli kellosyklin ensimmäistä tilaa. LED-16 bittiä 1, eli tilaa 2, jne. LED-12 vastaa tietenkin bittiä 5, eli kellosyklin tilaa 6. Punaiset LED-valot 11-0 kertovat koneen sen hetkisen komentosanan tilan. Taulukko 5 esittää mikä LED-valo vastaa mitäkin komentosignaalia. Miinusmerkki, -, signaalin edessä kertoo kyseisen signaalin olevan käänteinen, eli komplementtisignaali.

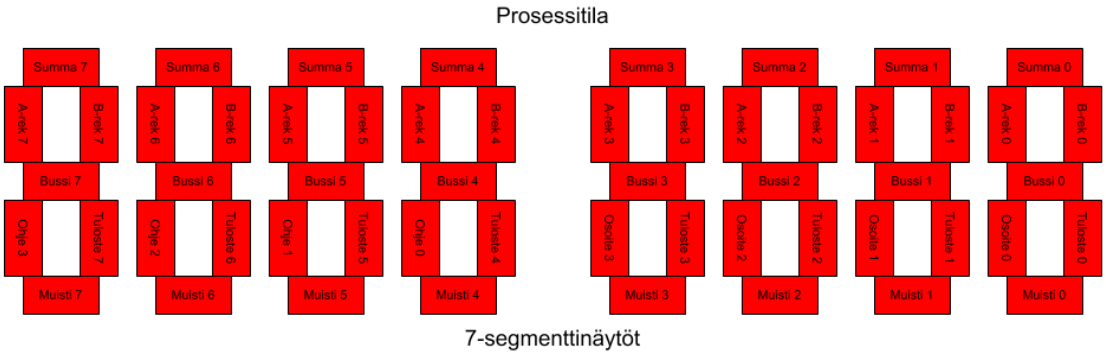
Taulukko 5. Komentosanan LED-valot.

LED-valo/ Komentosana	Signaalin toiminto
11	Askella ohjelmalaskuria
10	Siirrä ohjelmalaskurin arvo bussiin
9	-Siirrä bussin arvo muistiosoite rekisteriin
8	Siirrä muistin arvo bussiin
7	-Siirrä bussin arvo komentorekisteriin
6	Siirrä komentorekisterin arvo bussiin
5	-Siirrä bussin arvo A-rekisteriin
4	Siirrä A-rekisterin arvo bussiin
3	Aktivoi vähennyslaskumoodi
2	Siirrä yhteenlaskupiirin arvo bussiin
1	-Siirrä bussin arvo B-rekisteriin
0	-Siirrä bussin arvo tulosterekisteriin

2.6.5 Rekisterien, bussin ja laskupiirien valot

Rekisterien, bussin ja laskupiirien signaalit on sidottu kehitysalustan 7-segmenttinäyttöihin. Mahdollisten ulostulosignaalien määrän lisäämiseksi, sekä koneen rakenteen yksinkertaistamiseksi, 7-segmenttinäyttöjä kohdeltiin sarjana tavallisia LED-valoja numeronäyttöjen sijaan.

Koska kehitysalustassa oli 8 7-segmenttinäyttöä, niiden avulla oli mahdollista esittää kuusi eri 8-bittistä, ja kaksi eri 4-bittistä, binäärilukua lukemalla lukujen kannat näyttöjen perusteella ja kantojen bitit eri näyttöjen vastaavista segmenteistä. Kuva 19 esittää miten koneen eri signaalit on sidottu 7-segmenttinäyttöjen valoihin.



Kuva 19. 7-Segmenttinäyttöihin sidotut signaalit.

Ylimmistä vaakasegmenteistä voidaan lukea yhteenlaskupiirin hetkellinen arvo, keskimmäisestä vaakasegmenteistä bussin hetkellinen arvo ja alimmasta vaakasegmentistä valitun muistisanan hetkellinen arvo. Ylemmät pystysegmentit kertovat A- ja B-rekisterien arvot, A vasemmalla, B oikealla. Alempi oikea pystysegmentti kertoo tulosterekisterin arvon. Ja alempi vasen pystysegmentti on jaettu ohje- ja muistiosoiterekisterien kesken. Ohjerekisterin neljä ylempää bittiä, jotka sisältävät ohjekoodin, biteissä 7-4 ja muistiosoiterekisteri biteissä 3-0.

2.6.6 Esittämättömät arvot.

Ohjelmalaskurin arvoa, sekä ohjerekisterin neljän alemman bitin arvoja ei sidottu mihinkään ulostuloihin. Niiden arvot siirretään aina niitä käytettäessä bussiin, josta ne voidaan lukea tarvittaessa.

3 TIETOKONEEN TESTAAMINEN

Tietokoneen testaamiseen käytettiin Malvinon ja Brownin kirjasta lainattuun esimerkki-ohjelmaan, joka on esitetty taulukossa 6, taulukon X merkinnät esittävät muistipaikkoja, joidenka arvoilla ei ole väliä. (Malvino & Brown 1999: 145-146)

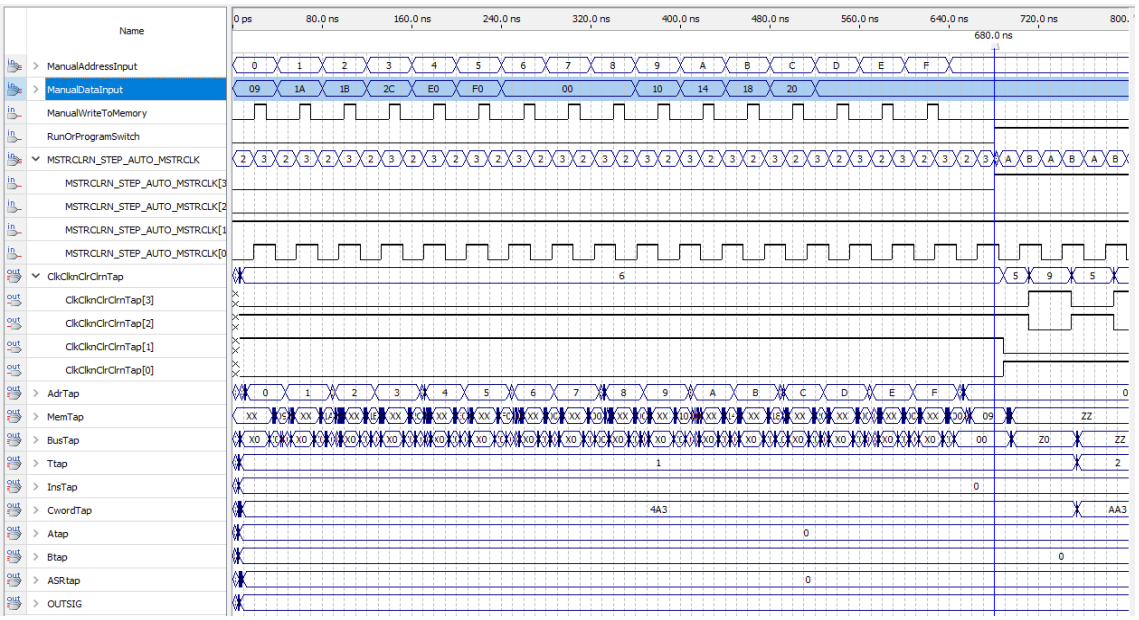
Ohjelma valittiin, koska se käyttää SAP-1 tietokoneen koko ohjejoukkoa, ja se oli jo valmiiksi esitetty kirjassa sekä memnoniikka, heksadesimaali- että binäärimuodossa. Kyseinen testiohjelma suorittaa laskutoimituksen 16+20+24-32, ja kirjoittaa sen ratkaisun, 28, tulosterekisteriin. (Malvino & Brown 1999: 145-146)

Taulukko 6. Testiohjelma: 16+20+24-32. (Malvino & Brown 1999: 145-146)

Muistiosoite (d/h/b)	Memnoninen ohje	Heksadesimaaliohje	Binääriohje
d0/h0/b0000	LDA h9	h09	b0000 1001
d1/h1/b0001	ADD hA	h1A	b0001 1010
d2/h2/b0010	ADD hB	h1B	b0001 1011
d3/h3/b0011	SUB hC	h2C	b0010 1100
d4/h4/b0100	OUT	hEX	b1110 XXXX
d5/h5/b0101	HLT	hFX	b1111 XXXX
d6/h6/b0110	XXX	hXX	bXXXX XXXX
d7/h7/b0111	XXX	hXX	bXXXX XXXX
d8/h8/b1000	XXX	hXX	bXXXX XXXX
d9/h9/b1001	h10 (d16)	h10	b0001 0000
d10/hA/b1010	h14 (d20)	h14	b0001 0100
d11/hB/b1011	h18 (d24)	h18	b0001 1000
d12/hC/b1100	h20 (d32)	h20	b0010 0000
d13/hD/b1101	XXX	hXX	bXXXX XXXX
d14/hE/b1110	XXX	hXX	bXXXX XXXX
d15/hF/b1111	XXX	hXX	bXXXX XXXX

Kuvat 20 ja 21 esittävät signaalikaavioita kyseisen ohjelman simuloidusta ohjelmoinnista ja ajosta kun tietokoneen SAP1.bdf piiri on Quartus projektin ylimmän tason piiri.

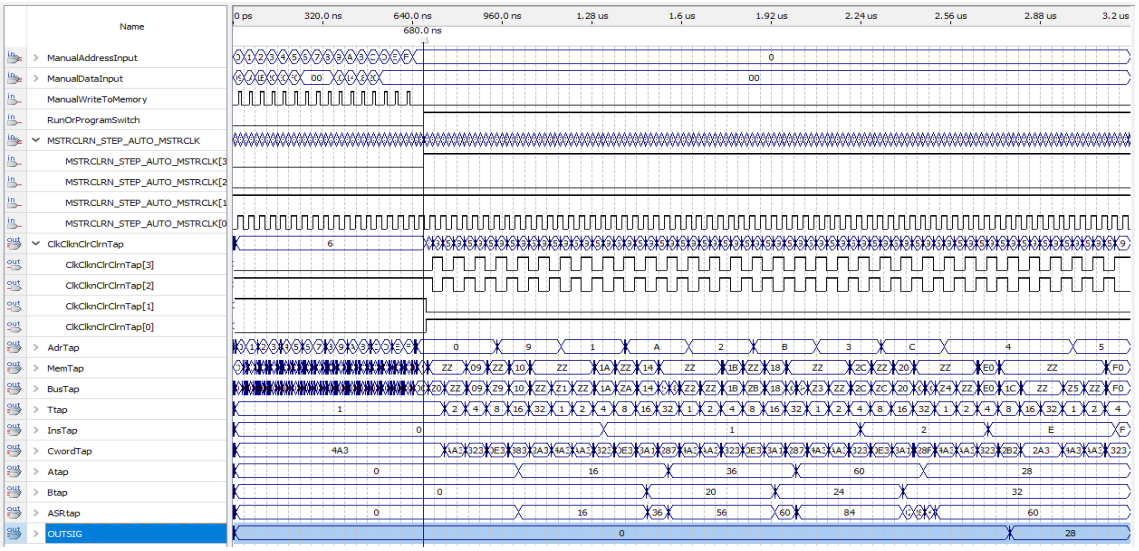
Kuvan 20 korostettu ManualDataInput rivi esittää ohjelman ohjelmointia ManualAdressInput rivin esittämiin muistiosoitteisiin. ManualWriteToMemory rivi sisältää ohjelman ohjeiden muistiin kirjoittamisen hallintasignaalin. RunOrProgramSwitch ja MSTRCLRN_STEP_AUTO_MSTRCLK signaalit sisältävät tietokoneen ajonhallinnan signaalit. Ohjelmoinnin aikana RunOrProgramSwitch ja MSTRCLRN_...[3] ovat matalalla, ja nousevat korkeiksi ohjelman ajon alkaessa ajanhetkellä 680 ns, ..._STEP_...[2] ja ..._AUTO..._[1] signaalit ovat koko ajon ajan vastaavasti matalalla ja korkealla, mikä tarkoitta että konetta ajetaan automaattisella ajokellolla. Ja ..._MSTRCLK[0] signaali sisältää koneen mestarikellon, johon sen automaattinen ajokello perustuu.



Kuva 20. Tietokoneen signaalikaavio testiohjelman simuloidulle ohjelmoinnille.

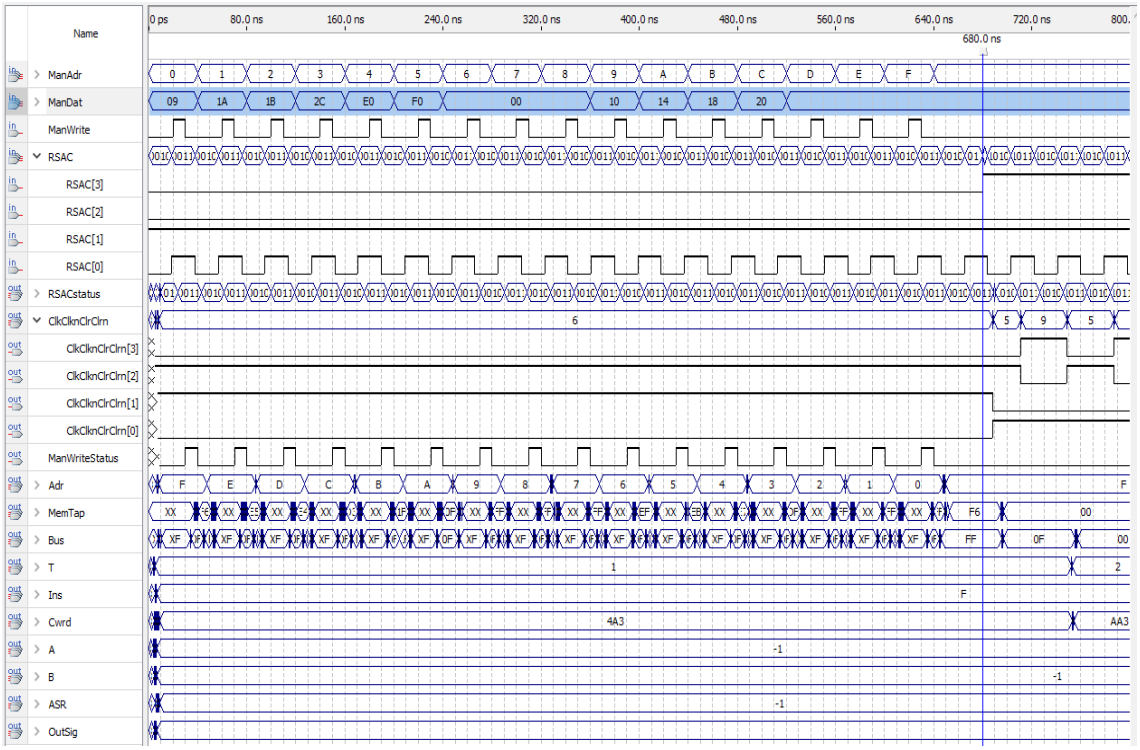
Kuva 21 esittää puolestaan ohjelman loppuun ajoa samassa simulaatiossa. Korostettu OUTSIG signaali sisältää tietokoneen tulosterekisterin sisällön ja simulaation lopussa voi nähdä sen sisältävän testiohjelman suorituksen tuloksen, 28. AdrTap, MemTap, BusTap, Ttap, InsTap, CwordTap, Atap, Btap ja ASRtap signaaleista voi vastaavasti lukea tietokoneen osoiterekisterin sisällön, valitun muistiosoitteen sisällön, bussin sisällön, kellosyklin tilakellon tilan, suoritettavana olevan ohjeen, suoritettavana olevan komentosanan, A-rekisterin sisällön, B-rekisterin sisällön, sekä yhteen- ja vähennyslaskupiirin sisällön kullakin ajanhetkellä. Huomaa myös tietokoneen ajokellon ClkClknClrClrn[3] pysähtyvän ajon lopussa kun ohjerekisterin neljä ylempää bittiä saa-

vat arvon hF, eli b1111. Mikä vastaa tietokoneen saamaa HLT komentoa. Kaikissa kuvissa pystyviiva ajanhetkellä 680 ns esittää tietokoneen moodin vaihtumista ohjelmointitilasta ajotilaan.

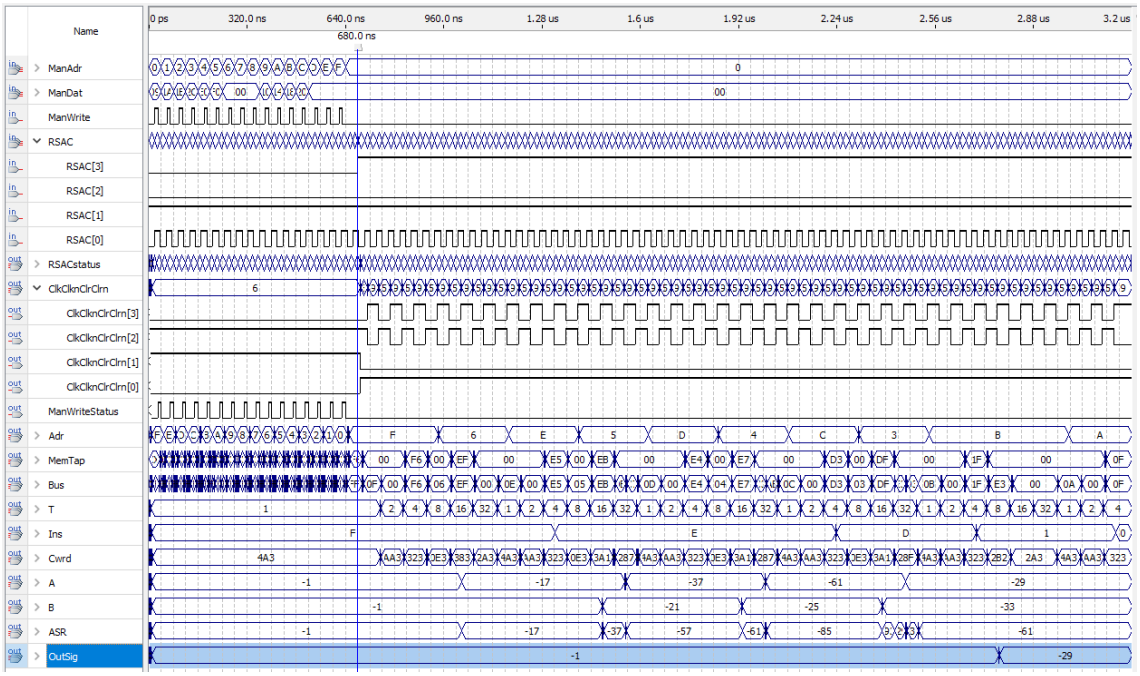


Kuva 21. Tietokoneen signaalikaavio testiohjelman simuloidulle ajolle.

Kuvat 22 ja 23 esittävät saman simulaation tulosta kun se suoritetaan SAPSOCKET.bdf piirin ollessa Quartus projektin ylimpänä tasona. Ainoat erot niiden välillä ovat että RunOrProgramSwitch ja MSTRCLRN_...[3] signaalit ovat yhdistetty samaan RSAC[3] signaaliin, hieman eroja käytetyissä signaalinimissä, RSACstatus ja ManWriteStatus signaalit jotka ovat vain ulostulokopioita vastaavista sisääntulosignaaleista, sekä se että Adr, MemTap, Bus, Ins, A, B, ASR ja OutSig signaalit ovat kuvien 20 ja 21 vastaavien signaalien komplementteja, koska ne tullaan ohjaamaan kehitysalustan seitsensegmenttinäyttöihin, joiden segmentit valaistuvat signaalin ollessa matalalla ja sammuvat sen ollessa korkealla. Huomaa kuvassa 23 korostetun OutSig signaalin lopullinen arvo -29, mikä on kokonaisluvun 28, eli testiohjelman suorittaman laskutoimituksen tuloksen, komplementti.



Kuva 22. Fyysisten sisään- ja ulostulojen signaalikaavio testiohjelman simuloidulle ohjelmoinnille.

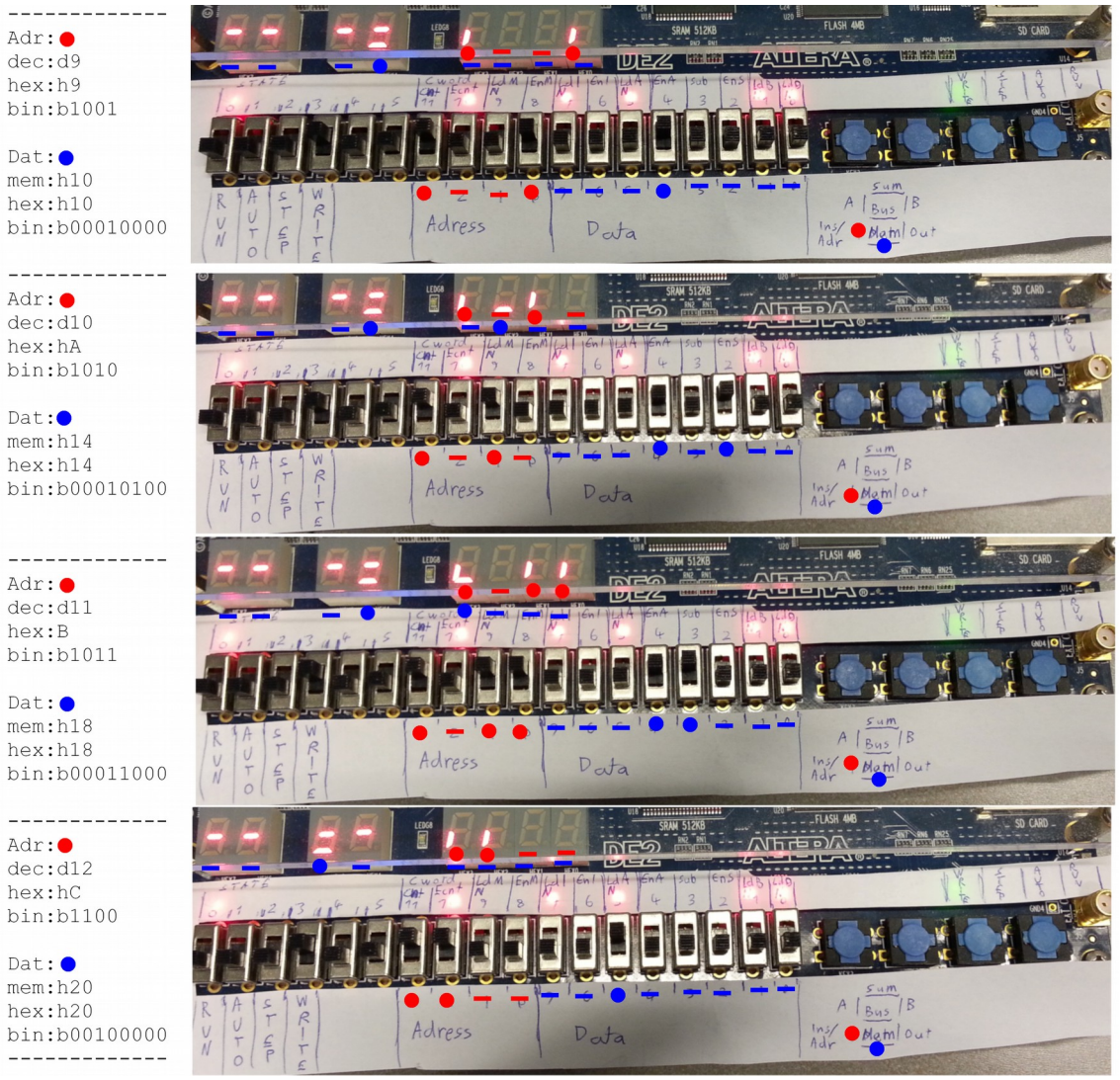


Kuva 23. Fyysisten sisään- ja ulostulojen signaalikaavio testiohjelman simuloidulle ajolle.

Kuva 24 ja 25 sisältävät kommentoidut valokuvat testiohjelman ohjelmoinnista kehitysalustalle, johon on ajettu sisään käännetty versio SAPSOCKET.bdf piirikaaviosta. Punaiset merkinnät esittävät ohjelmoitavan muistiosoitteen valintaa ja siniset merkinnät kyseiseen muistiosoitteeseen ohjelmoitavaa dataa. Kuvien vasemman reunan teksti sisältää vastaavan valokuvan muistiosoitteen ja siihen ohjelmoitavan datan arvot sekä memnonisessa, heksadesimaalisessa, että binäärisessä muodossa, huomaa kuinka ne vastaavat taulukon 6 sisältöä.

```
Ohjelma
-------------
Adr:●
dec:d0
hex:h0
bin:b0000

Dat:●
mem:LDA h9
hex:h09
bin:b00001001

-------------
Adr:●
dec:d1
hex:h1
bin:b0001

Dat:●
mem:ADD hA
hex:h1A
bin:b00011010

-------------
Adr:●
dec:d2
hex:h2
bin:b0010

Dat:●
mem:ADD hB
hex:h1B
bin:b00011011

-------------
Adr:●
dec:d3
hex:h3
bin:b0011

Dat:●
mem:SUB hC
hex:h2C
bin:b00101100

-------------
Adr:●
dec:d4
hex:h4
bin:b0100

Dat:●
mem:OUT
hex:hE0
bin:b11100000

-------------
Adr:●
dec:d5
hex:h5
bin:b0101

Dat:●
mem:HLT
hex:hF0
bin:b11110000

-------------
```

Kuva 24. Testiohjelman ohjeiden ohjelmointi testialustalle.



Kuva 25. Testiohjelman datan ohjelmointi testialustalle.

Kuva 26 esittää kuvissa 24 ja 25 ohjelmoidun testiohjelman ajon tulosta automaattisella ajokellolla. Ylemmän valokuvan viereinen teksti toistaa ohjelmoidun testiohjelman sisällön, ja siniset merkinnät sekä alemman kuvan viereinen teksti esittää ohjelman ajon tulosta, mikä sattuu olemaan desimaaliluku 28, eli testiohjelman suorittaman laskutoimituksen oikea tulos. Kuva 26 ei sisällä yhtäkään välivaihetta, koska tietokone laski oikean tuloksen niin nopeasti, että oikea tulos ilmestyi tulosterekisterin segmentteihin heti ajo kytkimen kytkemisen jälkeen.

```
Ohjelma:
---------------
h0:LDA h9( d16)
h1:ADD hA(+d20)
h2:ADD hB(+d24)
h3:SUB hC(-d32)
h4:OUT
h5:HLT
h9:h10(d16)
hA:h14(d20)
hB:h18(d24)
hC:h20(d32)
---------------
Tuloste:●
dec:d28
hex:h1C
bin:b0001 1100
```

Kuva 26. Ohjelmoidun testiohjelman tulos automaattisella kellolla ajettuna.

Kuvat 27-37 esittävät kuvissa 24 ja 25 ohjelmoidun testiohjelman ajamista manuaalisesti kytketyllä kellosignaalilla. Vihreät merkinnät esittävät tietokoneen prosessoinnin ajastusta, punaiset bussiin siirrettävää dataa tai ohjelmalaskurin askeltamista ja siniset bussista siirrettyä dataa. Seitsensegmenttinäyttöjen alla olevat pisteet vastaavat 1 bittejä ja viivat 0 bittejä. State rivin valot esittävät kellosyklin tilaa, Cword rivin valot komentosanan tilaa ja seitsensegmenttinäyttöjen valot nimilapun oikean alakulman piirroskaavion mukaisia rekistereiden, muistin, bussin ja laskupiirin sisältöjä. Vasemman reunan teksti esittää mitä ohjetta ollaan suorittamassa, mikä kellosyklin tila on menossa, mitä datan siirtoja sen tilan aikana ollaan tekemässä, sekä miten kyseiset siirrot vaikuttavat eri rekisterien, bussin, muistin ja yhteenlaskupiirin sisältöihin.

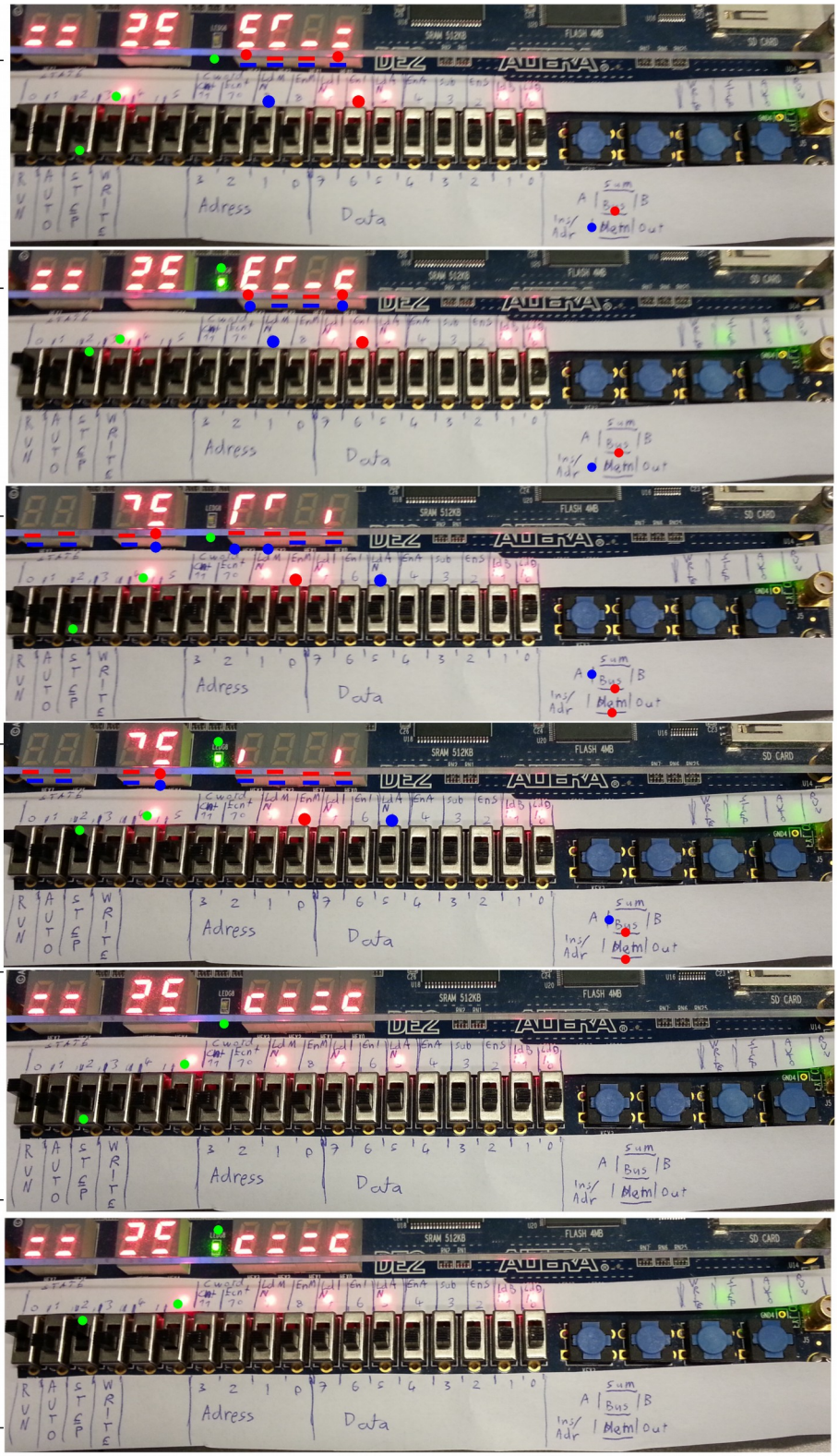
Kuvat 27 ja 28 esittävät "LDA h9" komennon suorittamista. Kellosyklin tilan 0 aikana ohjelmalaskurin arvo "b0000" siirretään muistiosoiterekisteriin, tilan 1 aikana ohjelmalaskuria askelletaan yhdellä, tilan 2 aikana muistiosoiterekisterin arvon "b0000" mukaisen muistiosoitteen sisältö, eli, komento "LDA h9", siirretään ohjerekisteriin, tilan 3 aikana ohjerekisterin alempien neljän bitin arvo, eli "h9" siirretään muistiosoiterekisteriin, tilan 4 aikana muistiosoiterekisterin arvon "h9" mukaisen muistiosoitteen sisältö, eli numero "d16" siirretään A-rekisteriin, ja tilan 5 aikana ei tehdä mitään. Kuvassa 27 esitettyjä tiloja 0-2, joiden aikana suoritettava ohje haetaan ohjelmalaskurista ja ohjelmalas-

kuria askelletaan, kutsutaan fetch-, eli hakusykliksi. Kun taas kuvan 28 tilojen 3-5 aikana ohje "LDA h9" varsinaisesti suoritetaan. (Malvino & Brown 1999: 142-152)

```
Manuaalinen Ajo
LDA h9,fetch-sykli
------------------
T0Low:●
Komentosana:
Pcntr->Bus ●
Bus   ->MAR ●

Tila:
Pcntr:bXXXX 0000
Bus   :bXXXX 0000●
MAR   :bXXXX 0101●
------------------
T0High:●
Komentosana:
Pcntr->Bus ●
Bus   ->Mar ●

Tila:
Pcntr:bXXXX 0000
Bus   :bXXXX 0000●
MAR   :bXXXX 0000●
------------------
T1Low:●
Komentosana:
Pcntr+1●

Tila:
Pcntr:bXXXX 0001
------------------
T1High:●
Komentosana:
Pcntr+1●

Tila:
Pcntr:bXXXX 0001
------------------
T2Low:●
Komentosana:
RAM   ->Bus  ●
Bus   ->Ireg ●

Tila:
RAM   :b0000 1001●
Bus   :b0000 1001●
Ireg  :b0000 ????●
------------------
T2High:●
Komentosana:
RAM   ->Bus  ●
Bus   ->Ireg ●

Tila:
RAM   :b0000 1001●
Bus   :b0000 1001●
Ireg  :b0000 1001●
------------------
```

Kuva 27. Testiohjelman manuaalinen ajo, komennon LDA h9 fetch-sykli.

```
Manuaalinen Ajo
LDA h9,LDA-sykli
------------------
T3Low: ●
Komentosana:
IregL->Bus ●
Bus  ->MAR ●

Tila:
IregL:bXXXX 1001
Bus  :bXXXX 1001 ●
MAR  :bXXXX 0000 ●
------------------
T3High:●
Komentosana:
IregL->Bus ●
Bus  ->MAR ●

Tila:
IregL:bXXXX 1001
Bus  :bXXXX 1001 ●
MAR  :bXXXX 1001 ●
------------------
T4Low: ●
Komentosana:
RAM  ->Bus ●
Bus  ->Areg ●

Tila:
RAM  :b0001 0000 ●
Bus  :b0001 0000 ●
Areg :b0001 1100 ●
------------------
T4High:●
Komentosana:
RAM  ->Bus ●
Bus  ->Areg ●

Tila:
RAM  :b0001 0000 ●
Bus  :b0001 0000 ●
Areg :b0001 0000 ●
------------------
T5Low: ●
Komentosana:
-

Tila:
-
------------------
T5High:●
Komentosana:
-

Tila:
-
------------------
```



Kuva 28. Testiohjelman manuaalinen ajo, komennon LDA h9 LDA-sykli.

Kuvat 29 ja 30 esittävät "ADD hA" komennon suorittamista. Kuva 29 esittää kyseisen komennon fetch-sykliä, ja vastaa siten kuvaa 27 eri ohjelmalaskurin ja muistissa olevan ohjeen arvoilla. Kuvassa 30 suoritetaan varsinainen "ADD hA" komento. Tilan 3 aikana ohjerekisterin alemmissa neljässä bitissä sijaitseva muistiosoite "hA" siirretään muistiosoite rekisteriin, tilassa 4 kyseisen muistiosoitteen sisältö, luku "d20", siirretään B-rekisteriin, ja tilassa 5 yhteenlaskupiirissä oleva A- ja B- rekisterien summa siirretään A-rekisteriin. (Malvino & Brown 1999: 142-152)

Kuvien 31 ja 32 esittämä "ADD hB" komento toimii vastaavasti. Vain ohjelmalaskurin arvolla, sekä muistissa olevilla komennolla ja datalla on eri arvot. Samoin kuvien 33 ja 34 esittämä "SUB hC" komento joka poikkeaa normaalista ADD komennosta vain siten että tilan 5 aikana komentosana sisältää yhteenlaskupiirin vähennyslaskumoodin hallintasignaalin, sen sisällön bussiin siirron hallintasignaalin lisäksi. (Malvino & Brown 1999: 142-152)

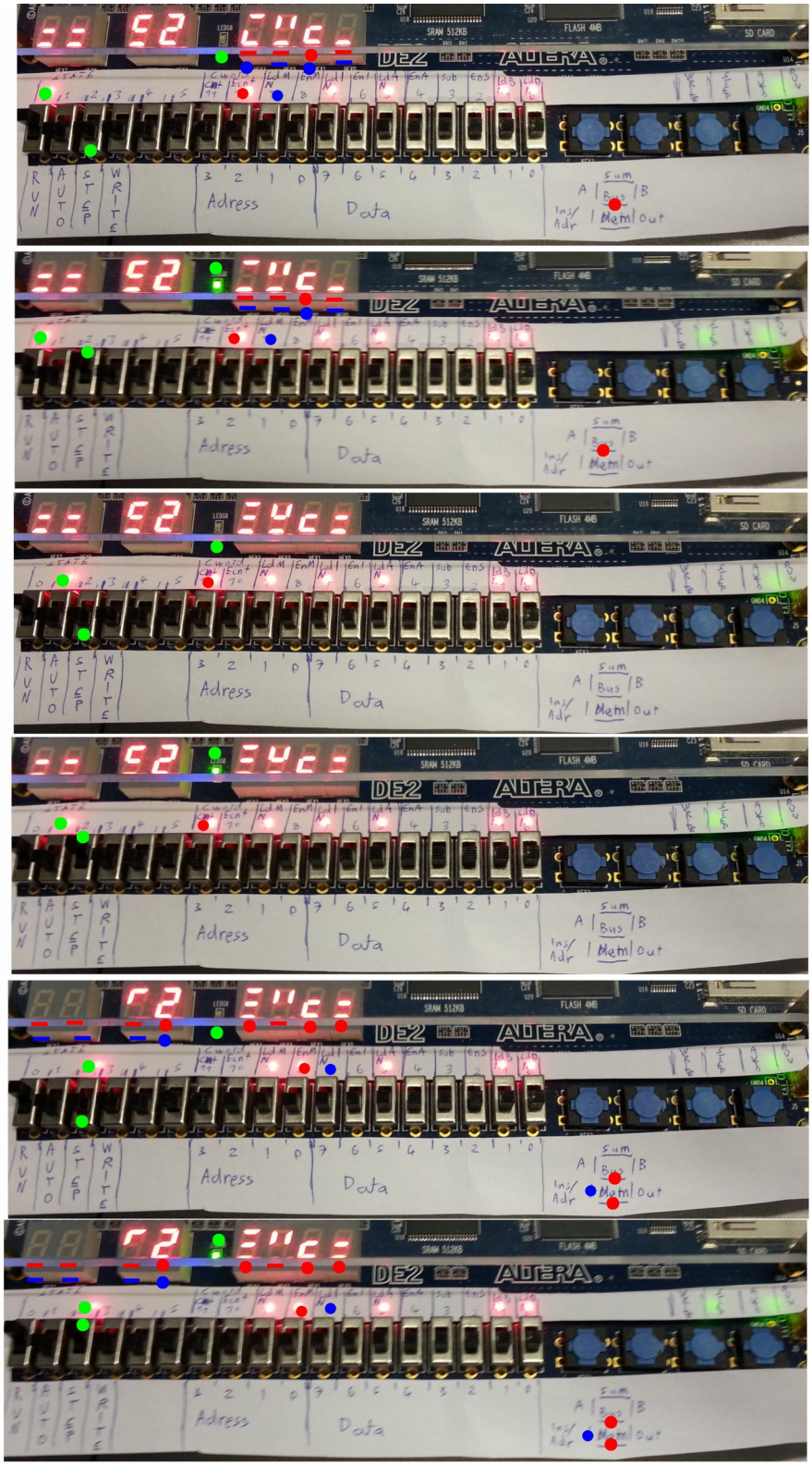
```
Manuaalinen Ajo
ADD hA,fetch-sykli
------------------
T0Low:●
Komentosana:
Pcntr->Bus ●
Bus  ->MAR ●

Tila:
Pcntr:bXXXX 0001
Bus  :bXXXX 0001●
MAR  :bXXXX 1001●
------------------
T0High:●
Komentosana:
Pcntr->Bus ●
Bus  ->Mar ●

Tila:
Pcntr:bXXXX 0001
Bus  :bXXXX 0001●
MAR  :bXXXX 0001●
------------------
T1Low:●
Komentosana:
Pcntr+1●


Tila:
Pcntr:bXXXX 0010


------------------
T1High:●
Komentosana:
Pcntr+1●


Tila:
Pcntr:bXXXX 0010


------------------
T2Low:●
Komentosana:
RAM  ->Bus ●
Bus  ->Ireg ●

Tila:
RAM  :b0001 1010●
Bus  :b0001 1010●
Ireg :b0000 ????●
------------------
T2High:●
Komentosana:
RAM  ->Bus ●
Bus  ->Ireg ●

Tila:
RAM  :b0001 1010●
Bus  :b0001 1010●
Ireg :b0001 1010●
------------------
```

Kuva 29. Testiohjelman manuaalinen ajo, komennon ADD hA fetch-sykli.

```
Manuaalinen Ajo
ADD hA,ADD-sykli
------------------
T3Low: ●
Komentosana:
IregL->Bus ●
Bus  ->MAR ●

Tila:
IregL:bXXXX 1010
Bus  :bXXXX 1010 ●
MAR  :bXXXX 0001 ●
------------------
T3High:●
Komentosana:
IregL->Bus ●
Bus  ->MAR ●

Tila:
IregL:bXXXX 1010
Bus  :bXXXX 1010 ●
MAR  :bXXXX 1010 ●
------------------
T4Low: ●
Komentosana:
RAM  ->Bus ●
Bus  ->Breg ●

Tila:
RAM  :b0001 0100 ●
Bus  :b0001 0100 ●
Breg :b0010 0000 ●
------------------
T4High:●
Komentosana:
RAM  ->Bus ●
Bus  ->Breg ●

Tila:
RAM  :b0001 0100 ●
Bus  :b0001 0100 ●
Breg :b0001 0100 ●
------------------
T5Low: ●
Komentosana:
Sum  ->Bus ●
Bus  ->Areg ●

Tila:
Sum  :b0010 0100 ●
Bus  :b0010 0100 ●
Areg :b0001 0000 ●
------------------
T5High:●
Komentosana:
Sum  ->Bus ●
Bus  ->Areg ●

Tila:
Sum  :b0011 1000 ●
Bus  :b0011 1000 ●
Areg :b0010 0100 ●
------------------
```

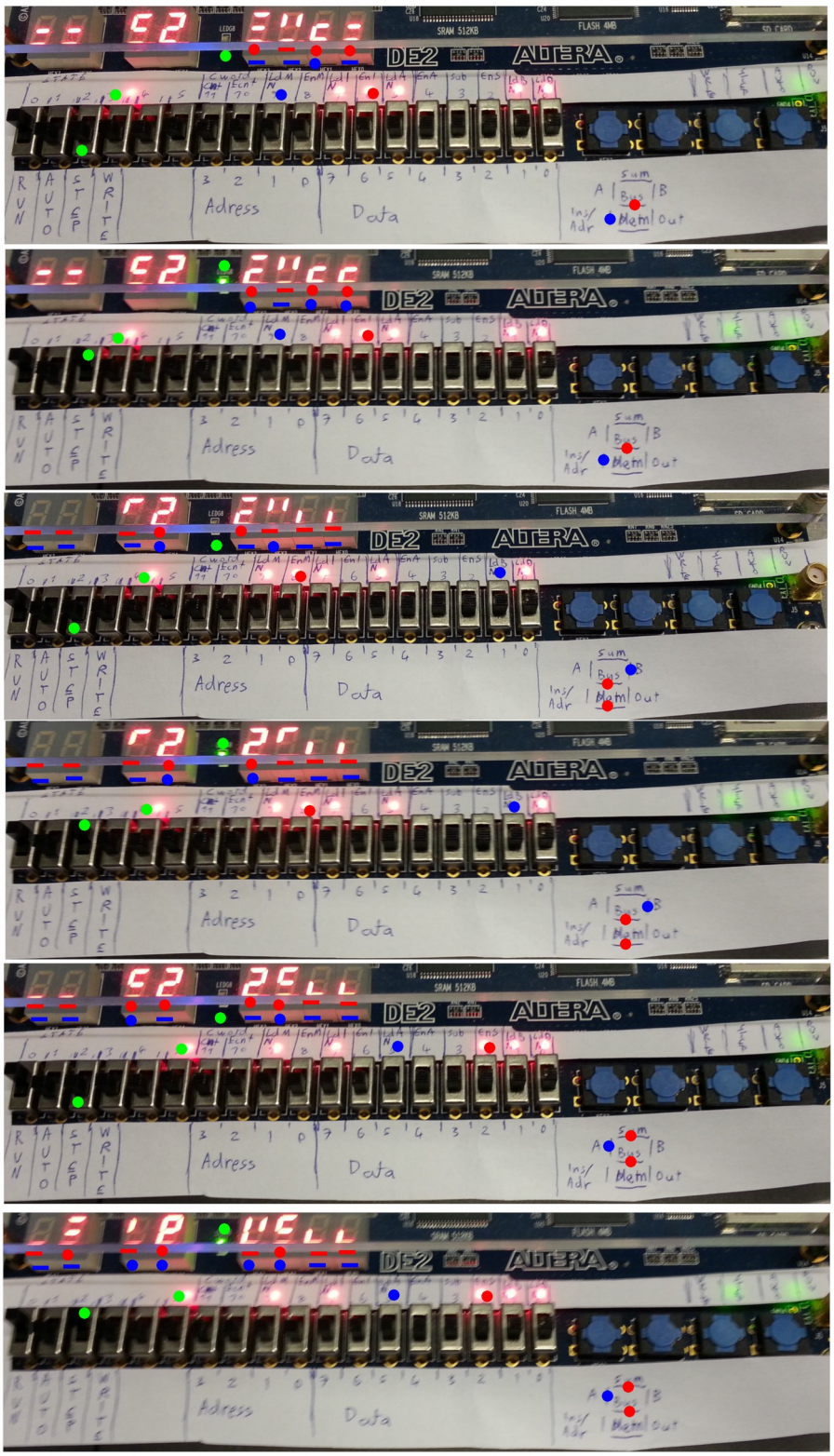
Kuva 30. Testiohjelman manuaalinen ajo, komennon ADD hA ADD-sykli.

```
Manuaalinen Ajo
ADD hB,fetch-sykli
------------------
T0Low:●
Komentosana:
Pcntr->Bus ●
Bus  ->MAR ●

Tila:
Pcntr:bXXXX 0010
Bus  :bXXXX 0010●
MAR  :bXXXX 1010●
------------------
T0High:●
Komentosana:
Pcntr->Bus ●
Bus  ->Mar ●

Tila:
Pcntr:bXXXX 0010
Bus  :bXXXX 0010●
MAR  :bXXXX 0010●
------------------
T1Low:●
Komentosana:
Pcntr+1●


Tila:
Pcntr:bXXXX 0011


------------------
T1High:●
Komentosana:
Pcntr+1●


Tila:
Pcntr:bXXXX 0011


------------------
T2Low:●
Komentosana:
RAM  ->Bus ●
Bus  ->Ireg ●

Tila:
RAM  :b0001 1011●
Bus  :b0001 1011●
Ireg :b0001 ????●
------------------
T2High:●
Komentosana:
RAM  ->Bus ●
Bus  ->Ireg ●

Tila:
RAM  :b0001 1011●
Bus  :b0001 1011●
Ireg :b0001 1011●
------------------
```



Kuva 31. Testiohjelman manuaalinen ajo, komennon ADD hB fetch-sykli.

```
Manuaalinen Ajo
ADD hB,ADD-sykli
------------------
T3Low: ●
Komentosana:
IregL->Bus ●
Bus  ->MAR ●

Tila:
IregL:bXXXX 1011
Bus  :bXXXX 1011 ●
MAR  :bXXXX 0010 ●
------------------
T3High:●
Komentosana:
IregL->Bus ●
Bus  ->MAR ●

Tila:
IregL:bXXXX 1011
Bus  :bXXXX 1011 ●
MAR  :bXXXX 1011 ●
------------------
T4Low: ●
Komentosana:
RAM  ->Bus ●
Bus  ->Breg ●

Tila:
RAM  :b0001 1000 ●
Bus  :b0001 1000 ●
Breg :b0001 0100 ●
------------------
T4High:●
Komentosana:
RAM  ->Bus ●
Bus  ->Breg ●

Tila:
RAM  :b0001 1000 ●
Bus  :b0001 1000 ●
Breg :b0001 1000 ●
------------------
T5Low: ●
Komentosana:
Sum  ->Bus ●
Bus  ->Areg ●

Tila:
Sum  :b0011 1100 ●
Bus  :b0011 1100 ●
Areg :b0010 0100 ●
------------------
T5High:●
Komentosana:
Sum  ->Bus ●
Bus  ->Areg ●

Tila:
Sum  :b0101 0100 ●
Bus  :b0101 0100 ●
Areg :b0011 1100 ●
------------------
```



Kuva 32. Testiohjelman manuaalinen ajo, komennon ADD hB ADD-sykli.

```
Manuaalinen Ajo
SUB hC,fetch-sykli
------------------
T0Low:●
Komentosana:
Pcntr->Bus ●
Bus  ->MAR ●

Tila:
Pcntr:bXXXX 0011
Bus  :bXXXX 0011●
MAR  :bXXXX 1011●
------------------
T0High:●
Komentosana:
Pcntr->Bus ●
Bus  ->Mar ●

Tila:
Pcntr:bXXXX 0011
Bus  :bXXXX 0011●
MAR  :bXXXX 0011●
------------------
T1Low:●
Komentosana:
Pcntr+1●


Tila:
Pcntr:bXXXX 0100


------------------
T1High:●
Komentosana:
Pcntr+1●


Tila:
Pcntr:bXXXX 0100


------------------
T2Low:●
Komentosana:
RAM  ->Bus ●
Bus  ->Ireg ●

Tila:
RAM  :b0010 1100●
Bus  :b0010 1100●
Ireg :b0001 ????●
------------------
T2High:●
Komentosana:
RAM  ->Bus ●
Bus  ->Ireg ●

Tila:
RAM  :b0010 1100●
Bus  :b0010 1100●
Ireg :b0010 1100●
------------------
```

Kuva 33. Testiohjelman manuaalinen ajo, komennon SUB hC fetch-sykli.

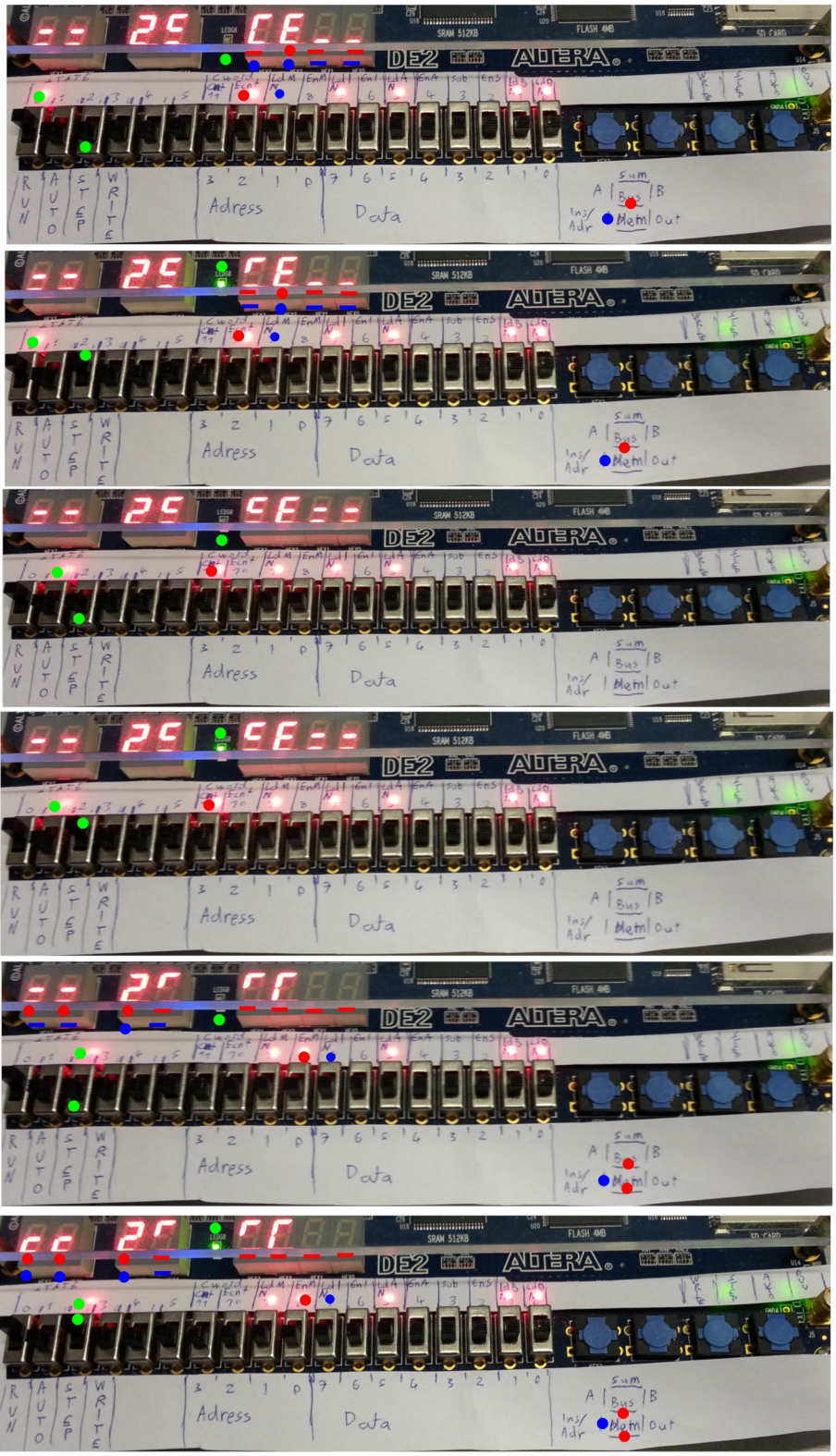
```
Manuaalinen Ajo
SUB hC,SUB-sykli
------------------
T3Low: ●
Komentosana:
IregL->Bus ●
Bus  ->MAR ●

Tila:
IregL:bXXXX 1100
Bus  :bXXXX 1100 ●
MAR  :bXXXX 0011 ●
------------------
T3High:●
Komentosana:
IregL->Bus ●
Bus  ->MAR ●

Tila:
IregL:bXXXX 1100
Bus  :bXXXX 1100 ●
MAR  :bXXXX 1100 ●
------------------
T4Low: ●
Komentosana:
RAM  ->Bus ●
Bus  ->Breg ●

Tila:
RAM  :b0010 0000 ●
Bus  :b0010 0000 ●
Breg :b0001 1000 ●
------------------
T4High:●
Komentosana:
RAM  ->Bus ●
Bus  ->Breg ●

Tila:
RAM  :b0010 0000 ●
Bus  :b0010 0000 ●
Breg :b0010 0000 ●
------------------
T5Low: ●
Komentosana:
Sub  Enable●
Sum  ->Bus ●
Bus  ->Areg ●
Tila:
Sum  :b0001 1100 ●
Bus  :b0001 1100 ●
Areg :b0011 1100 ●
------------------
T5High:●
Komentosana:
Sub  Enable●
Sum  ->Bus ●
Bus  ->Areg ●
Tila:
Sum  :b1111 1100 ●
Bus  :b1111 1100 ●
Areg :b0001 1100 ●
------------------
```

Kuva 34. Testiohjelman manuaalinen ajo, komennon SUB hC SUB-sykli.

Kuvat 35 ja 36 esittävät puolestaan "OUT" komennon suorittamista. Kuva 35 on jälleen kerran normaali fetch-sykli eri ohjelmalaskurin ja muistin sisällön arvoilla. Kuvassa 36 suoritetaan varsinainen "OUT" komento. Tilan 3 aikana A-rekisterin sisältö siirretään tulosterekisteriin ja tilojen 4 ja 5 aikoina ei tehdä mitään. Tulosterekisterin lopulliseksi arvoksi tulee "b0001 1100", eli "d28", mikä on jälleen kerran testiohjelman suorittaman laskutoimituksen oikea vastaus. (Malvino & Brown 1999: 142-152)

Kuva 37 esittää "HLT" komennon suorittamista. Se alkaa jälleen kerran normaalilla fetch-syklillä, mutta tällä kertaa tietokoneen kello pysähtyy heti tilassa 2, kun muistissa oleva "HLT" komento "b1111 XXXX" siirtyy ohjerekisteriin, ja sitä kautta ohjedekoo-deriin, mikä välittömästi HLT ohjeen saatuaan katkaisee sekä manuaalisen, että mestari-kellon yhteydet tietokoneen ajokelloon. (Malvino & Brown 1999: 142-152, 158-159)

```
Manuaalinen Ajo
OUT,fetch-sykli
------------------
T0Low:●
Komentosana:
Pcntr->Bus ●
Bus  ->MAR ●

Tila:
Pcntr:bXXXX 0100
Bus  :bXXXX 0100●
MAR  :bXXXX 1100●
------------------
T0High:●
Komentosana:
Pcntr->Bus ●
Bus  ->MAR ●

Tila:
Pcntr:bXXXX 0100
Bus  :bXXXX 0100●
MAR  :bXXXX 0100●
------------------
T1Low:●
Komentosana:
Pcntr+1●


Tila:
Pcntr:bXXXX 0101


------------------
T1High:●
Komentosana:
Pcntr+1●


Tila:
Pcntr:bXXXX 0101


------------------
T2Low:●
Komentosana:
RAM  ->Bus ●
Bus  ->Ireg ●

Tila:
RAM  :b1110 0000●
Bus  :b1110 0000●
Ireg :b0010 ????●
------------------
T2High:●
Komentosana:
RAM  ->Bus ●
Bus  ->Ireg ●

Tila:
RAM  :b1110 0000●
Bus  :b1110 0000●
Ireg :b1110 0000●
------------------
```

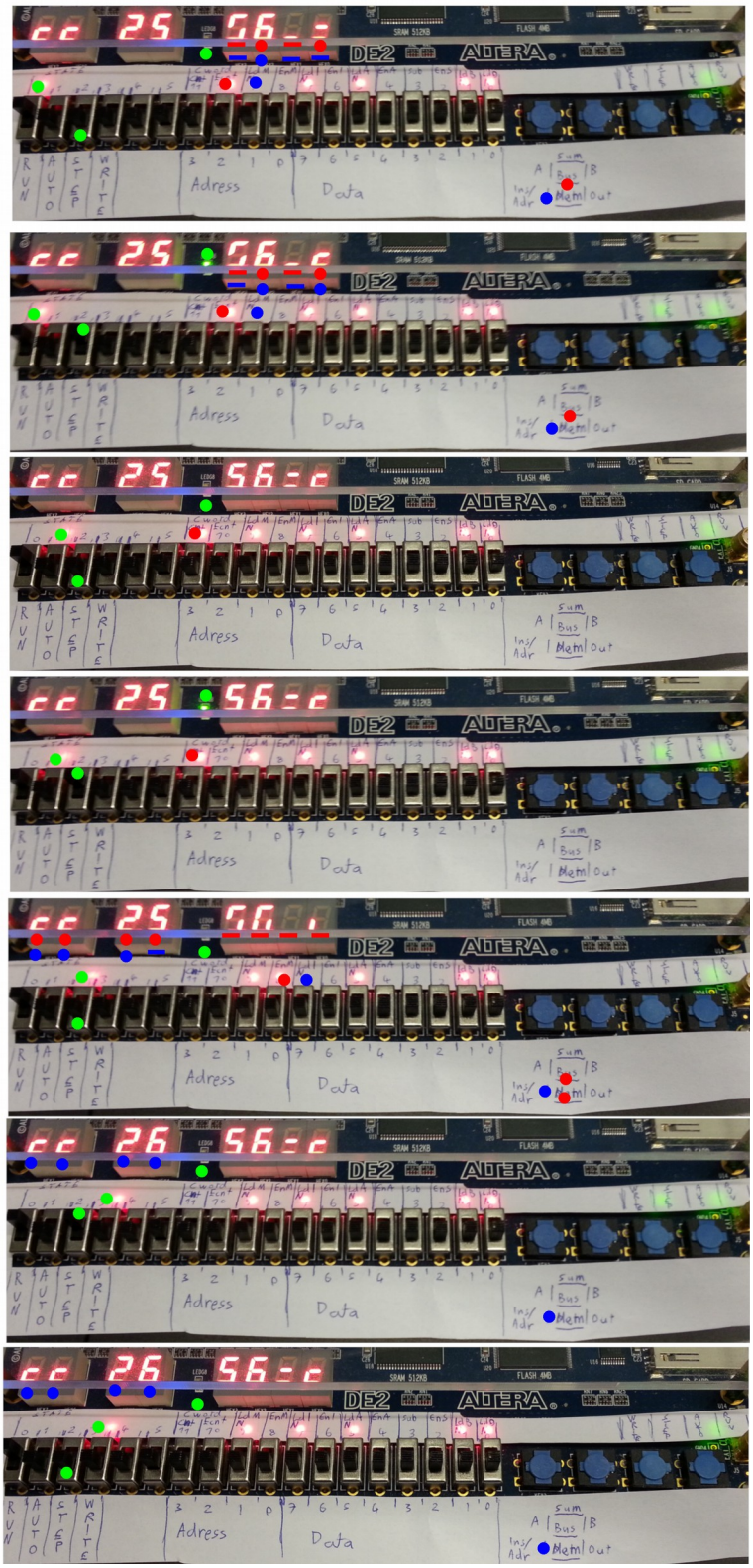


Kuva 35. Testiohjelman manuaalinen ajo, komennon OUT fetch-sykli.

```
Manuaalinen Ajo
OUT,OUT-sykli
------------------
T3Low:●
Komentosana:
Areg ->Bus  ●
Bus  ->Oreg ●

Tila:
Areg :b0001 1100 ●
Bus  :b0001 1100 ●
Oreg :b0000 0000 ●
------------------
T3High:●
Komentosana:
Areg ->Bus  ●
Bus  ->Oreg ●

Tila:
Areg :b0001 1100 ●
Bus  :b0001 1100 ●
Oreg :b0001 1100 ●
------------------
T4Low:●
-

Tila:
-
------------------
T4High:●
Komentosana:
-

Tila:
-
------------------
T5Low:●
Komentosana:
-

Tila:
-
------------------
T5High:●
Komentosana:
-

Tila:
-
------------------
```

Kuva 36. Testiohjelman manuaalinen ajo, komennon OUT OUT-sykli.

```
Manuaalinen Ajo
HLT
------------------
T0Low:  ●
Komentosana:
Pcntr->Bus ●
Bus  ->MAR ●
Tila:
Pcntr:bXXXX 0101
Bus  :bXXXX 0101 ●
MAR  :bXXXX 0100 ●
------------------
T0High:●
Komentosana:
Pcntr->Bus ●
Bus  ->Mar ●
Tila:
Pcntr:bXXXX 0101
Bus  :bXXXX 0101●
MAR  :bXXXX 0101 ●
------------------
T1Low:  ●
Komentosana:
Pcntr+1 ●

Tila:
Pcntr:bXXXX 0110


------------------
T1High: ●
Komentosana:
Pcntr+1 ●

Tila:
Pcntr:bXXXX 0110


------------------
T2Low:  ●
Komentosana:
RAM  ->Bus  ●
Bus  ->Ireg ●
Tila:
RAM  :b1111 0000 ●
Bus  :b1111 0000 ●
Ireg :b1110 ???? ●
------------------
Clock Stop: ●
Komentosana:
-

Tila:
Ireg :b1111 ???? ●


------------------
```



Kuva 37. Testiohjelman manuaalinen ajo, komento HLT.

4 YHTEENVETO SEKÄ KOMMENTTEJA TOTEUTUKSESTA

Tietokoneen käännettyä versiota kehitysalustalle ajettaessa havaittiin alussa hieman ongelmia. Koneen kellojen ajastuksessa oli aluksi koneen toiminnan sekoittavia synkronointiongelmia sekä ajastussimulaatioissa, että kehitysalustalla ajettaessa. Ongelma ratkesi lopulta kun Internetistä löytyi ajastusanalyysiraportti, joka kertoi että Quartuksen fitterin asetuksista pitää kytkeä "Optimize Hold Timing fo All Paths" optio päälle, jotta sisäisillä piireillä toteutettujen laskureiden ja kellojen ajastukset eivät menisi sekaisin. (Timing Analysis 2009: 4)

Edellisen ongelman ratkaisun jälkeen tietokone ajoi testiohjelman onnistuneesti läpi sekä ajastus simulaatiossa, että fyysisellä kehitysalustalla, joten tämä kahdeksan bittinen tietokone näyttää toimivan niin kuin sen pitää.

Tietokoneen toteutuksessa olisi silti vielä myös parantamisen varaa. SAP1.bdf piirikaavio on hieman sekava, koska tietokoneen eri komponenttipiirien datan lukemiseksi seitsensegmenttinäyttöihin, siihen lisättiin viime hetkellä useita eri "tap" signaaleita, jotka on sijoitettu kaavioon hieman epäsiististi.

Lisäksi normaalia nollaussignaalia ja käänteistä kellosignaalia ei käytetä tässä toteutuksessa ollenkaan, ja niiden generointi hallintasekvensserissä on täysin turhaa. Ne ovat toteutuksessa vain sen vuoksi että Malvinon ja Brownin (1999) kirjan toteutuksessa käytettiin sekalaisia TTL piirejä, joista osan kello ja reset signaalit olivat käänteisiä ja osan normaaleja, mikä vaati molemman tyyppisten signaalien generointia.

Myös kellosyklin tilan rinkilaskurin piiri on hitusen monimutkaisempi kuin tarpeen, koska se on tehty kirjan esimerkkipiirin mukaan, ja sen ensimmäinen bitti on käännetty päinvastaiseksi, koska kirjan toteutuksessa käytetyissä JK-kytkimissä oli vain reset, eikä ollenkaan preset sisääntuloa.

Lisäksi tietokoneen signaalien nimeämiskäytäntöjä voisi myös yhtenäistää. Tällä hetkellä ne ovat vielä hieman sekavia.

Tämän raportin kuvaama tietokone päätyi silti toimimaan niin kuin sen pitikin, ja sen toteutus oli varsin opettavainen projekti, joka selvensi minulle varsin hyvin miten tietokoneet toimivat matalalla tasolla.

LÄHDELUETTELO

A.B. Malvino & J.A. Brown (1999). Digital Computer Electronics Third Edition. Toim. Glencoe McGraw-Hill

Ben Eater (2016). 8-bit computer RAM intro. [Verkkovideodokumentti] Saatavissa: https://www.youtube.com/watch?v=FnxPIZR1ybs

Timing Analysis of Internally Generated Clocks in Timequest v.2.0. (2009). [Verkkodokumentti] Saatavissa: https://www.mixdown.ca/redmine/attachments/download/71/Timing_Analysis_of_Internally_Generated_Clocks_in_Timequest_v2.0.pdf