Tampereen yliopisto

Toni Fürpass

JÄRJESTELMÄPIIRIT

Matalan tehonkulutuksen toteutusteknologiat

Kandidaatintyö
Informaatioteknologian ja viestinnän tiedekunta
Tarkastaja: Katja Laine
Toukokuu 2025

TIIVISTELMÄ

Toni Fürpass: Järjestelmäpiirit, matalan tehonkulutuksen toteutusteknologiat.
(Englanniksi System-on-Chip – Low power implementation technologies.)
Kandidaatintyö
Tampereen yliopisto
Tieto- ja sähkötekniikan kandidaattiohjelma, sähkötekniikka
Toukokuu 2025

Tässä kirjallisuuskatsauksessa käsitellään järjestelmäpiirien yleisrakennetta ja vastataan kysymyksiin missä laitteissa järjestelmäpiirejä hyödynnetään, mitä ominaisuuksia järjestelmäpiirien hyödyntämisellä elektroniikkalaitteissa saavutetaan, mikä on matalan tehokulutuksen merkitys elektroniikkalaitteissa, mistä tekijöistä tehonkulutus koostuu ja mitkä ovat yleisimmät matalan tehonkulutuksen optimointimenetelmät. Lähteet painottuvat kirjallisiin teoksiin, konferenssijulkaisuihin ja vertaisarvioituihin artikkeleihin.

Energiankulutuksen ja IoT-laitteiden trendit näyttävät suuntaa kasvavalle tarpeelle kehittää järjestelmäpiirejä ja keskittyä matalan tehonkulutuksen saavuttamiseen. Järjestelmäpiirit ovat sulautetuissa järjestelmissä ja mobiililaitteissa käytettyä teknologiaa, joka integroi suurimman osan tai kaikki laitteen toiminnallisista osista yhdelle mikrosirulle.

Tehonkulutus koostuu laitteen toimintojen aiheuttamista dynaamisista tehohäviöistä ja käytetystä transistoriteknologiasta sekä koosta riippuvista staattisista tehohäviöistä. Dynaamisten häviöiden optimointi keskittyy kellosignaalin taajuuden ja käyttöjännitteen minimointiin. Staattisten tehohäviöiden optimointi keskittyy vuoto- ja oikosulkuvirtojen vaikutusten minimointiin.

Yleisimmät matalan tehonkulutuksen menetelmät liittyvät laitteen toiminnalliseen suunnitteluun eli dynaamisten häviöiden minimointiin. Kellosignaali ajoittaa järjestelmän toimintaa ja on nopeimmin vaihtuva signaali, joten tavoitteena on optimoida kellosignaalin taajuus mahdollisimman matalaksi. Yleinen ja halpa matalan tehonkulutuksen menetelmä on nimeltään clock gating, jossa järjestelmä jaetaan toimintojen mukaan eri osiin. Nämä osat jaetaan niitä ajoittaviin kelloverkkoihin, joiden avulla järjestelmän osat voidaan eristää kellosignaalin muutoksista, kun ne eivät ole aktiivisia. Toimimattomana olevat järjestelmän osat voidaan myös eristää käyttöjännitteestä, kunnes niiden toimintaa taas tarvitaan. Kyseistä menetelmää kutsutaan nimellä power gating.

Unitilat ja matalamman suoritustehon tilat optimoivat järjestelmän toimintaa skaalaamalla kellotaajuus ja käyttöjännite matalampaan tasoon tai eristämällä piirin osat clock gating ja power gating -menetelmillä. Käytön aikaista dynaamista jännitteen ja taajuuden skaalausta kutsutaan nimellä dynamic voltage and frequency scaling. Laitteiden toiminnat, jotka vaativat hetkittäistä suorittamista voidaan optimoida menetelmällä, jossa ohjelmallisesti ennustetaan toimintaa ja jaetaan toiminnon toteuttava osa uni- ja toiminta-ajanjaksoihin. Tätä jaksollista herättämis- ja sammuttamisjaksotusta kutsutaan nimellä duty cycle. Edellä mainituista menetelmistä ja niitä hallitsevista toteutuksista koostuu dynaaminen tehonhallinta eli dynamic power management.

Matalan tehonkulutuksen menetelmät lisäävät viivettä ja heikentävät suoritustehoa. Suoritustehoa parantavien arkkitehtuurien ja ajoitusta optimoivien suunnittelumenetelmien hyödyntäminen mahdollistaa jännitteen ja kellosignaalin taajuuden madaltamisen. Näitä suunnitterakenteita kutsutaan nimillä rinnakkaisuus ja liukuhihnoitus.

Useiden tutkimusten ja konfferenssi julkaisujen päätelmien perusteella matalan tehonkulutuksen optimoinnin merkitys elektroniikkasuunnittelussa on kasvanut ja tulee kasvamaan entisestään. Järjestelmäpiirit mahdollistavat tehonkulutuksen optimointia niissä sovellutuksissa, joissa se on laitteen käytön kannalta merkittävintä. Tämä kirjallisuusselvitys kertoo yleisimmistä tehonkulutuksen optimointimenetelmistä.

Avainsanat: Järjestelmäpiirit, matala tehonkulutus

Tämän julkaisun alkuperäisyys on tarkastettu Turnitin Originality Check -ohjelmalla.

TEKOÄLYN KÄYTTÖ OPINNÄYTTEESSÄ

Opinnäytteessäni on käytetty tekoälysovelluksia:

☐ Ei
☒ Kyllä

Ilmoitukseni mukaan olen käyttänyt opinnäytteessäni tutkielmaprosessin aikana seuraavia tekoälysovelluksia:

Tekoälysovellusten nimet ja versiot:
Scopus AI (hakutyökalu, joka hyödyntää generoivaa tekoälyä: GenAI)

Käyttötarkoitus:
Scopus AI -tekoälyä on käytetty kirjoitelmassa esiintyvien käsitteiden perusymmärryksen hankkimiseen ja lähteiden hakuun. Yhtäkään tekoälyn tuottamaa tekstiä ei ole käytetty suoraan tämän kirjoitelman sisällössä.

Osiot, joissa tekoälyä on käytetty:
Scopus AI -tekoälyllä hankittuja lähteitä löytyy vaihtelevin määrin kaikista luvuista.

Olen tietoinen siitä, että olen täysin vastuussa koko opinnäytteeni sisällöstä, mukaan lukien osat, joissa on hyödynnetty tekoälyä, ja hyväksyn vastuun mahdollisista eettisten ohjeiden rikkomuksista.

ALKUSANAT

Kandidaatintyöni aihe oli selvä jo pitkään. Valitsin sulautetut järjestelmät pääaineekseni SoC-HUB:in vierailuluennon pohjalta, jonka kävi pitämässä Ari Kulmala ja Timo Hämäläinen. Aiheen rajaukseen sain apua Ari Kulmalalta, joten hänelle iso kiitos tuesta ja ajatuksista. Työ on yleispiirteinen kirjallisuuskatsaus yleisimpiin keinoihin madaltaa tehonkulutusta järjestelmäpiireissä. Yleispiirteisyydestä huolimatta, työn sisältö painottuu vertaisarvioiduista artikkeleista kerättyyn tietoon sekä alan kirjallisuuteen. Toivon työni tarjoavan selkeän ensikatsauksen järjestelmäpiireihin ja niiden suunnittelumenetelmiin. Vähintäänkin kirjoitelman tekeminen oli itselleni hyvin opettavainen kokemus. Suuret kiitokset vielä ohjaajalleni Katja Laineelle kaikesta palautteesta ja avusta työn aikana.

Tampereella, 4.5.2025

Toni Fürpass

SISÄLLYSLUETTELO

1. JOHDANTO 1
2. JÄRJESTELMÄPIIRIT YLEISESTI 3
2.1 Järjestelmäpiirin määritelmä 3
2.2 Järjestelmäpiirien käyttökohteet 6
2.3 Motivaatio tehonkulutuksen optimoinnille 7
3. MATALAN TEHONKULUTUKSEN SUUNNITTELUMENETELMÄT 9
3.1 Tehonkulutuksen muodot 9
3.2 Asynkroninen ja synkroninen järjestelmä 10
3.3 Usean kellosignaalin järjestelmät 12
3.4 Clock gating -menetelmä 15
3.5 Power gating -menetelmä 18
3.6 Dynaaminen tehonhallinta 19
3.7 Liukuhihnoitus ja rinnakkaisuus 22
4. YHTEENVETO JA PÄÄTELMÄT 25
LÄHTEET 28

LYHENTEET JA MERKINNÄT

ACEEE	American Council for an Energy-Efficient Economy, suom. amerikkalainen energiatehokkuusneuvosto.
ACG	Automatic clock gating, suom. automaattinen kellon portitus.
ALU	Arithmetic logic unit, suom. aritmeettislooginen yksikkö.
BT	Bluetooth. langaton tiedonsiirtotekniikka.
CG	Clock gating. Viittaa kellosignaalin hallintarakenteeseen.
CLK	Clock, viittaa kellosignaaliin.
CMOS	Complementary Metal-Oxide-Semiconductor, suom. komplementaarinen metallioksidipuolijohde.
CPU	Central prosessing unit, suom. keskusprosessori.
DC	Duty cycle, suom. toimintajakso.
DFF	Data/delay Flip-Flop, suom. datakiikku tai D-kiikku.
DPM	Dynamic power management.
DSP	Digital signal processor, suom. digitaalinen signaaliprosessori.
DVFS	Dynamic voltage and frequency scaling, suom. dynaaminen jännitteen ja taajuuden skaalaus.
EDA	Electronic design automation, suom. elektronisen suunnittelun automaatio.
EN	Enable. Tekstissä viitataan käyttöönottosignaaliin (sisääntulo).
EN_L	Enable latch, kiikun käyttöönottosignaali (ulostulo).
FIFO	First in first out, suom. ensin sisään, ensin ulos.
GALS	Globally asynchronous, locally synchronous, suom. globaalisti asynkroninen ja paikallisesti synkroninen.
GALDS	Globally asynchronous, locally dynamic system, suom. globaalisti asynkroninen, paikallisesti dynaaminen järjestelmä.
GC	Gray code. Nimitys binääriesitykselle.
GCG	Global clock generator, suom. globaali kellosignaaligeneraattori.
GNSS	Global navigation satellite system, suom. globaali navigointi satelliittijärjestelmä.
GPU	Graphical processing unit, suom. grafiikkasuoritin/näytönohjain.
HDL	Hardware description language, suom. laitteistokuvauskieli.
HdS	Hardware dependent software, suom. laitteisto riippuvainen ohjelmisto.
IC	Integrated circuit, suom. mikropiiri/integroitu piiri.
ICLK	Inverted clock, viittaa invertoituun kellosignaaliin.
IIR	Infinite impulse response. Ominaisuus, jota sovelletaan lineaarisiin aikainvariantteihin järjestelmiin.
IP	Intellectual property, suom. omistusoikeus.
IoT	Internet of things, suom. esineiden internet.
I/O	Input / Output, suom. siirräntä.
LTE	Long term evolution. Langaton tiedonsiirtotekniikka.
MCU	Microcontroller unit, suom. mikrokontrolleriyksikkö.
MOS	Metal-oxide-semiconductor, suom. metallioksidipuolijohde.
MPEG	Moving pictures experts group. Datavirta standardi.
MUX	Multiplexer, suom. multiplekseri.
NMOS	N-type metal–oxide–semiconductor, suom. N-tyyppinen metallioksidipuolijohde.
NoC	Network-on-Chip. Alijärjestelmien välinen kommunikointilogiikka.
PMOS	P-type metal-oxide-semiconductor, suom. P-tyyppinen metallioksidipuolijohde.

PMU	Power management unit, suom. tehonhallintayksikkö.
RAM	Random access memory, suom. hajasaantimuisti.
ROM	Read only memory, suom. lukumuisti.
Rx	Radio receiver, suom. radiovastaanotin.
RTL	Register transfer level, suom. rekisterisiirtymätaso.
SoC	System-on-Chip, suom. järjestelmäpiiri.
TFET	Tunnel field effect transistor, suom. tunnelikanavatransistori.
Tx	Radio transmitter, suom. radiolähetin.
ULP	Ultra-low power, suom. erittäin matala tehonkulutus.
USB	Universal serial bus, suom. universaali sarjaväylä.
V	Voltage, suom. jännite.
VLSI	Very large system integration, suom. todella suurien järjestelmien integrointi.
WiFi	Wireless fidelity. Langattoman verkon standardi.

C_L	Latauskapasitanssi.
C_{ref}	Referenssilatauskapasitanssi.
f_{clk}	Kellosignaalin taajuus.
f_{ref}	Referenssikellosignaalin taajuus.
$I_{leakage}$	Vuotovirta.
I_{sc}	Oikosulkuvirta.
P_{dyn}	Dynaaminen tehonkulutus.
P_{ref}	Referenssitehonkulutus.
P_{total}	Kokonaistehonkulutus.
p_t	Tilamuutoksen todennäköisyys.
V	Jännitteen vaihtelu.
V_{dd}	Käyttöjännite.
V_{ref}	Referenssijännite.

1. JOHDANTO

Transistorien määrän kasvu mikrosiruilla sekä prosessorien suoritustehon kasvu ovat tuoneet mukanaan haasteita energiankulutuksen ja elektroniikan lämpenemisen näkökulmasta. Kehittyvät elektroniikkatuotteet kuten IoT-laitteet (Internet of things, esineiden internet), puettava teknologia ja jatkuvasti päällä olevat laitteet vaativat toimintaiän, lämpöominaisuuksien, helppokäyttöisyyden ja pienikokoisuutensa puolesta matalaa tehonkulutusta.

Energiantuotanto ja -kysyntä tarjoavat ajankohtaisia haasteita, kun ne rinnastetaan ilmaston lämpenemisen ja kestävän kehityksen kanssa. Tutkimus [1] analysoi vuosien 2000–2020 tilastoja, tuoden esiin korrelaation bruttokansantuotteen kasvun ja energiankulutuksen välillä. Tutkimus [2 s.2] teki selvityksen viimeisen kymmenen vuoden ajalta, jonka pohjalta voidaan todeta tekoälyn kehityksen kasvaneen merkittävästi. Samassa tutkimuksessa todetaan, kuinka suurempien parametrien hyödyntäminen tekoälyn kouluttamisessa vaatii paljon energiaa. Raportin [3] mukaan on arvioitu, että vuonna 2015 IoT-laitteita oli markkinoilla 237 miljoonaa ja niiden ennustettiin kasvavan 923 miljoonaan. Kasvun ennustettiin tapahtuvan vuoteen 2020 mennessä. Lukemien tarkkuudesta riippumatta julkaisuiden [1, 2, 4] perusteella energiankulutuksen, IoT-teknologian ja tekoälyn trendit ovat pitkään olleet kasvussa. Järjestelmäpiirejä hyödynnetään akkukäyttöisissä mobiililaitteissa sekä sulautetuissa järjestelmissä, joten IoT-laitteiden yleistyminen, tekoälyn vaikutukset energiankulutukseen sekä energiankulutuksen trendit painottavat matalan tehonkulutuksen merkitystä elektroniikkasuunnittelussa.

Elektroniikkalaitteen suunnittelu vaatii valintoja suorituskyvyn, hinnan ja tehonkulutuksen välillä. Nämä osa-alueet ovat ristiriidassa keskenään, sillä yhden osa-alueen optimointi vaikuttaa muihin päinvastaisesti. Uusien teknologioiden ansiosta kaikilla osa-alueilla on tultu eteenpäin.

Järjestelmäpiirit ovat integroituja mikropiirejä, joita käytetään mobiililaitteissa sekä sulautetuissa järjestelmissä. Järjestelmäpiirit toteuttavat suurimman osan tai kaikki elektroniikkalaitteen toiminnoista. Useat matalaa tehonkulutusta vaativat käyttökohteet on toteutettu järjestelmäpiirejä hyödyntämällä, sillä järjestelmäpiireille ominainen laitteen komponenttien integroiminen vähentää ylimääräistä kommunikointilogiikkaa, säästää logiikkaan liittyviä tehohäviöitä ja mahdollistaa laitteen pienemmän koon.

Tulevissa luvuissa pohditaan järjestelmäpiirien roolia elektroniikkalaitteissa sekä matalan tehonkulutuksen asemaa elektroniikkalaitteiden kehityksessä. Edellä olevista kysymyksistä siirrytään tarkastelemaan, miten tehonkulutus tapahtuu elektroniikassa, mistä tekijöistä se koostuu sekä mitä menetelmiä tehonkulutuksen madaltamiselle on käytettävissä.

Luvussa 2 kerrotaan järjestelmäpiireistä yleispiirteittäin, tuodaan esiin järjestelmäpiirien käyttökohteita ja esitetään esimerkkejä matalan tehonkulutuksen järjestelmäpiirien käyttökohteista. Luku 3 esittelee tehonkulutuksen muodot sekä tehonkulutuksen optimointiin hyödyllisiä suunnitteluteknologioita. Luku 4 sisältää yhteenvedon, ja luo kokonaiskuvan tehonkulutuksen sekä järjestelmäpiirien merkityksestä elektroniikkalaitteiden suunnittelussa.

2. JÄRJESTELMÄPIIRIT YLEISESTI

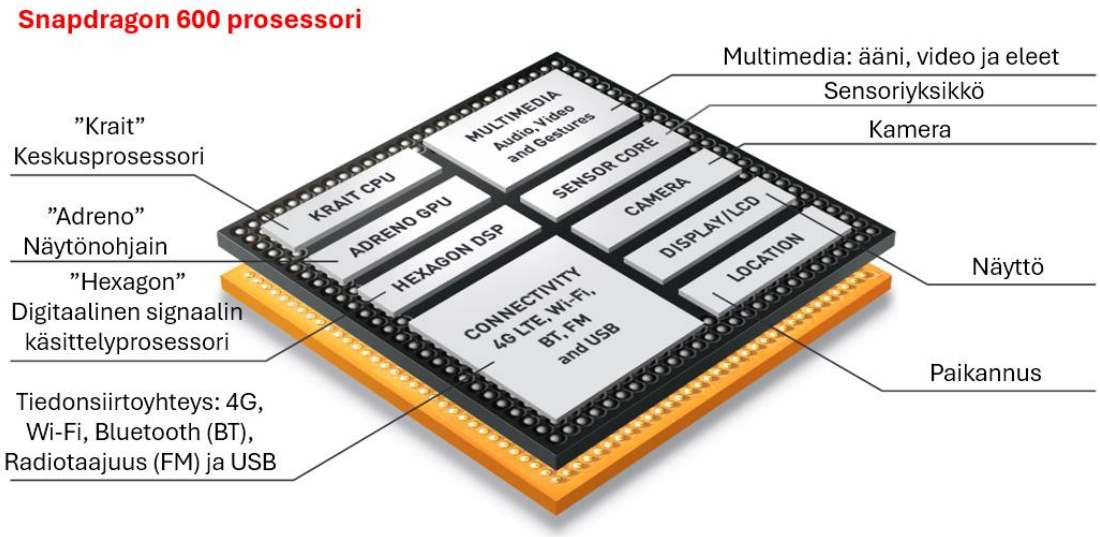
Järjestelmäpiiri toteuttaa suurimman osan tai kaikki elektroniikkalaitteen toiminnoista. Se jakautuu pienempiin kokonaisuuksiin, kuten alijärjestelmiin tai yksittäisen tehtävän suorittaviin osiin. Erilaiset prosessorit, muistikomponentit, mikrokontrollerit, vastaanottimet ja lähettimet muodostavat alijärjestelmiä ja yhdessä ne muodostavat kokonaisuuden, joka toteuttaa elektroniikkalaitteen toiminnot. Kyseinen kokonaisuus integroituna yhdelle mikrosirulle on järjestelmäpiiri.

2.1 Järjestelmäpiirin määritelmä

Järjestelmä on ryhmä toiminnallisia osia tai "lohkoja", jotka jokainen toteuttavat jonkun funktion ja yhdessä toteuttavat kaikki elektroniikkalaitteen toiminnot. Lohkot ovat komponentteja tai alijärjestelmiä. Alijärjestelmä on useista pienistä toiminnallisista osista koostuva kokonaisuus, joka toteuttaa osiensa avulla jonkin laitteen toiminnalle oleellisen funktion ja toimii osana isompaa järjestelmää. Järjestelmän eri osat koordinoivat keskenään, tarjoten omat palvelunsa. Alijärjestelmät yleensä sisältävät yhden tai useamman prosessorin tai hyödyntävät toisen alijärjestelmän prosessoria. [5, luku 1.1]

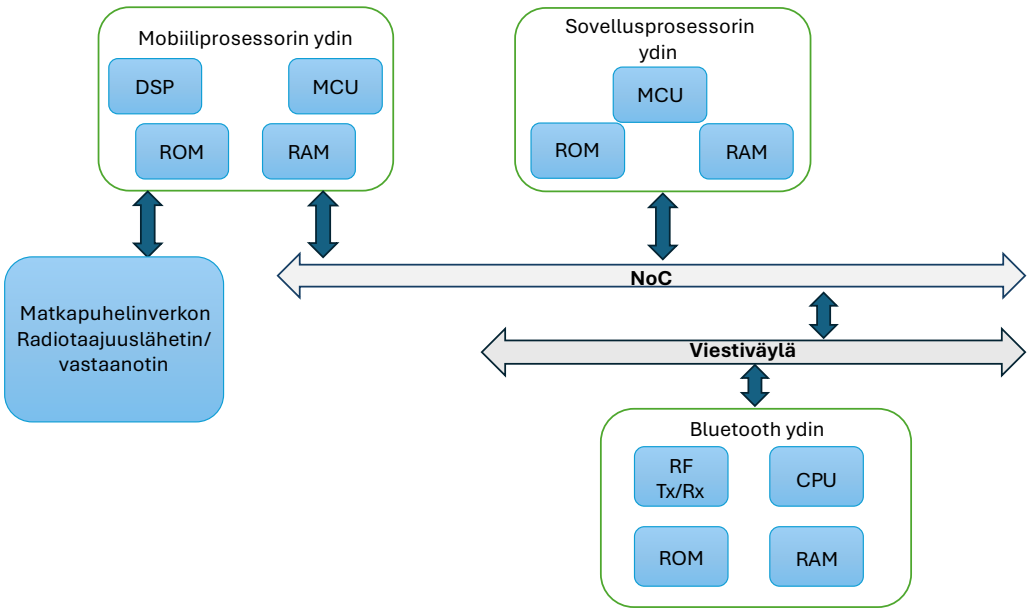
Järjestelmäpiiri (engl. System-On-Chip, SoC) on yhdellä mikrosirulla sijaitseva, useista toiminnallisista lohkoista koostuva järjestelmä. Järjestelmäpiirit tuotetaan hyödyntäen VLSI-teknologioita (engl. Very large-scale integration), joilla tuotetaan laajoja integroituja piirejä (engl. Integrated circuit, IC). CMOS-teknologia (engl. Complementary Metal-Oxide-Semiconductor) on yleisin suurien IC-järjestelmien kehityksessä hyödynnetty transistoriteknologia. [5, luku 2.5] Transistorit ovat aktiivikomponentteja, joiden läpi kulkevaa virtaa (tarkemmin sen kanavaa) voidaan hallita hilalle tuodulla jännitteellä.

Järjestelmäpiirin rakennetta voidaan havainnollistaa kuvan 2.1 Snapdragon 600 -järjestelmäpiirin lohkokaaviolla [6]. Lohkokaaviosta nähdään keskusprosessori (Central processing unit, CPU), näytönohjain (Graphical processing unit, GPU), digitaalisten signaalien prosessori (Digital signal processor, DSP), lisäksi multimedialle, antureille, paikannukselle, verkkoyhteyksille, kameralle ja näytölle on omat alijärjestelmänsä.



Kuva 2.1. *Qualcomm Snapdragon 600 -järjestelmäpiiri ja sen alijärjestelmät. Muokattu lähteestä [6].*

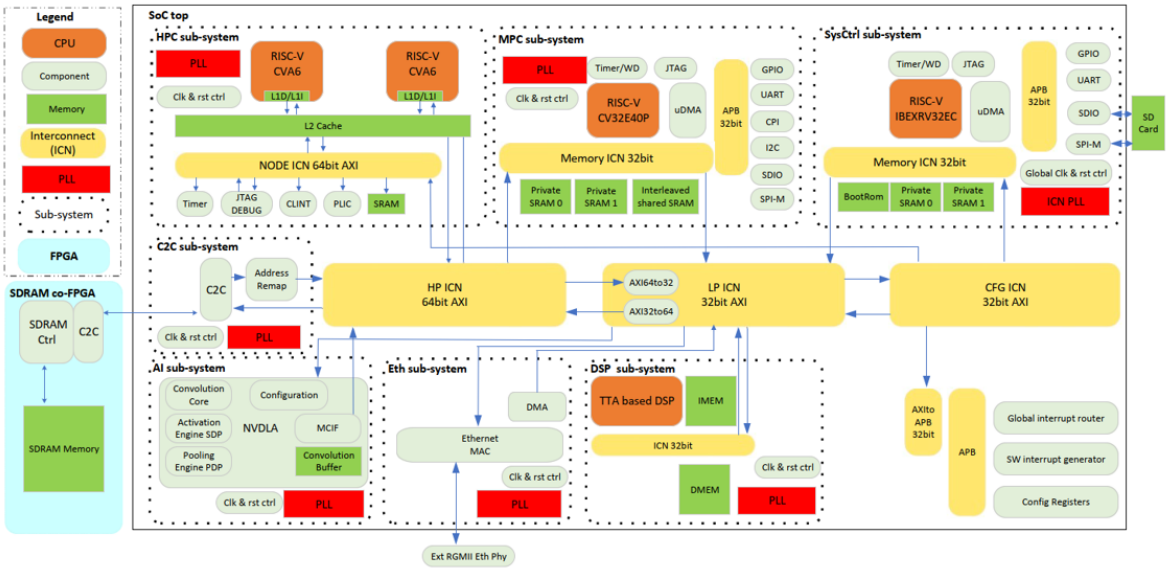
Kuva 2.1 antaa selkeän jaon eri alijärjestelmien välille. Elektroniikkalaitteen toiminnalle on olennaista, että sen osat voivat kommunikoida keskenään. Alijärjestelmien kommunikointia voidaan havainnollistaa kuvan 2.2 lohkokaaviolla. Kuvasta 2.2 nähdään, kuinka alijärjestelmien välillä kulkee kaksi kommunikointiväylää. Järjestelmän sisäinen kommunikointi tapahtuu ”Network-on-Chip” eli NoC-yhteyttä hyödyntäen, joka on samankaltainen logiikaltaan kuin reitittimissä hyödynnetty tekniikka. Laitteen ulkopuolisten signaalien (muun muassa muista laitteista, näppäimistöstä ja lähettimistä tulevat signaalit) käsittely tapahtuu alijärjestelmissä, jotka on yhdistetty viestiväylään (engl. Peripheral bus).



Kuva 2.2. Kaavio on tehty teoksessa [5 s. 5] esitetyn puhelimeen suunnitellun järjestelmäpiirin lohkokaavion pohjalta.

Kuvan 2.2 alijärjestelmien sisällöt koostuvat useista prosessoreista, kuten keskusprosessoreista (engl. Central processing unit, CPU), digitaalisen signaalin käsittelyprosessoreista (engl. Digital signal processor, DSP), mikrokontrollereista (engl. Microcontroller unit, MCU), hajasaanti- ja lukumuisteista (engl. Random access memory, RAM, read only memory, ROM) ja lisäksi lohkot voivat sisältää radiolähetin- (engl. Radio transmitter, Tx) ja radiovastaanotinyksiköitä (engl. Radio receiver, Rx). [5 s. 5]

Tampereen yliopistolla toimivan SoC-HUB:in julkaiseman Ballast-järjestelmäpiirin lohkokaaviosta, joka on esitelty kuvassa 2.3, voidaan havainnollistaa alijärjestelmien monimutkaisuutta. Erilaisten komponenttien, kuten prosessorien, muistin, ja kommunikointilogiikan määrä ja sijoittelu voivat vaihdella paljonkin eri järjestelmien välillä. Integrointi yhdelle sirulle vaatii suunnittelijoilta laaja-alaista ymmärrystä ja osaamista.



Kuva 2.3. *SoC-HUB:in julkaiseman Ballast-järjestelmäpiirin lohkokaavio [7].*

Valmiita suunnitteluratkaisuja (ohjelmisto- sekä laitteistotasolla) hyödynnetään järjestelmäpiirien monimutkaisuuden ja suunnittelusta markkinoille kestävän ajan lyhentämiseksi. Valmiita laitteisto- sekä ohjelmistotason lohkoja kutsutaan IP-lohkoiksi (engl. Intellectual property, IP). Artikkelissa [8 s. 1] IP:llä viitataan ainoastaan laitteistotason ratkaisuiden uudelleenkäyttöön, mutta ohjelmistoratkaisujen uudelleenkäyttö on myös merkittävä keino järjestelmäpiirien suunnittelun nopeuttamiseksi. [8, 9, 10, 11]

Julkaisussa [9 s.4] huomautetaan, että järjestelmäpiirien kehityksen nopeuttamiseksi ohjelmistokehityksellä on yhtä kriittinen rooli kuin laitteistosuunnittelulla. Julkaisu [9 s.4] viittaa artikkelien [10] ja [11] sisältöihin, jotka molemmat painottavat ohjelmistoratkaisujen kasvavaa merkitystä järjestelmäpiirien kehityksessä sekä IP-lohkojen uudelleenkäytön

merkitystä järjestelmäpiirisuunnittelun nopeuttamisessa. Artikkeli [10] tarjoaa esimerkkejä työkaluista ja muista toteutusmetodeista, joilla laitteisto- ja ohjelmistotason haasteita ratkaistaan. Artikkeli [10] esittelee myös laitteistoriippuvaisen HdS-ohjelmointirajapinnan (engl. Hardware dependent Software, HdS) laitteiston ja ohjelmiston samanaikaisen suunnittelun haasteiden ratkaisuksi. Artikkelissa [11] keskitytään enemmän uudelleenkäytön hyötyihin järjestelmäpiirien suunnittelun tuottavuuden parantamisessa, ja käydään läpi IP-lohkojen kehitysvaiheet ja kehityksessä hyödynnetyt työkalut sekä metodit.

Teos [5 s. 15] kuvaa järjestelmäpiirien avulla integroitujen elektroniikkalaitteiden hyötyjen olevan laitteen pienempi koko, optimoitu suoritusteho, pienemmät tuotantokustannukset sekä luotettava valmistus. Haasteiksi teoksessa [5 s. 15] luetellaan suuret suunnittelukustannukset, korkeat laatuvaatimukset, pitkät kehitysajat (jos valmiita IP-lohkoja ei ole käytettävissä) sekä korkean suunnittelijoilta vaaditun osaamistason. [12 s. 2]

Järjestelmäpiirit ovat siis puolijohdekomponenteista rakennettuja mikropiirejä. Ne ovat laajoja integroituja järjestelmiä, jotka kykenevät toimimaan itsenäisesti ja toteuttavat suurimman osan elektroniikkalaitteen toiminnoista. Järjestelmäpiirien monimutkaisuus johtaa pitkiin kehitysaikoihin, minkä vuoksi kehitysaikojen lyhentämiseksi yritysten kannattaa myös hyödyntää valmiita IP-lohkoja.

2.2 Järjestelmäpiirien käyttökohteet

Järjestelmäpiirien käyttökohteita on lukuisia. Teoksessa [13, luku 1.2] mainitaan, että järjestelmäpiirejä hyödynnetään IoT-käyttökohteissa (teollisuudessa sekä kodin laitteissa), autoteollisuudessa sekä terveydenhuollon laitteissa. Muita teoksen mainitsemia käyttökohteita ovat älypuhelimet ja muut mobiililaitteet, datakeskukset, lääketieteelliset laitteet, satelliittien kommunikaatio- ja avaruusteknologia. Käyttökohteita on muitakin, kuten Nokian julkaisema [14] järjestelmäpiiri tietoliikennetekniikan sovellutuksiin tai MoSys:n ja Fujitsun [15] kulutuselektroniikkalaitteisiin suunniteltu järjestelmäpiiri.

IoT-laitteiden täytyy olla jatkuvasti päällä ja niiden täytyy kuunnella langattomia signaaleja, joten niiden suunnittelussa matala tehonkulutus sekä langattomien signaalien sulava vastaanottaminen ja lähetys ovat keskeisessä asemassa. Järjestelmäpiirejä on tutkittu erilaisissa langattoman tiedonsiirron toteutuksissa, kuten esimerkiksi tutkimusartikkelissa [16] selvitettiin langattoman lähiverkon WiFi-SoC-mikrokontrollerin hyödyntämistä vedenpinnan korkeusmittauksissa. Artikkeli [17] puolestaan esittelee erittäin matalan tehonkulutuksen (engl. Ultra low power, ULP) IoT-järjestelmäpiiri toteutuksen.

Järjestelmäpiireille on sovellutuksia lääketieteellisissä laitteissa, jotka mittaavat verensokeritasoja, sydämen sykettä, lähettävät hätäviestin tai seuraavat muita terveystiloja. Pienikokoiset puettavat laitteet vaativat akunkeston ja suorituskyvyn puolesta matalaa tehonkulutusta. [18 s.1] Julkaisussa [18 s.1] mainitaan esimerkkinä, että implantoitavan lääketieteellisen järjestelmäpiirin keskimääräisen tehonkulutuksen pitää olla alle 10 µA ja laitteen täytyy kestää 10 vuotta yhdellä AA-paristolla.

Järjestelmäpiirejä käytetään siis laitteissa, jotka vaativat elektroniikaltaan pientä kokoa, joiden virtalähde on rajallinen, lämpöominaisuudet halutaan optimoida ja suorituskyky sovittaa eri toimintojen vaatimusten mukaan (esim. virransäästö tai korkeampi suoritusteho). Eri prosessorien, muistin ja siirräntälogiikan (engl. Input/output, I/O) integrointi mahdollistavat optimointia elektroniikan koon, suorituskyvyn ja tehonkulutuksen suhteen, mitkä tekevät järjestelmäpiireistä käyttökohteiltaan laaja-alaisen.

2.3 Motivaatio tehonkulutuksen optimoinnille

Artikkeli [9] vuodelta 2007 kuvaa järjestelmäpiirien yhtenä ratkaisevana haasteena matalan tehonkulutuksen toteuttamisen. Monet järjestelmäpiirien käyttökohteista soveltavat pientä kokoa vaativia laitteistoja, kuten puhelimet, älykellot ja puettava teknologia. Kannettavien laitteistojen akkujen tulee kestää eikä laitteisto saa ylikuumentua. Akullisten laitteiden lisäksi myös verkkovirtaan kytkettyjen laitteiden suunnittelussa tehohäviöiden minimointi on tärkeää lämpöhäviöiden ja pakkaussuunnittelun aiheuttamien kustannusten näkökulmasta. Lämpenemisen minimointi myös parantaa sirun toimintakyvyn luotettavuutta. [9]

Artikkelit [9] ja [19] vuosilta 2007 ja 2005 kuvaavat, kuinka elektroniikkalaitteiden pienenevän koon ja kasvavan suorituskyvyn myötä tehohäviöiden optimointi on tärkeässä roolissa paremman suorituskyvyn ja pitkän akunkeston omaavien laitteiden suunnittelussa. Artikkelin [19] yhteenvedossa todetaan, että jos järjestelmäpiirien suunnittelussa onnistutaan ylittämään tehonkulutuksen ja integroinnin tuomat suunnitteluhaasteet, niin silloin nähtäisiin järjestelmäpiirien integrointia viestintälaitteissa, kannettavissa laitteissa sekä kodinkoneissa. Julkaisun [19] esittämä järjestelmäpiirien potentiaali on ACEEE:n (Amerikkalainen energiatehokkuusneuvosto) julkaisun [3] päätelmien perusteella toteutunut. ACEEE:n raportti [3] vuodelta 2017 toteaa IoT-laitteiden arvioidun määrän vuonna 2015 olevan 237 miljoonaa, jonka arvioitiin kasvavan 923 miljoonaan laitteeseen vuoteen 2020 mennessä. IoT-analytics:in raportti [20] (koko raportti ei ollut saatavilla mutta tieto on saatu julkaisijan omilta sivuilta) vuodelta 2024 arvioi vuoden 2023 lopussa IoT-laitteiden määrän olleen 16,6 miljardia. Järjestelmäpiirit ovat keskeinen teknologia IoT-laitteissa, joten ennusteet IoT-laitteiden määrän kasvusta koskevat myös järjestelmäpiirejä.

Julkaisussa [21] pohditaan alle mikrometrin komponenteista suunniteltujen IC-piirien tehonkulutuksen vaikutuksia. Matala tehonkulutus mainitaan ”dominoivaksi” kehityskohteeksi integroitujen piirien kehityksessä. Samalla julkaisussa [21 s. 1] mainitaan, että ”Laitteiden ylikuumeneminen johtaa sirun vahingoittumiseen, joka johtaa suurempien ja kalliimpien pakkausten sekä jäähdytysjärjestelmien käyttöön”, joten tehonkulutuksen rooli laitteiden pakkausominaisuuksissa sekä niiden aiheuttamissa kustannuksissa on merkittävä monissa laitteissa.

3. MATALAN TEHONKULUTUKSEN SUUNNITTELUMENETELMÄT

Tehonkulutuksen optimointi koostuu dynaamisten (laitteen toiminnasta aiheutuvien) tehohäviöiden sekä staattisten (teknologian rakenteesta ja koosta aiheutuvien) tehohäviöiden minimoinnista. Dynaamiset tehohäviöt ovat riippuvaisia jännitteen toisesta potenssista sekä kellosignaalin taajuudesta. Jännitteen madaltamisella sekä kellosignaalin hidastamisella saavutetaan merkittäviä säästöjä laitteen tehonkulutuksessa. Useiden kellosignaalien hyödyntäminen, taajuuden sekä jännitteen madaltaminen sekä harvemmin toiminnassa olevien komponenttien eristys kellosignaalista tai jännitteestä, tarjoavat keinoja tehonkulutuksen madaltamiselle. Jännitteen ja taajuuden madaltaminen hidastavat järjestelmän toimintaa, jolloin suoritusaikojen hidastumista voidaan kompensoida hyödyntämällä rinnakkaisuus ja liukuhihnoitus rakenteita. Järjestelmän ajoitusta kuvataan saman kellosignaalin määräämänä synkroniseksi, ja globaalista kellosignaalista riippumattomana asynkroniseksi.

3.1 Tehonkulutuksen muodot

Tehonkulutus voidaan jakaa staattisiin ja dynaamisiin tehohäviöihin. Staattinen tehonkulutus on puolijohdekomponenteissa tapahtuvaa tehohäviötä, joka ei aiheudu komponenttien tilavaihdoksen, kellosignaalin tai ulostulon muutosten vaikutuksesta. Dynaamisia tehohäviöitä aiheutuu puolijohdekomponenttien kapasitiivisista latautumis- ja purkautumisreaktioista. Staattiset tehohäviöt ovat riippuvaisia käytetystä puolijohdetekniikasta ja kokoluokasta. Dynaamiset tehohäviöt riippuvat laitteen toiminnasta, johon voidaan suoraan suunnittelun menetelmillä ja erilaisilla arkkitehtuureilla vaikuttaa. [21]

Tutkimuksessa [22] määritellään tehonkulutukselle dynaamiset ja staattiset tekijät, jotka kuvataan kaavalla (1)

$$P_{total} = p_t(C_L \cdot V \cdot V_{dd} \cdot f_{clk}) + (I_{sc} \cdot V_{dd} + I_{leakage} \cdot V_{dd}) \qquad (1)$$

Kokonaistehonkulutuksen P_{total} ensimmäinen termi $p_t(C_L \cdot V \cdot V_{dd} \cdot f_{clk})$ kuvaa tilamuutoksista aiheutuvaa dynaamista tehohäviötä mikä on kokonaistehonkulutuksen kannalta tutkimuksen [22] mukaan merkittävin tehohäviön lähde**.** Matalan tehonkulutuksen suunnittelu, laitteen toiminnasta aiheutuvien tekijöiden suhteen, keskittyy minimoimaan C_L, f_{clk} ja V_{dd} termit. C_L on lataus kapasitanssi, f_{clk} kellosignaalin taajuus ja p_t on todennä-

köisyys sille, että tehoa kuluttava muutos tapahtuu. Useimmissa tapauksissa jännitevaihtelua kuvaava V on sama kuin V_{dd} eli lähdejännite. [22] Dynaamiset tekijät voidaan sieventää kaavan (2) mukaisesti.

$$P_{dyn} = (C_L \cdot V^2{}_{dd} \cdot f_{clk}) \quad (2)$$

Kokonaistehonkulutuksen P_{total} oikealla olevat tekijät kaavassa (1) kuvaavat transistorien rakenteesta johtuvia staattisia tehohäviöitä. Staattiset tekijät koostuvat oikosulkuvirrasta I_{sc}, käyttöjännitteestä V_{dd} ja vuotovirrasta $I_{leakage}$. Oikosulkuvirta muodostuu, kun N- ja P-tyyppiset metallioksidi-puolijohteet (engl. Metal-oxide-semiconductor, MOS) eli NMOS- ja PMOS-puolijohteet ovat aktiivisia samaan aikaan. Vuotovirrat ovat riippuvaisia käytetystä teknologiasta. [22]

Mikropiirien komponenteissa tapahtuu siis laitteen toiminnasta (signaalien bittiarvojen muutoksista) aiheutuvaa tehohäviötä sekä staattista eli komponenttien rakenteesta johtuvaa (vuotovirtojen aiheuttamaa) tehohäviötä. Tulevissa luvuissa keskitytään enemmän dynaamisten tehohäviöiden minimointiin.

Dynaamisen tehonkulutuksen minimoinnin keskeinen kohde on järjestelmän kellosignaalin taajuus f_{clk}. CMOS-teknologian yleisin dynaamisen tehonkulutuksen lähde on kondensaattorien tai muiden komponenttien kapasitiivisten ominaisuuksien nopea latautuminen ja purkautuminen. [23] Kellosignaali on järjestelmää ajoittava signaali, joka on tällöin myös piirin nopeimmin muuttuva signaali. Kellosignaalin taajuudella on tällöin merkittävä vaikutus piirin tehonkulutukseen sen muuttuvan luonteen ja laaja-alaisen vaikutusalueen vuoksi.

Käyttöjännite vaikuttaa koko järjestelmään ja koska dynaaminen tehonkulutus on riippuvainen käyttöjännitteen toisesta potenssista, on käyttöjännitteen madaltamisella merkittäviä vaikutuksia dynaamiselle tehonkulutukselle. Jännitteellä on myös vaikutuksia staattisiin tehohäviöihin, joten jännitteen merkitys on entistä suurempi kokonaistehonkulutuksen madaltamisessa.

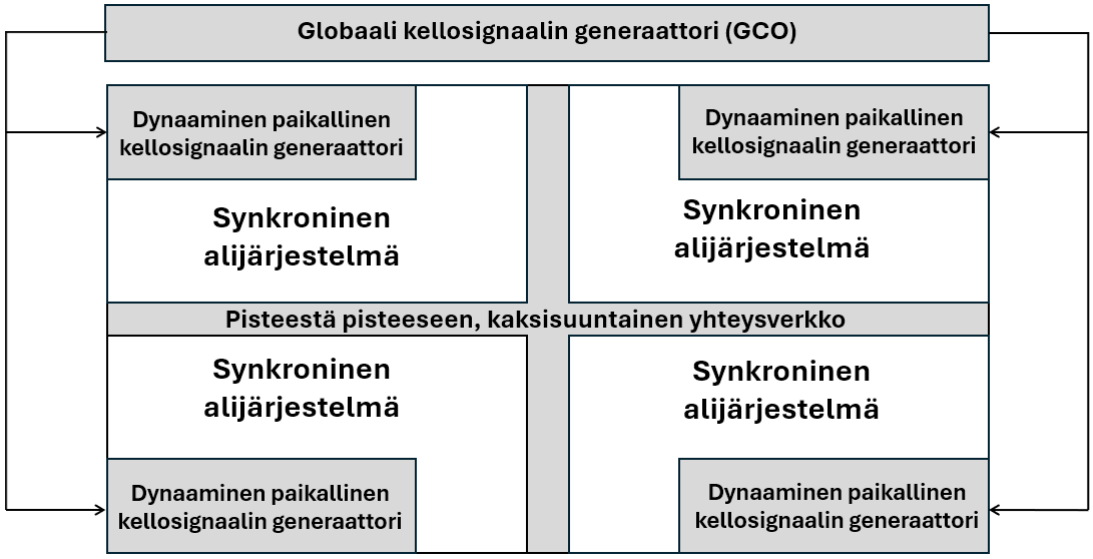
3.2 Asynkroninen ja synkroninen järjestelmä

Järjestelmän toiminnan ajoittaminen tapahtuu generoidun kellosignaalin kontrolloimana. Kellosignaalin mukaan ajoitettua järjestelmää kutsutaan synkroniseksi järjestelmäksi. Teoksen [21 s. 722] mukaan kellosignaali voi muodostaa jopa 45 % järjestelmän kokonaistehonkulutuksesta. Tutkimuksen [24] järjestelmässä kellosignaalin osuus tehonkulutuksesta oli jopa 70 %. Artikkeli [25] esitteli ACG-arkkitehtuurin (engl. Automatic clock gating, ACG), jonka avulla kellosignaalin aiheuttamaa tehonkulutusta voidaan madaltaa

arviolta 17–50 %. Molemmat tutkimukset [24, 25] tukevat kirjassa [21] esiintyvää väitettä kellosignaalin merkittävästä osuudesta järjestelmän tehonkulutuksen kannalta.

Järjestelmä voidaan suunnitella hierarkkisesti useampaan lohkoon. Kuvasta 3.1 nähdään esimerkki järjestelmästä, joka koostuu neljästä osasta, jotka kommunikoivat keskenään mutta toimivat asynkronisesti omaan tahtiinsa. [26] Asynkronisuus siis viittaa siihen, ettei järjestelmän osa päivitä tietojaan globaalin kellosignaalin määräämillä hetkillä ja toiminta voi tällöin ajallisesti poiketa järjestelmän muista osista. Asynkronisen järjestelmän yhtenä etuna synkroniseen verrattuna on, että signaalit voidaan asettaa matalaksi (vastaa matalampaa jännitettä) heti kun niitä ei enää tarvita [26]. Synkronisessa järjestelmässä joudutaan odottamaan seuraavaa kellojaksoa.

Kuvassa 3.1 on globaali kellosignaaligeneraattori, kaksisuuntainen kommunikointiverkko (engl. Point-to-point bidirectional communication network), paikallisesti synkroniseen tahtiin toimivat alijärjestelmät ja niiden dynaamiset kellosignaaligeneraattorit.



Kuva 3.1 *Kuvaus järjestelmästä, joka on asynkroninen mutta sen alijärjestelmät toimivat synkronisesti. Kuva on muokattu lähteestä [26].*

Järjestelmä on globaalisti asynkroninen eli sen osien toiminta ei ole ajoitettu samaan kellosignaaliin. Osien toiminta ajoittuu sisäisesti luodun kellosignaalin tahtiin. Alijärjestelmät tuottavat kellosignaalinsa globaalin kellosignaalin pohjalta. Alijärjestelmien sisällä muistikomponentit päivittävät tilansa samaan aikaan, joten ne ovat paikallisesti synkronisia. Kuvattua järjestelmää kutsutaan nimellä GALS (globally asynchronous, locally synchronous).

Tutkimuksessa [26] järjestelmään lisättiin lohkojen sisälle mahdollisuus muuttaa paikallista kellosignaalin taajuutta ajon aikana, joka tekee siitä globaalisti asynkronisen, paikallisesti dynaamisen järjestelmän. Nimitys järjestelmälle on tällöin GALDS ja se kuvaa

järjestelmän mahdollisuutta säätää kellosignaalien taajuutta piirin toiminnan aikana. [26] Dynaaminen taajuuden skaalaus mahdollistaa tehonkulutuksen optimoinnin käyttötarkoituksen ja alijärjestelmien aktiivisuuden mukaan.

Tutkimus [26] esittelee raamit usean kellosignaalin sisältävän järjestelmän suunnitteluun. Rakenne vaatii toimiakseen kaksisuuntaisen asynkronisen FIFO-puskurin (engl. First in, first out, FIFO), joka viittaa puskurin järjestykseen. FIFO:n ulostulojärjestys datalle on sama kuin sisääntulo järjestys. FIFO-puskuri tarvitaan eri lohkojen kommunikointia varten, sillä ajoitetaan datan siirto lohkosta toiseen ja määritellään datan prosessoinnin järjestys. Asynkronisten järjestelmien kommunikoinnissa kaksisuuntaisten FIFO-puskureiden molempiin päihin lisätään synkronointilogiikkaa [27]. Lisäksi alijärjestelmät tarvitsevat kellosignaalin generointiin dynaamisia kellosignaaligeneraattoreita. Järjestelmä tarvitsee vielä digitaalisesti kontrolloidun oskillaattorin, jolla luodaan globaali kellosignaali.

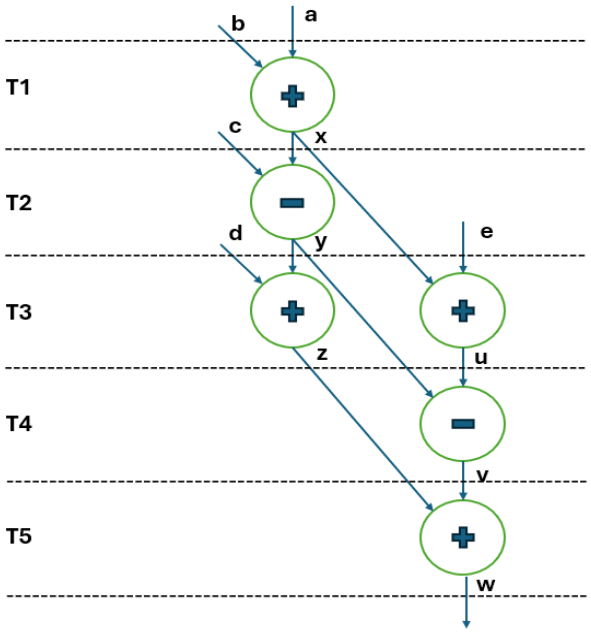
Järjestelmän datan hitainta kulkureittiä kutsutaan kriittiseksi poluksi (engl. Critical path), joka määräytyy mahdollisten kulkureittien suurimman viiveen perusteella [28]. Kellosignaalilla varmistetaan eri laskentapiirien ajoitus ja toiminnallisuus niin, että kaikki tarvittava data on oikeassa osassa laskentapiiriä, kun sitä tarvitaan. Kellosignaali täytyy ajoittaa minimissään kriittisen polun eli järjestelmän hitaimman osan mukaan, oikean ajoituksen varmistamiseksi [29]. Yhdessä dynaamisen jännitteen ja taajuuden skaalauksen (engl. Dynamic voltage and frequency scaling, DVFS) kanssa tutkimuksen [26] esittelemät raamit tarjoavat modulaarisen rakenteen järjestelmäpiirien suunnitteluun.

Asynkronisella suunnitelulla mahdollistetaan järjestelmän osien jako niin, että jokaisen alijärjestelmän sisällä kellosignaalin taajuus saadaan mahdollisimman lähelle kyseisen alijärjestelmän kriittistä polkua. Tämä mahdollistaa kokonaisuudessaan matalampia taajuuksia ja johtaa tehonkulutuksen säästöihin. Asynkronisuus tarjoaa mahdollisuuksia tehonkulutuksen optimoinnille, mutta asynkronisuus ei ole aina tavoiteltua sen monimutkaisuuden vuoksi. Synkronisille järjestelmille löytyy tehonkulutusta optimoivia ratkaisuja, niiden suunnittelu on yksinkertaisempaa ja oikean toiminnan takaaminen luotettavampaa.

3.3 Usean kellosignaalin järjestelmät

Tutkimuksessa [23] usean kellosignaalin vaikutuksia tehonkulutukseen testattiin vertailemalla kuvan 3.3 esittämän rakenteen toteutusta kuvan 3.4 toteutukseen. Molemmat piirit noudattavat kuvan 3.2 mukaista toiminnallisuutta, jossa sisääntulosignaaleille a, b,

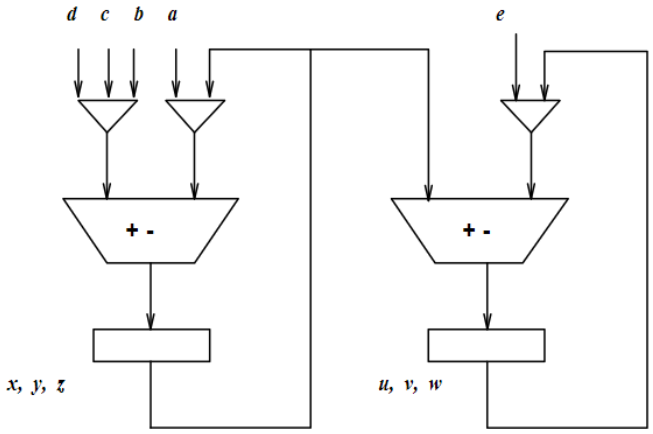
c, d ja e, tehdään laskutoimituksia kellojaksojen T1, T2, T3, T4 ja T5, määräämässä järjestyksessä.



Kuva 3.2 *Ajanjaksoihin T1, T2… T5, jaettu laskentapiirin toiminnallinen kuvaus. Kaavio tehty julkaisun [23] kuvan 1 pohjalta.*

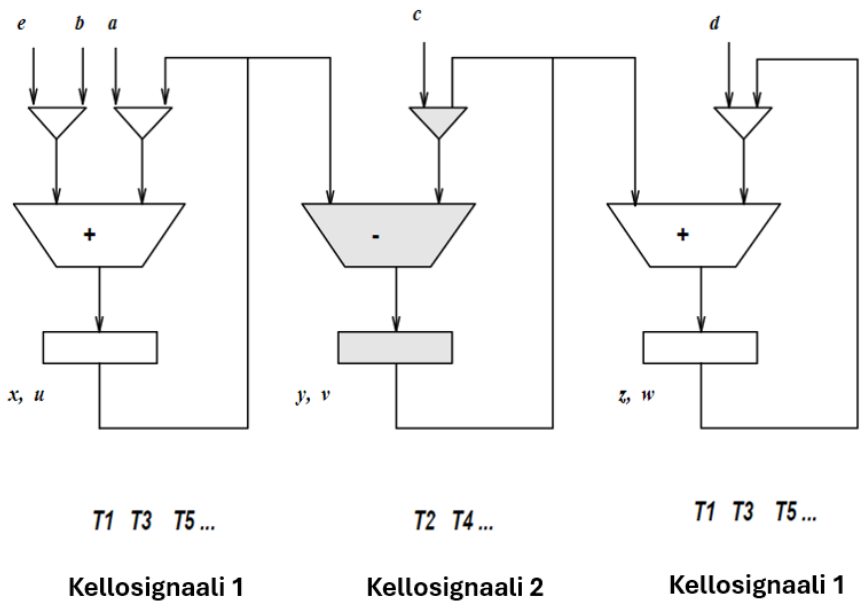
Laskentalogiikka muodostaa yhteen- ja vähennyslaskuketjun. Rakenteen idea on havainnollistaa laskentaketjua, joka voidaan jakaa vaiheisiin ja demonstroida vaiheiden ajoittamista kahden kellosignaalin avulla. Yhden kellosignaalin ajoitusta, rakennetta ja tehonkulutusta voidaan sitten verrata kahden kellosignaalin toteutukseen.

Kuvassa 3.3 MUX-komponenteilla (engl. Multiplexer, MUX) valitaan sisääntulosignaalit, jotka syötetään ALU-laskentayksikön (engl. Arithmetic logical unit, suom. arithmeettinen laskentayksikkö, ALU) läpi, jonka ulostulo syötetään edelleen seuraavaan laskentayksikköön. Laskentavaiheita hallitaan yhdellä kellosignaalilla.



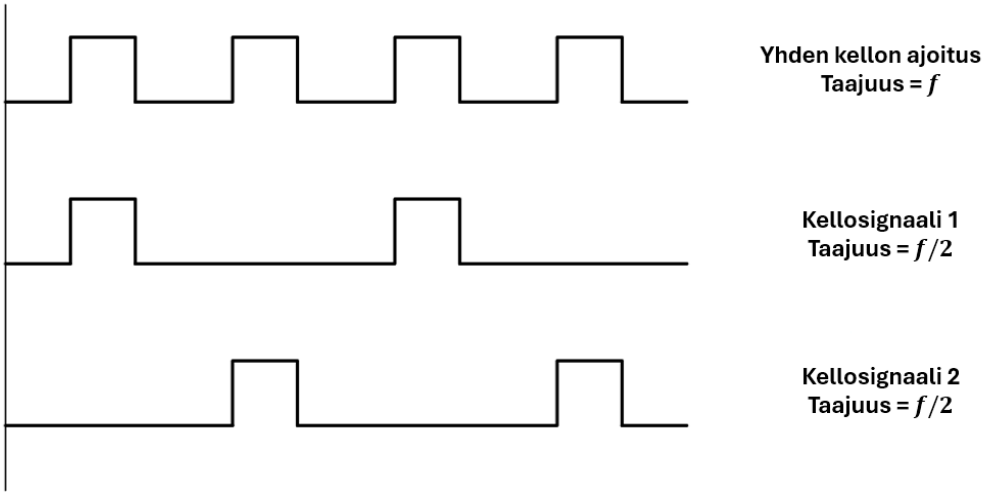
Kuva 3.3. *Yhden kellosignaalin ohjaama piiri, kuva lainattu julkaisusta [23].*

Kuvassa 3.4 laskentalogiikka on toteutettu kahden kellosignaalin tahdittamana. Kellosignaalit on jaettu vuoroittaisiin vaiheisiin, joissa yhteenlaskuoperaatiot toteutetaan samaan aikaan ja vähennyslaskuoperaatio vuorottain yhteenlaskuoperaatioiden kanssa. Laskentalogiikkaa ja täten komponentteja on nyt enemmän mutta samalla kellotaajuudella laskennasta suoriuduttaisiin nopeammin. Tällöin usean kellosignaalin hyödyntäminen mahdollistaa matalamman kellotaajuuden ja saman suoritustehon yhden kellosignaalin toteutukseen verrattuna.



Kuva 3.4 *Piirilogiikan summaoperaatiot on ajoitettu kellosignaalin 1 mukaan ja vähennysoperaatiot on ajoitettu kellosignaalin 2 mukaan. Muokattu lähteestä [23].*

Kuvasta 3.5 nähdään alkuperäinen kellosignaali ja sen ajoitukseen sovitetut kaksi erillistä kellosignaalia. Huomataan, että yhteen- ja vähennyslaskuoperaatioita ajoittavat kellosignaalit toimivat vuorotellen. Taajuus alkuperäiseen kellosignaaliin verrattuna on saatu puolitettua.



Kuva 3.5 *Kellosignaalien ajoituskaavio. Muokattu lähteestä [23].*

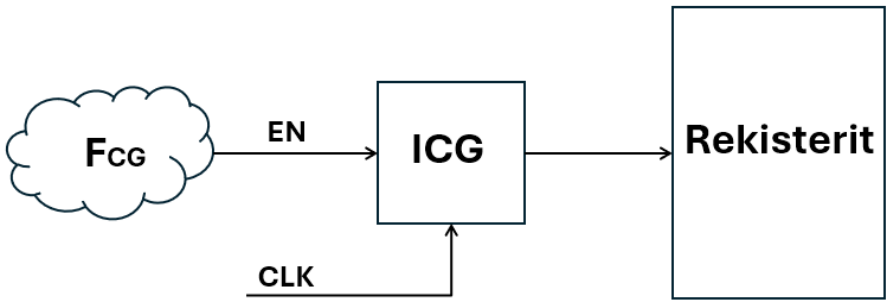
Usean kellosignaalin järjestelmien suunnittelu on monimutkaisempaa, sillä eri järjestelmien toimivuus keskenään täytyy varmentaa. Asynkronisilla usean kellosignaalin järjestelmillä voidaan kellosignaalien taajuus sovittaa (järjestelmien tarpeiden rajoissa) matalammaksi kuin synkronisilla järjestelmillä.

3.4 Clock gating -menetelmä

Transistoreista koostuvien logiikkaporttien läpi kulkee sähkövirtaa huolimatta siitä, onko niillä toiminnallista tarkoitusta kyseisen kellojakson aikana. Piireissä on paljon potentiaalia tehonkulutuksen optimoinnin kannalta, jos logiikkaportit voidaan kytkeä irti kellosignaalista, kun niitä ei tarvita. Clock gating -menetelmässä haasteena on logiikkaporttien virheettömän toiminnallisuuden takaaminen. [24]

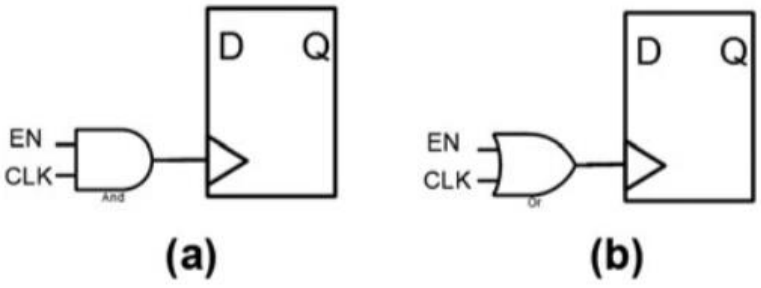
Clock gating -menetelmä keskittyy dynaamisen tehonkulutuksen minimoimiseen, vähentämällä kellosignaalista aiheutuvia häviöitä niissä osissa piiriä, jotka eivät ole aktiivisina. Synkronisen järjestelmän kelloverkon eri haarat jakautuvat eri toimintoihin liittyviin komponentteihin. CG-menetelmän ideana on jakaa järjestelmä niin, että haarautuva kellosignaali voidaan sulkea epäaktiivisten osien kohdalla, ja näin päästää kellosignaali ainoastaan tilamuutoksen alaisiin osiin järjestelmää. CG-menetelmän vaikutus tehonkulutukseen on suurin niissä järjestelmissä, joissa on paljon muistikomponentteja (rekistereitä), joiden ei tarvitse usein päivittää tilaansa tai ovat epäaktiivisia. [21]

Loogiset laskentapiirit koostuvat porttiverkoista ja rekisterikomponenteista, joiden laskentalogiikan lopputulos tallentuu rekisterin ulostuloon ja päivitetään kellosignaalin määräämän ajan välein. Kuvista 3.6–3.8 nähdään yksinkertainen kuvaus rekisterisiirtymätason (register transfer level, RTL) CG-rakenteesta, jolla voidaan ohjata kellosignaalin päivittymistä rekistereille. Kuvassa 3.6 on esiteltynä invertoitu (engl. Inverted clock gate, ICG) CG-rakenne, jonka tarkempi toiminta on kuvattu kuvissa 3.7 ja 3.8.



Kuva 3.6. *Laskentalogiikalta (Fcg) kulkeutuva EN-signaali ohjaa ICG-moduulissa kellosignaalin (Clk) pääsyä rekistereille. Muokattu lähteestä [21].*

Kuvan 3.6 ICG-moduuli koostuu kombinatorisista AND- ja OR-porteista (Tulevat sanoista ”ja” sekä ”tai”), jotka näkyvät kuvassa 3.7. Kombinatoriset portit päästävät läpi yksittäisen bitin riippuen sisääntuloihin tulevista biteistä. Kuvassa 3.7 vasemmalla oleva AND-portti toimii siten, että jos molemmat sen ulostuloista ovat ykkösiä, niin sen ulostulo on ykkönen. Muussa tapauksessa AND-portin ulostulo on nolla. Oikealla oleva OR-portti puolestaan näyttää ulostulossaan arvon yksi, jos toinen tai molemmat sen sisääntuloista ovat arvoltaan yksi.

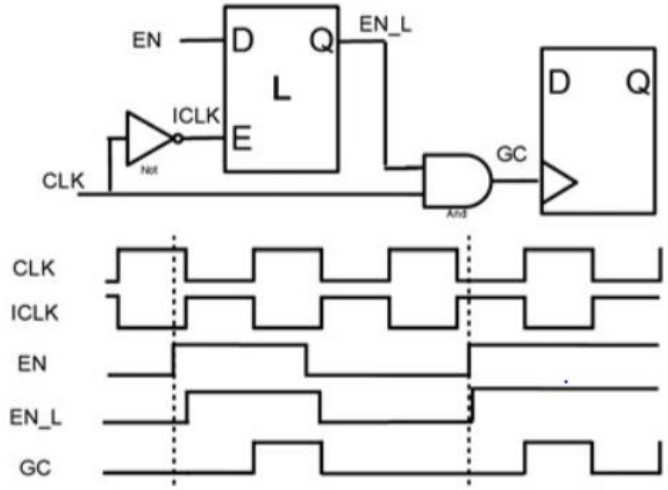


Kuva 3.7. *Kuvassa (a) on AND-portilla ja (b) on OR-portilla toteutettu CG-rakenne [21].*

Kun piirin kellosignaalin päivittyminen halutaan estää, täytyy sen tila myös tallentaa johonkin mikä pitää tietoa siitä onko piirin haara toiminnassa. Kuvassa 3.8 CG-rakenne on muodostettu kiikun (engl. Latch) ja AND-portin avulla. Kiikun ”muisti” eli ulostulo päivitetään silloin kun se saa päivitykseen liittyvän käskysignaalin. D-kiikku (engl. D-Flip-Flop,

DFF) päivittää tilansa jokaisen kellojakson aikana ja rekisterit koostuvat useista D-kiikuista [30].

Kuvan 3.8 kiikku (merkitty L) päivittää ICLK-signaalin tahdin mukaan, ennen varsinaisen kellosignaalin (CLK) seuraavan jakson alkua, ulostuloonsa EN-signaalin. EN_L-signaalilla ohjataan AND-porttia ja portin avulla hallitaan kellosignaalin kulkeutumista portista eteenpäin. Logiikan lopussa olevan rekisterin tilalla voisi esimerkin puolesta olla suurempi osa järjestelmää, joka voidaan esitetyillä rakenteilla katkaista kellosignaalista. Järjestelmässä kulkeva signaali ei tällöin reagoisi kellosignaalin muutoksiin.



Kuva 3.8 *AND-portilla ja sitä hallitsevan EN-signaalin tilan tallentavalla kiikulla muodostettu CG-rakenne ja sen signaalien aikakaavio [21].*

Mikropiirejä suunnitellaan laitteistotason rakenteisiin tarkoitetuilla ohjelmistoilla (engl. Hardware description language, HDL), jotka käännetään eri ohjelmistoilla toiminnallisesta kuvauksesta fyysisen rakenteen toteuttavaan muotoon. CG-rakenteiden implementointi tapahtuu pääosin suunnittelua automatisoivien työkalujen avulla (engl. Electronic design automation, EDA). [21 s.4–6]

Tutkimuksessa [24] analysoitiin langattoman Bluetooth-mikrokontrollerin tehonkulutusta, joka testattiin Clock gating ja gray code (binääri esitys, joka minimoi laskennassa bittimuutoksia) -menetelmien kanssa. Tuloksissa todetaan kellosignaaliverkon kuluttavan noin 70 % tehonkulutuksesta ja CG-metodilla onnistuttiin pienentämään kellosignaaliverkon kulutusta lähes 99 % ja koko järjestelmän kulutusta noin 86 %. Tulokset tukevat väitteitä kellosignaalin merkittävyydestä laitteen tehonkulutukselle.

Kellosignaaliverkostojen hallintaa voidaan automatisoida. ACG-logiikkaa (engl. Automatic clock gating) hyödynnetään julkaisussa [25], jossa dynaamiset tehohäviöt ja järjestelmän kellosignaalin jakeluverkko saadaan julkaisun mukaan lähes häviöttömiksi ACG-logiikkaa hyödyntämällä. ACG-arkkitehtuurissa luodaan kelloverkkoa kontrolloiva rakenne, jolla jokainen kellosignaalikomponentti voi lukea ja ymmärtää oman ”fanout” arvonsa (sisään tulevien signaalien määrä, joka kykenee tuottamaan ulostulon) ja sulkea kellosignaalin tarvittaessa. ACG-arkkitehtuurin hyödyntäminen saavuttaa tutkielman [25 s. 1] mukaan kokonaisuudessaan 17–50 % hyödyn tehonkulutuksessa.

Kellosignaalin tehonkulutuksen optimointi clock gating -menetelmää hyödyntäen saavuttaa merkittäviä säästöjä tehohäviöissä. Menetelmän hyödyntäminen ei ole kuitenkaan aina suoraviivaista, mikä osoittautuu eri kiikkurakenteita käytettäessä.

Esimerkiksi synkronisen järjestelmän bittien tilojen tallennukseen ja laskentalogiikan ajoitukseen käytetään transistoreista rakentuvia D-kiikkuja, jotka ovat mikropiireissä yleisimmin käytettyjä kiikkutyyppejä. Niistä muodostuu rekisteri, jonka ulostulo päivittyy joka kellojakson alussa. [31]

Kellosignaalin on todettu olevan merkittävä tehonkulutuksen lähde minkä vuoksi on kehitetty DET-kiikkuja (engl. Double edge trigerred flip-flop, DETFF), joiden tila päivitetään kellojakson nousevalla sekä laskevalla reunalla. DETFF-komponenttien etu on siinä, että ne kykenevät käsittelemään saman määrän dataa samassa ajassa kuin perinteiset D-kiikut mutta kellotaajuus voidaan puolittaa, jolloin saavutetaan tehonkulutuksen säästöjä. DETFF-komponenttien ja clock gating -menetelmän käytössä tulee vastaan haasteita, sillä komponenteissa esiintyy tällöin asynkronista käyttäytymistä. [31]

3.5 Power gating -menetelmä

Integroitujen piirien vähemmän aktiivisten osien kohdalla vuotovirrat ovat merkittävä osa kokonaistehonkulutuksesta. Mitä pienemmän kokoluokan teknologioihin mennään sitä merkittävämmäksi vuotovirrat muodostuvat. [32, luku 7] Elektroniset järjestelmät, joilla on eri alijärjestelmien välillä vähemmän osia samanaikaisesti käytössä tai niiden odotusajat ovat pitkiä, soveltavat parhaiten power gating -menetelmää. Power gating -menetelmällä saadaan muodostettua pienin vuotovirta, kun mainitut piirteet järjestelmälle toteutuvat. [32 s. 278] Power gating -menetelmä hyödyntää transistoritasolla unitilatransistoreja (engl. sleep transistors), jotka suunnitellaan jännitelähteen ja piirin väliin. Unitilatransistoreilla voidaan sulkea toimimaton piiri irti jännitelähteen ja maan väliltä. [32 s. 280]

Clock gating -menetelmällä päästään pitkälle matalan käyttöasteen piirien tehonkulutuksen optimoinnissa mutta vuotovirtojen merkityksen kasvaessa, power gating -menetelmän hyödyntämisellä on kriittinen rooli tehonkulutuksen optimoinnissa. [33 s. 69] Power gating -menetelmä on idealtaan hyvin samankaltainen clock gating -menetelmän kanssa. Toimimaton piirin osa halutaan eristää toimintaa ohjaavasta kellosignaalista (Clock gating) tai käyttöjännitteestä (Power gating).

Ero clock gating ja power gating -menetelmien toteutuksessa on se, että power gating vaatii monimutkaisempia ratkaisuja transistoritasolla sekä ylimääräistä logiikkaa, jolla eristetyn piirin sisältämät tiedot tallennetaan ulkoiseen muistiin ja tarpeessa siirretään takaisin. Power gating -menetelmä on ylimääräisten muistikomponenttien sekä kontrollilogiikan vuoksi monimutkaisempi ja kalliimpi toteuttaa kuin clock gating -menetelmä. Pienenevien transistorien käytössä power gating -menetelmän rooli tehonkulutuksen optimoinnissa kasvaa [32, 33].

3.6 Dynaaminen tehonhallinta

Järjestelmäpiireillä toimivilla laitteilla on tavallisesti monia eri toimintoja, jotka eivät vaadi järjestelmältä jatkuvaa toimintaa tai korkeinta suoritustehoa. Aikaisemmin mainittujen menetelmien, kuten clock gating ja power gating:in lisäksi laitteissa voidaan käytön aikana muuttaa jännitettä ja taajuutta prosessoreiden ja alijärjestelmien tarpeen mukaan. Jännitteen ja taajuuden muuttamista kutsutaan nimellä dynaaminen jännitteen ja taajuuden skaalaus (engl. Dynamic voltage and frequency scaling, DVFS).

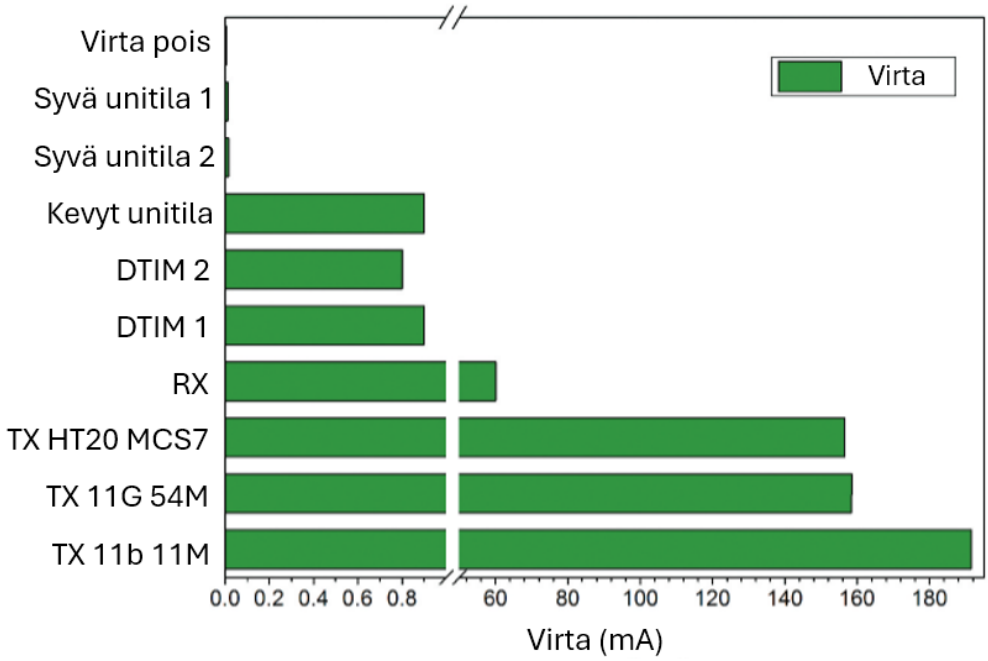
Dynaamisella tehonhallinnalla (engl. Dynamic power management, DPM) prosessori tai alijärjestelmä siirretään pienemmän tehonkulutuksen tilaan ja sen tärkeimmät oheiskomponentit toimivat matalammalla taajuudella. Vaihtoehtoisesti DPM asettaa prosessorin unitilaan. [34] Unitilat toteutetaan power gating ja clock gating -menetelmillä ja matalamman suoritustehon tilat DVFS:n avulla. Artikkelissa [34] huomautetaan, että DPM-tekniikka vaatii prosessoreilta ylimääräistä energiaa eri tilojen siirtymisen välillä.

Toimintajakso (engl. Duty cycle, DC) viittaa laitteen toimintaan, jossa laite on oletuksena unitilassa ja herätetään ajoittain tarkistamaan ympäristön tila ja toteuttamaan odottavia tehtäviä. Toimintajakson tahti (DC-rate) viittaa siihen, kuinka usein ja lyhyillä väliajoilla laite herätetään. Korkeampi toimintajakson tahti viittaa lyhyempiin unitila-aikoihin. Korkeammalla toimintajakson tahdilla saadaan laitteen tarjoamasta palvelusta sulavampi, mutta se johtaa suurempaan tehon kulutukseen. [35]

Suunnittelijan on hyvä huomioida tilasiirtymien aiheuttama tehohäviö ja välttää keskittymästä pelkästään suorituksen aikaiseen kulutukseen. Jos esimerkiksi ajoitetaan lyhyitä

ajanjaksoja, joiden välillä prosessori siirretään unitilaan ja herätetään tilasta, on hyvä huomioida tilojen siirtymisen aiheuttama tehonkulutus toiminnan aikaisen kulutuksen lisäksi. Yksi tämän kaltainen operaatio on laitteen kommunikointi tukiaseman (tai muun lähettimen) kanssa, jossa ajoittaisella yhteyksien kuuntelulla taataan toimiva yhteys laitteiden välillä.

Tutkimuksessa [36] esiteltiin 40 nm CMOS-teknologialla toteutettu matalan tehonkulutuksen järjestelmäpiiri. Järjestelmäpiirin suunnittelussa hyödynnettiin DPM-menetelmiä. Kuvassa 3.9 näkyy järjestelmäpiirin virrankulutus eri tehonkulutuksen tiloissa.



Kuva 3.9. *WiFi-järjestelmäpiirin eri toimintatiloja ja niiden virrankulutus. Muokattu lähteestä [36 s. 16].*

Kuvan 3.9 kaaviosta nähdään, kuinka unitilojen sekä matalamman suoritustehon tilojen suunnittelun vaikutus laitteen tehonkulutukselle. Järjestelmäpiirissä mainitaan hyödynnettynä dynaaminen jännitteen ja taajuuden skaalaus, Power gating, Clock gating sekä useiden jännitetasojen hyödyntäminen. Tutkimuksen [36] järjestelmäpiirissä hyödynnettiin myös muita suunnittelun menetelmiä, tehonkulutuksen ja suorituskyvyn optimoimiseksi.

DVFS-tekniikka parantaa tehonkulutusta madaltamalla prosessorin käyttöjännitettä sekä taajuutta. DVFS-tekniikassa taajuuden muuttaminen aiheuttaa epäsuorasti tehohäviöitä. [34] Artikkeli [37 s.2] esittelee kolme tärkeää komponenttia jännitteen dynaamiseen säätelyyn. Käyttöjärjestelmän täytyy kyetä skaalaamaan prosessorien nopeutta, joten nopeuden säätelypiiri tarvitaan minimijännitteen syöttämiseksi. Toiseksi tarvitaan mikropro-

sessori, jonka käyttöalue jännitteen suhteen on mahdollisimman laaja. Kolmanneksi prosessorien taajuuden säätelyyn vaaditaan ohjelmistotason toteutus, sillä artikkelin [37] mukaan laitteiston arkkitehtuurilla ei voida erotella onko prosessorin toteuttama käsky osana vaativaa laskentaa vai onko käskyn suoritusaika vähemmän kriittinen laitteen toiminnan kannalta.

Artikkelin [37] vertailuanalyysi näytti vähemmän laskentaa vaativille ääni- ja käyttöliittymätoiminnoille 4.5- ja 3.5-kertaisen parannuksen energian kulutukseen, kun hyödynnettiin dynaamista jännitteen skaalausta. Toisaalta artikkelissa todetaan, että paljon laskentaa vaativa MPEG-testaus (Moving pictures experts group, MPEG, datavirtastandardi) todensi 11 % parannuksen tehonkulutuksessa dynaamisen jännitteen skaalauksen kanssa.

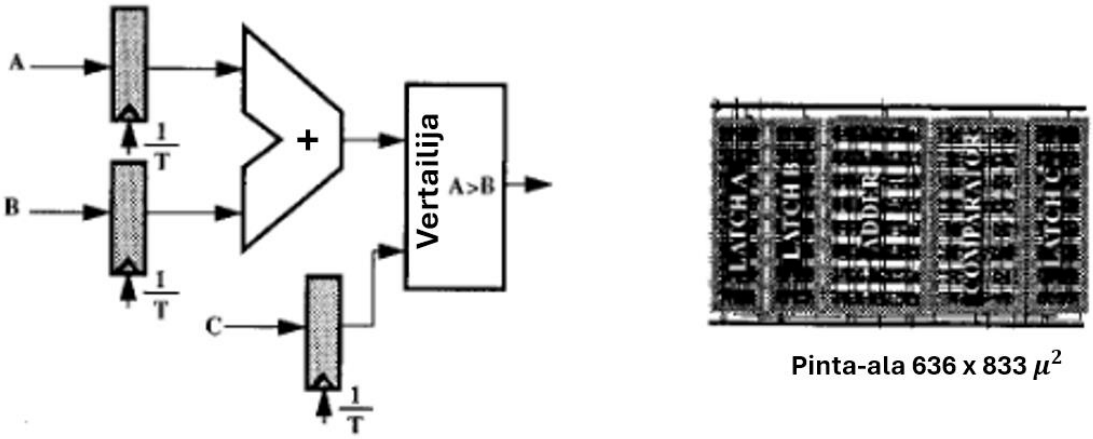
Järjestelmäpiirien tehonhallintayksikkö (Power management unit, PMU) tarjoaa rajapinnan prosessoreille, jonka avulla prosessorit kyetään asettamaan matalamman tehonkulutuksen tilaan. Tiloja voidaan määritellä useita, riippuen laitteen toiminnasta ja vaaditusta reagointinopeudesta. Näiden tilojen valinta voidaan toteuttaa eri logiikalla riippuen esimerkiksi siitä, kuinka kauan aikaa viimeisestä lepotilasta on. [38 s.1]

Tutkimus [38] esittää huomion, etteivät moderneille käyttöjärjestelmille suunnitellut dynaamiset tehonhallinnan käytännöt sovellu tehokkaasti prosessoreille, jotka on suunniteltu sulautetuille järjestelmille. Ongelma esiintyy prosessorien keskeytyspyyntöjen (engl. Interrupt request) havainnoinnissa. Yleiset tehonhallinnan käytännöt eivät tutkimuksen [38] mukaan välttämättä kykene tunnistamaan käyttöliittymälaitteista tulevia keskeytyspyyntöjä, kun prosessorit ovat syvän unen tilassa. Kyseinen tutkimus tarjoaa 25 % tehonkulutusta parantavan tehonhallintamenetelmän sulautettujen järjestelmien prosessoreihin.

Tutkimuksen huomio eri käyttökohteiden eroavaisuuksista, on hyvä muistaa järjestelmäpiirien suunnittelussa. Järjestelmäpiirien rakenne vaihtelee käyttökohteiden ja laitteen tarpeiden mukaan, jolloin edes yleisimpien tehonkulutuksen optimointimenetelmien vaikutukset eivät ole välttämättä helposti ennustettavissa. Samalla on huomioitavaa, että tehonkulutusta aiheuttavan käyttöjännitteen ja kellosignaalin taajuuden madaltaminen vaikuttavat suoritustehoon. Suunnittelijan täytyy täten suhtautua kriittisesti erilaisiin suunnittelumenetelmiin ja harkita niitä aina tapauskohtaisesti.

3.7 Liukuhihnoitus & rinnakkaisuus

Tutkimuksessa [22] selvitettiin kuvissa 3.11 ja 3.12 näkyvän logiikan avulla arkkitehtuurillisia keinoja, jotka mahdollistavat käyttöjännitteen madaltamisen ja samalla ylläpitävät suoritustehoa. Kuvassa 3.10 on esitelty lähtökohtainen arkkitehtuuri, jonka jännite- ja taajuusarvoja sekä tehonkulutusta, on verrattu rinnakkaisuus ja liukuhihnoitus arkkitehtuureihin. Kuvan 3.10 datapolku koostuu ALU-laskentayksiköstä, rekistereistä ja arvojen vertailijasta. Sisääntulot A ja B summataan yhteen ja verrataan sisääntulon C kanssa.



Kuva 3.10. *Logiikka koostuu signaalien tilat tallentavista rekistereistä, laskenta ALU-yksiköstä (kuvassa summain "+") sekä arvojen vertailuyksiköstä. Oikealla "Area" kuvaa rakenteessa käytettyjen komponenttien viemää tilaa piiristä. Kuva on muokattu lähteestä [22].*

Analysointiin soveltuva teknologia on tutkimuksessa [22] oletettu 2.0 µm kokoluokkaan. Datapolun suurimman mahdollisen suoritustehon huonoin suoritusaika on arvioitu tutkimuksessa noin 25 ns kellojaksoon, kun käytetään 5 V käyttöjännitettä. Tutkimuksen mukaan jännitettä ei voida kyseisessä tilanteessa madaltaa ilman suoritusajan hidastumista. [22]

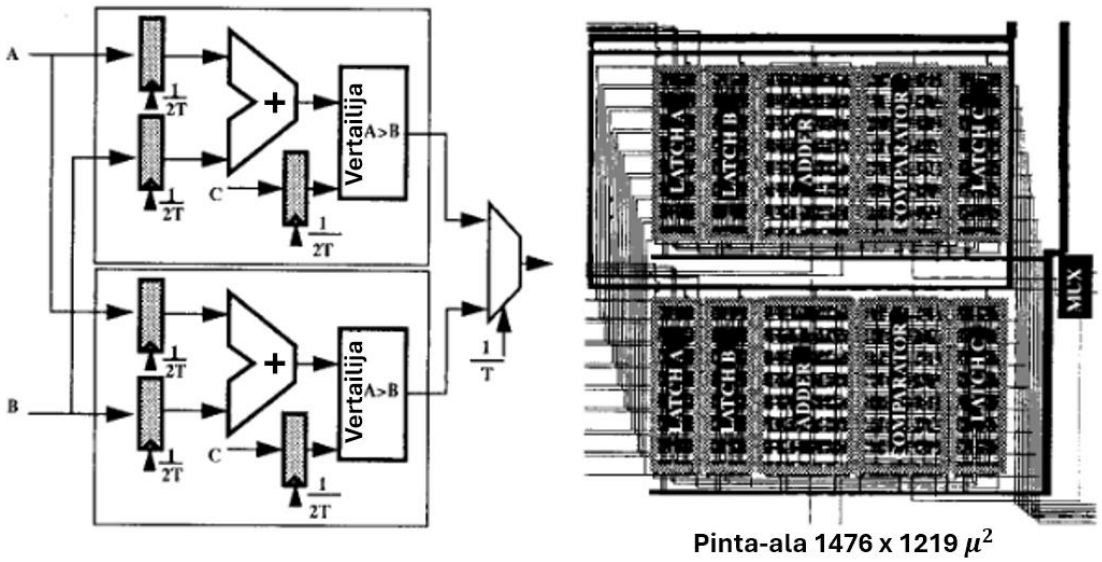
Tutkimuksessa käytettiin kuvan 3.10 kytkentää referenssinä, jonka teho määräytyy kaavalla (3), joka pohjautuu luvussa 3.1 esitettyyn dynaamisen tehohäviön lausekkeeseen (2).

$$P_{ref} = C_{ref}V^2{}_{ref}f_{ref} \qquad (3)$$

P_{ref} kuvaa referenssi kytkennän dynaamisista tehohäviöistä koostuvaa tehonkulutusta. C_{ref} on kokonaiskapasitanssi, V_{ref} on käyttöjännite ja f_{ref} on järjestelmän kellosignaalin taajuus. Yksi ratkaisu käyttöjännitteen alentamiselle on rinnakkaisuus, jossa kytkentä kopioidaan kuvan 3.11 mukaisesti. Tällöin järjestelmä kykenee samaan suoritustehoon kuin referenssikytkentä, mutta rinnakkaisten kytkentöjen laskentatahti eli taajuus voidaan hidastaa puoleen aikaisemmasta. Datapolun kellojakson vaatimus muuttuu siis 25

ns jaksosta 50 ns jaksoon. Tällöin tehon lausekkeen mukaan jännite voidaan madaltaa 5:stä 2.9 V:iin. [22]

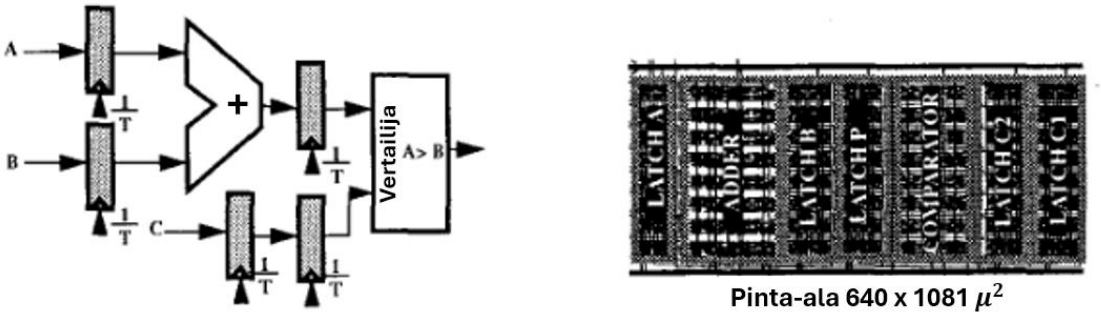
Kuvassa 3.11 piirilogiikan monistaminen mahdollistaa samanaikaista datan käsittelyä. Samanaikaisuus mahdollistaa järjestelmän taajuuden puolittamisen samalla suoritusteholla kuin alkuperäinen piiri. Tällöin siis kellosignaalin aiheuttama tehohäviö pienenee mutta piiri kykenee suorittamaan laskennan yhtä nopeasti kuin referenssipiiri.



Kuva 3.11. *Rinnakkainen rakenne (vas.) ja pinta-ala, jonka komponentit vievät piiristä (oik.). Kuva on muokattu lähteestä [22].*

Rinnakkaisuus mahdollistaa käyttöjännitteen madaltamisen samalla suoritusteholla. Tutkimuksessa kuitenkin huomioidaan, että rinnakkaisuudella on käytettäviä komponentteja kasvattava vaikutus, joka samalla lisää myös tarvittavien johtimien määrää ja vaikuttaa tehonkulutukseen sitä kasvattaen. Mikrometriteknologiaa pienemmillä teknologioilla johdinten kapasitiivinen vaikutus myös kasvaa, joka kasvattaa tehonkulutusta. [22]

Rinnakkaisuuden lisäksi toinen yleisesti käytössä oleva arkkitehtuuri on liukuhihnoitus. Liukuhihnoitusta havainnollistettiin tutkimuksessa [22] kuvan 3.12 mukaisella rakenteella. Kuvassa 3.12 liukuhihnoitus on toteutettu sijoittamalla ylimääräiset rekisterit laskentaa suorittavan ALU-yksikön ja komparaattorin väliin. Kellotaajuus on asetettava järjestelmässä hitaimman datapolun mukaan, joka määräytyy sen mukaan kauanko signaalilla kestää kulkea yhdeltä rekisteriltä seuraavalle. Liukuhihnoituksella järjestelmä voidaan jakaa vaiheisiin, jonka avulla kriittinen polku eli datan hitain kulkureitti saadaan lyhyemmäksi. Referenssilogiikkaan verrattuna kriittinen polku kulki aikaisemmin laskentayksikön ja vertailuyksikön läpi. Liukuhihnoituksessa nämä kaksi yksikköä saadaan eroteltua omiin vaiheisiinsa.



Kuva 3.12. *Liukuhihnoitus: rekistereillä eri vaiheisiin eroteltu laskentalogiikka (vas.) ja logiikan komponenttien käyttämä pinta-ala (oik.). Kuva on muokattu lähteestä [22].*

Kuvan 3.12 tapauksessa kriittinen polku määräytyy sen mukaan aiheuttaako vertailija- vai ALU-yksikkö enemmän viivettä järjestelmässä. Verrattuna referenssirakenteeseen kriittinen polku on nyt ajallisesti lyhyempi, koska nyt ALU ja vertailija on eroteltu rekistereiden avulla. Kellosignaalin taajuutta voidaan lyhyemmän kriittisen polun ansiosta madaltaa. Käyttöjännite saadaan tässäkin kytkennässä madallettua 2.9 V:iin. Etuna liukuhihnoituksessa, kun verrataan rinnakkaisuuteen, on pienempi pinta-ala. Pienempi pinta-ala tarkoittaa, että tarvittavien komponenttien määrä on myös pienempi. [22]

Rinnakkaisuus- ja liukuhihnoitusmenetelmien yhdistäminen tuo tutkimuksen [22] päätelmien mukaan parhaan hyödyn verrattuna vain toisen menetelmän hyödyntämiseen. Rinnakkaisuus saavutti 0.36-kertaisen, liukuhihnoitusmenetelmä 0.39-kertaisen ja yhdistelmärakenne 0.2-kertaisen tehonkulutuksen alkuperäiseen kytkentään verrattuna.

Kaikkien rakenteiden suhteen tutkimus [22] huomioi, ettei rinnakkaisuuden ja liukuhihnoituksen käyttö sovellu rekursiivisiin järjestelmiin kovin hyvin. Tutkimuksessa [22] mainittuja rekursiivisia järjestelmiä ovat muun muassa IIR-filtterit (engl. Infinite impulse response), epälineaariseen laskentaan keskittyvät järjestelmät sekä adaptiivista pakkausta toteuttavat algoritmit.

Julkaisu [22] on elektroniikan kehityksen suhteen jo aika vanha lähde vuodelta 1992. Teos [4 s. 71] vuodelta 2014 esittää edellä annettuja havaintoja tukevia tuloksia rinnakkaisuuden ja liukuhihnoituksen hyödyistä. Dynaamisen tehonkulutuksen yhtälö (2) ei ole julkaisun [22] iästä riippumatta muuttunut. Tehonkulutukseen vaikuttaa merkittävästi käyttöjännite V_{dd} ja sen alentaminen vaikuttaa teoksen [4 s. 21] mukaan CMOS-porttilogiikan viiveisiin negatiivisesti. Mainitussa teoksessa suorituskyvyn ylläpidoksi ehdotetaan rinnakkaisuus- ja liukuhihnoitusrakenteita. Rinnakkaisuus ja liukuhihnoitus mahdollistavat täten käyttöjännitteen alentamisen suorituskykyä parantamalla.

4. YHTEENVETO JA PÄÄTELMÄT

Järjestelmäpiirit muodostavat suurimman osan tai kaikki elektronisen laitteen toiminnoista. Useat prosessorit, muistikomponentit ja muut laitteen komponentit integroidaan yhdelle sirulle, joka mahdollistaa elektroniikan koon, tehonkulutuksen ja suoritustehon optimoinnin.

Monet laitteet tarvitsevat kokovaatimusten, lämpöominaisuuksien, tehonkulutuksen tai käyttöiän puolesta järjestelmäpiirejä. Tehonkulutuksella on rooli lämpöominaisuuksien, käyttöiän ja yhä pienenevän elektroniikan suhteen, joten sen optimointi on kasvavassa roolissa järjestelmäpiirien ja matalaa tehonkulutusta vaativien laitteiden suunnittelussa. Mainittuihin laitteisiin, jotka vaativat pientä kokoa, käyttävät rajallista virtalähdettä tai vaativat pitkää käyttöikää, kuuluvat muun muassa IoT-laitteet, erilaiset anturiverkot, tukiasemat sekä mobiililaitteet, kuten älypuhelimet ja älykellot.

Tehonkulutus jaetaan staattiseen ja dynaamiseen tehohäviöön, joista dynaaminen viittaa toiminnallisiin bittien arvojen muutoksista aiheutuviin tehohäviöihin, ja staattinen viittaa transistoreissa tapahtuviin vuotovirtoihin. Järjestelmää ajoittava kellosignaali ja käyttöjännite ovat keskeisimmät tekijät dynaamisten tehohäviöiden minimoinnissa. Kellosignaali kuluttaa tehoa laitteen toiminnasta riippumatta. Kellosignaali on myös järjestelmän nopeimmin toimiva signaali, joten sen käytön optimoinnilla on paljon merkitystä laitteen tehonkulutukselle. Jännitteen vaikutus on merkittävin, sillä eri jännitetasoja vastaavien bittien arvojen vaihtelu ja käyttöjännitteestä aiheutuvat tehohäviöt ovat verrannollisia jännitteen toiseen potenssiin.

Transistorien pienenevä koko on kasvattanut staattisia häviöitä, joihin reagoidaan transistoritason ratkaisuilla. Useat lähteet tarjoavat ristiriitaisia havaintoja siitä ovatko dynaamiset vai staattiset häviöt merkittävämpiä. Järjestelmäpiirit eroavat toisistaan käyttökohteen ja tavoiteltujen ominaisuuksien puolesta. Varmaksi voidaan sanoa, että elektroniikan pienempi koko ja kasvavat vaatimukset suoritusteholle kasvattavat sekä dynaamisten että staattisten häviöiden minimoinnin merkitystä.

Laitteen tehonkulutus on helpointa optimoida dynaamisten tekijöiden suhteen ja yleisimpiin menetelmiin kuuluu laitteen käytön aikainen dynaaminen jännitteen ja taajuuden skaalaus (DVFS). Erilaisia uni- ja virransäästötiloja toteutetaan järjestämällä järjestelmän osia ryhmiin, joiden kellosignaaliverkko voidaan asettaa matalaksi tai tarpeessa järjestelmän osa voidaan kytkeä irti käyttöjännitteestä kokonaan. Kellosignaalin rajoitusmenetelmää kutsutaan nimellä ”Clock gating” ja jännitteestä irtikytkentää nimellä ”Power

gating”. Clock gating on rakenteeltaan yksinkertainen ja yleisin käytössä oleva tehonkulutuksen optimointimenetelmä. Tietokoneavusteiset suunnitteluohjelmat ovat pitkälle automatisoineet clock gating -menetelmän käyttöä. Power gating -menetelmä saavuttaa merkittäviä säästöjä mutta on suunnitteluarkenteeltaan monimutkaisempi ja kalliimpi, sillä se vaatii ulkoista muistilogiikkaa ja tiedonsiirtoon suunniteltua piirilogiikkaa.

Matalamman tehonkulutuksen tiloihin liittyy myös toiminnallinen jaksotus (engl. Duty cycle), jossa järjestelmän osa herätetään unitilasta tasaisin väliajoin suorittamaan tarpeellinen toiminta, kuten verkkoyhteyden ylläpito. Kyseisten toimintojen ei ole tarpeellista olla jatkuvasti päällä, mutta toimintajaksoihin liittyy tarkkaa suunnittelua oikean toiminnallisuuden takaamiseksi.

Järjestelmien ajoitus voi olla synkronista, eli globaalin kellosignaalin ajoittamaa. Asynkroninen järjestelmä toimii omalla kellosignaalillaan tai on kokonaan kellosignaalista riippumaton. Asynkronisella suunnitelulla eri toiminnoista vastaavien alijärjestelmien kellosignaali voidaan optimoida kyseisen alijärjestelmän tarpeisiin, jolla mahdollistetaan matalammat kellosignaalin taajuudet ja näin matalampi tehonkulutus.

Unitilan toteuttavista menetelmistä eli clock gating ja power gating -menetelmistä, dynaamisesta jännitteen ja taajuuden skaalauksesta (DVFS), toimintajaksoista (Duty cycle) sekä niitä hallinnoivista menetelmistä (DPM), muodostuu laitteen käytönaikaisten toimintojen tehonkulutusta optimoiva kokonaisuus. Tehonkulutuksen tilojen valinta, siirtyminen ja toteutus tarvitsevat kontrollilogiikkaa, joka kykenee käytön aikana tunnistamaan tehonkäytön tarpeet ja aktivoimaan tehonhallintaa toteuttavia menetelmiä. On huomioitavaa, että eri menetelmien käyttö ei ole käyttökohteesta toiseen aina selvää. Yksi esimerkki esiintyi yleisten käyttöjärjestelmissä hyödynnettyjen tehonhallintaohjelmien puutteissa, kun niitä sovellettiin sulautettujen järjestelmien käyttökohteissa.

Suoritusteho ja tehonkulutus vaikuttavat toisiinsa, jolloin erilaiset suoritustehoa parantavat arkkitehtuurit auttavat mahdollistamaan laitteen suoritustehon ylläpidon ja tehonkulutuksen minimoinnin. Rinnakkaisuus on yksinkertainen tapa mahdollistaa matalammat taajuudet ja käyttöjännite, joka toteutuu laskentalogiikkaa monistamalla. Menetelmän vaikutukset eivät ole suoraviivaisia ylimääräisen laskentalogiikan puolesta, joten harkintaa tulee soveltaa sitä käytettäessä. Rinnakkaisuuden lisäksi liukuhihnoitusta hyödynnetään laskentalogiikan eri vaiheisiin jaottelussa. Jaottelu mahdollistaa matalammat kellosignaalin taajuudet lyhyemmän kriittisen polun vuoksi, ja menetelmän toteutus vaatii rinnakkaisuuteen verrattuna vähemmän komponentteja sekä tilaa.

Uusilta laitteilta vaaditaan parempaa suoritustehoa, pienempää kokoa, pidempää käyttöikää ja parempaa akunkestoa. Jatkuvasti kehitetään uusia teknologioita, kuten XR-laseja tai nopeampia verkkoyhteysteknologioita. Uudet teknologiat ja jatkuva suoritustehon kasvu vaativat laitteilta parempaa optimointia ja näitä edistysaskeleita rajoittavat koon, lämpötilan, käyttöiän ja virtalähteen rajoitteet. Rajoitteiden ratkaisuksi järjestelmäpiireille ominainen integrointi ja matalan tehonkulutuksen menetelmät ovat keskeisessä asemassa, jolloin matalan tehonkulutuksen painottaminen järjestelmäpiirien suunnittelussa kasvattaa merkitystään. Yritysten ja suunnittelijoiden on hyvä perehtyä matalaan tehonkulutukseen ja siihen liittyviin menetelmiin elektroniikkalaitteiden jatkuvan kehityksen mahdollistamiseksi.

Läpi käydyt matalan tehonkulutuksen menetelmät ovat vain pintaraapaisu aiheesta, joten seuraavaksi mainittuina mahdollisia lukijaa kiinnostavia käsitteitä jatkolukemista varten: engl. Tunnel field effect transistor (TFET), adiabatic logic, level shifter, gray code, minimal leakage vector sekä adaptive body biasing.

LÄHTEET

[1] Xu, Ningyue. “Energy Consumption Forecasting: A Trend Analysis Based on Global Data from 2000-2020.” ACM International Conference Proceeding Series, ACM, 2023, s. 51–146, Saatavilla (Viitattu 28.1.2025): https://doi.org/10.1145/3653724.3653750.

[2] Yokoyama, André M., et al. “Investigating Hardware and Software Aspects in the Energy Consumption of Machine Learning: A Green AI-Centric Analysis.” Concurrency and Computation, vol. 35, no. 24, 2023, Saatavilla (Viitattu 20.1.2025): https://doi.org/10.1002/cpe.7825.

[3] ROGERS, Ethan A., et al. “Intelligent efficiency technology and market assessment.” American Council for an Energy-Efficient Economy (ACEEE), 2017. Saatavilla (Viitattu 10.4.2025): https://www.aceee.org/sites/default/files/publications/researchreports/ie1701.pdf.

[4] Tan, Nianxiong, et al. “Ultra-Low Power Integrated Circuit Design*:* Circuits, Systems, and Applications.” 1st ed. 2014, Springer, 2014, Saatavilla (Viitattu 14.4.2025): https://doi.org/10.1007/978-1-4419-9973-3.

[5] Chakravarthi, Veena S., and Shivananda R. Koteshwar. “System on Chip (SOC) Architecture.” Springer Nature Switzerland, 2023, s. 1–48, Saatavilla (Viitattu 26.2.2025): https://doi.org/10.1007/978-3-031-36242-2_3.

[6] Qualcomm, “Qualcomm Snapdragon 600 processors”, https://www.qualcomm.com, 2013, Saatavilla (Viitattu 25.4.2025): https://www.qualcomm.com/content/dam/qcomm-martech/dm-assets/documents/snapdragon-600-processor-product-brief.pdf.

[7] S. Lammi, “How to become a System-on-Chip design expert?”, https://blogs.tuni.fi, 30.8.2022, Saatavilla (Viitattu 9.4.2025): https://blogs.tuni.fi/cs/teaching/how-to-become-a-system-on-chip-design-expert/.

[8] Saha, Debasri, et al. “SoC: A Real Platform for IP Reuse, IP Infringement, and IP Protection.” VLSI Design, vol. 2011, Saatavilla (Viitattu 14.3.2025): https://doi.org/10.1155/2011/731957.

[9] Chen, Yen-Kuang, and S. Y. Kung. “Trend and Challenge on System-on-a-Chip Designs.” Journal of Signal Processing Systems, vol. 53, no. 1–2, 2008, Saatavilla (Viitattu 1.3.2025): https://doi.org/10.1007/s11265-007-0129-7.

[10] Pospiech, F., and S. Olsen. “Embedded Software in the SoC World. How HdS Helps to Face the HW and SW Design Challenge [Hardware Dependent Software].” Proceedings of the IEEE 2003 Custom Integrated Circuits Conference, 2003, Saatavilla (Viitattu 7.3.2025): https://doi.org/10.1109/CICC.2003.1249480.

[11] Savage, W., et al. “IP Reuse in the System on a Chip Era.” Proceedings 13th International Symposium on System Synthesis, vol. 2000-, IEEE, 2000, Saatavilla (Viitattu 3.4.2025): https://doi.org/10.1109/ISSS.2000.874022.

[12] Krishnamoorthy, Ramesh, et al. “Systematic Approach for State-of-the-Art Architectures and System-on-Chip Selection for Heterogeneous IoT Applications.” IEEE Access, vol. 9, 2021, Saatavilla (Viitattu 1.3.2025): https://doi.org/10.1109/ACCESS.2021.3055650.

[13] Chakravarthi, Veena S. “A Practical Approach to VLSI System on Chip (SoC) Design : A Comprehensive Guide.” Second edition., Springer, 2022. (Viitattu 7.3.2025).

[14] Nokia, ReefShark System-on-Chip technology, Saatavilla: (Viitattu 7.4.2025): https://www.nokia.com/mobile-networks/ran/reefshark/.

[15] SUNNYVALE, Calif. Design & Reuse, ”MoSys Inc. and Fujitsu Limited Extend 1T-SRAM Technology Agreement to 90nm ASIC/SoC Designs; Targets Portable Consumer Applications including Digital Cameras and Video Camcorders”, 2005, Saatavilla (Viitattu 7.4.2025): https://www.design-reuse.com/news/10206/mosys-fujitsu-limited-extend-1t-sram-technology-agreement-90nm-asic-soc-designs-targets-portable-consumer-applications-digital-cameras-video-camcorders.html.

[16] Suryono, Suryono, et al. “System on Chip (SOC) Wi-Fi Microcontroller for Multistation Measurement of Water Surface Level Using Ultrasonic Sensor.” Journal of Physics: Conference Series, vol. 1025, no. 1, 2018, Saatavilla (Viitattu 6.3.2025): https://doi.org/10.1088/1742-6596/1025/1/012029

[17] Liu, Xinjian, et al. “A Sub-μW Energy-Performance-Aware IoT SoC With a Triple-Mode Power Management Unit for System Performance Scaling, Fast DVFS, and Energy Minimization.” IEEE Journal of Solid-State Circuits, vol. 59, no. 7, 2024, Saatavilla (Viitattu 6.3.2025): https://doi.org/10.1109/JSSC.2024.3350449.

[18] Sridhara, S. R., et al. “Microwatt Embedded Processor Platform for Medical System-on-Chip Applications.” IEEE Journal of Solid-State Circuits, vol. 46, no. 4, 2011, Saatavilla (Viitattu 6.3.2025): https://doi.org/10.1109/JSSC.2011.2108910.

[19] Flynn, Michael & Hung, Patrick. (2005). Microprocessor Design Issues: Thoughts on the Road Ahead. Micro, IEEE. 25, 2005 (Viitattu 4.3.2025).

[20] Satyajit, Sinha. IoT-analytics, “State of IoT 2024: Number of connected IoT devices growing 13% to 18.8 billion globally”, 3.9.2024, Saatavilla (Viitattu 2.5.2025): https://iot-analytics.com/number-connected-iot-devices/

[21] Pouiklis, Georgios, and Georgios Ch Sirakoulis. “Clock Gating Methodologies and Tools: A Survey.” International Journal of Circuit Theory and Applications, vol. 44, no. 4, 2016, Saatavilla (Viitattu 23.3.2025): https://doi.org/10.1002/cta.2107.

[22] Chandrakasan, A. P., et al. “Low-Power CMOS Digital Design.” IEEE Journal of Solid-State Circuits, vol. 27, no. 4, 1992, Saatavilla (Viitattu 20.3.2025): https://doi.org/10.1109/4.126534.

[23] Papachristou, C. A., et al. “A Multiple Clocking Scheme for Low-Power RTL Design.” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 7, no. 2, 1999, Saatavilla (Viitattu 27.3.2025): https://doi.org/10.1109/92.766754.

[24] Hang Suan Wang, Asral Bahari Jambek, Kean Hong Boey, Rizalafande Che Ismail, Fazrul Faiz Bin Zakaria; “Low-power bluetooth microcontroller design using clock gating and gray code state encoding.” AIP Conf. 2024, Saatavilla (Viitattu 25.1.2025): https://doi.org/10.1063/5.0192105 sivut1-6.

[25] Lee, Jae-Gon, et al. “Fully Automated Hardware-Driven Clock-Gating Architecture With Complete Clock Coverage for 4 Nm Exynos Mobile SOC.” IEEE Journal of Solid-State Circuits, vol. 58, no. 1, 2023, Saatavilla (Viitattu 12.3.2025): https://doi.org/10.1109/JSSC.2022.3219410.

[26] Chattopadhyay, A., and Z. Zilic. “GALDS: A Complete Framework for Designing Multiclock ASICs and SoCs.” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 13, no. 6, 2005, Saatavilla (Viitattu 20.3.2025): https://doi.org/10.1109/TVLSI.2005.848825

[27] Chattopadhyay, A., and Z. Zilic. “High Speed Asynchronous Structures for Inter-Clock Domain Communication.” 2002 9th IEEE International Conference on Electronics, Circuits and Systems, vol. 2, IEEE, 2002, Saatavilla (Viitattu 29.4.2025): https://doi.org/10.1109/ICECS.2002.1046213.

[28] Mahan, Vishwaraj, and Manish I. Patel. “Implementation of Retiming Technique for Clock Period Optimization by Brute-Force Approach.” 2023 IEEE Devices for Integrated Circuit (DevIC), IEEE, 2023, Saatavilla (Viitattu 29.4.2025): https://doi.org/10.1109/DevIC57758.2023.10134994.

[29] Aparicio, Hernan, and Pablo Ituero. “A 40nm Critical Path Monitor for the Detection of Setup and Hold Time Violations.” 2018 28th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS), IEEE, 2018, Saatavilla (Viitattu 29.4.2025): https://doi.org/10.1109/PATMOS.2018.8464155.

[30] LaMeres, Brock J. “Introduction to Logic Circuits & Logic Design with VHDL.” 1st ed. 2017., Springer International Publishing, 2017, Saatavilla (Viitattu 20.3.2025): https://doi.org/10.1007/978-3-319-34195-8.

[31] Xiaowen Wang, and William H. Robinson. “Asynchronous Data Sampling Within Clock-Gated Double Edge-Triggered Flip-Flops.” IEEE Transactions on Circuits and Systems. I, Regular Papers, vol. 60, no. 9, 2013, Saatavilla (Viitattu 25.3.2025 https://doi.org/10.1109/TCSI.2013.2244431.

[32] Crone, Allan, et al. “Functional Verification of Low Power Designs at RTL.” Integrated Circuit and System Design, vol. 4644, Springer Berlin / Heidelberg, 2007, Saatavilla (Viitattu 20.3.2025): https://doi.org/10.1007/978-3-540-74442-9_28.

[33] Henzler, S. (Stephan). Power Management of Digital Circuits in Deep Sub-Micron CMOS Technologies. Springer, 2007, Saatavilla (Viitattu 28.4.2025): https://doi.org/10.1007/1-4020-5081-X.

[34] KANNAIAN, Vasanthamani, and Visalakshi PALANISAMY. “Energy-Efficient Scheduling for Real-Time Tasks Using Dynamic Slack Reclamation.” TURKISH JOURNAL OF ELECTRICAL ENGINEERING & COMPUTER SCIENCES, vol. 27, no. 4, 2019, Saatavilla (Viitattu 13.3.2025): https://doi.org/10.3906/elk-1806-170.

[35] Wanner, L., et al. “Hardware Variability-Aware Duty Cycling for Embedded Sensors.” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 21, no. 6, 2013, Saatavilla (Viitattu 25.3.2025): https://doi.org/10.1109/TVLSI.2012.2203325.

[36] Su, Han, et al. “A 40-Nm Low-Power WiFi SoC with Clock Gating and Power Management Strategy.” International Journal of Electronics, vol. 110, no. 9, 2023, Saatavilla (Viitattu 23.3.2025): https://doi.org/10.1080/00207217.2022.2118843.

[37] Burd, T. D., et al. “A Dynamic Voltage Scaled Microprocessor System.” IEEE Journal of Solid-State Circuits, vol. 35, no. 11, 2000, Saatavilla (Viitattu 18.3.2025): https://doi.org/10.1109/4.881202.

[38] You, Daecheol, and Ki-Seok Chung. “Dynamic Power Management for Embedded Processors in System-on-Chip Designs.” Electronics Letters, vol. 50, no. 18, 2014, Saatavilla (Viitattu 13.3.2025): https://doi.org/10.1049/el.2014.1374.