Kisfogyasztású CMOS logikai áramkörök vizsgálata és tervezési eljárásuk kidolgozása

Ph.D értekezés

Varga László
okl. villamosmérnök

Témavezető:
Dr. Hosszú Gábor
egyetemi docens
a műszaki tudomány kandidátusa

Budapest, 2009.

Tartalomjegyzék

1. Bevezetés

Az igen nagy bonyolultságú integrált áramkörök (VLSI) teljesítményfelvétele az eszközméretek csökkenéséből adódó nagyobb alkatrészsűrűség és gyorsabb működési sebesség kihasználásával egyre jelentősebb tényezővé válik. A félvezetőlapka teljesítményfelvétele, és az ezzel együtt járó melegedése meghatározza mind az áramkör megbízhatóságát, mind az élettartamát [31]. Mindez arra kényszeríti a tervezőket, hogy csökkentsék az áramkörök teljesítményfelvételét, és így az ezzel együtt járó melegedést is, ami által elkerülhetővé válik a jó hővezető – azonban drága – integrált áramköri tokok és az egyéb, hűtést segítő eszközök használata [32]. A kis fogyasztású áramkörök iránti igény csak tovább fokozódik a hordozható eszközök terjedésével, ahol a telep élettartama döntő fontosságú. Mindez azt eredményezte, hogy manapság a kis fogyasztás már nem kevésbé fontos tervezési szempont, mint a teljesítőképesség vagy a félvezetőlapka területe [23].

A kis fogyasztásra való törekvésnek legjobban megtérülő módja a teljesítményfelvételi követelmények figyelembe vétele már a tervezéskor. A digitális rendszertervezés valamennyi szintjén létezik a teljesítményfelvétel csökkentésére irányuló módszer, a rendszerszinttől egészen az eszközszintig [24]-[30]. A kis fogyasztású áramkörök megvalósításának lehetőségei igen sokrétűek, ugyanakkor sajnos, az egyes módszereknek nagyon is jól látható korlátaik vannak. Ezért a lehetséges megoldásokat a kívánt cél elérése érdekében rendszerint keverten alkalmazzák.

A teljesítményfelvétel csökkentésének egy újszerű, az egyéb megoldásoktól alapvetően eltérő módja a hőtanból jól ismert adiabatikus elv alkalmazása a CMOS áramköri kapcsolásokban. Az adiabatikus, töltésvisszanyerő elven működő kapcsolások energiahatékonyságát egyrészt az ohmos veszteségek lecsökkentése, másrészt a betáplált energia nagy részének ismételt felhasználása eredményezi. Az adiabatikus töltési elv a következőket mondja ki [50][73]:

- a kombinációs logika energiafogyasztása tetszőlegesen alacsony lehet, ha az áramkört kellően lassan működtetjük,
- a tárolóelembe történő adatbeírás tetszőlegesen kis energiaveszteséggel megvalósítható,
- egy tárolóelemben tárolt adat tetszőlegesen kis energiaveszteség árán lemásolható,
- azonban a tárolt adat utolsó másolatának kitörléséhez elkerülhetetlenül szükséges egy véges, le nem csökkenthető energiamennyiség.

Természetesen az adiabatikus elv gyakorlati alkalmazásának, – mint megannyi más teljesítményfelvétel csökkentésére irányuló megoldásnak – alapvető gondja a működési sebesség, hiszen ennek rovására a fogyasztás ugyan jelentősen mérsékelhető, ám ekkor az alkalmazhatósági terület alapvetően leszűkül. Az adiabatikus elven működő CMOS áramköri kapcsolások teljesítményfelvétele a statikus CMOS kapcsolásokhoz képest nagyságrendekkel kedvezőbb lehet, azonban az elméleti értékeket jelentősen rontják a gyakorlati áramkörök parazitái, valamint a

kapcsolás járulékos elemei, mint például az adiabatikus tápláláshoz szükséges lejtős jelalakokat előállító tápegység veszteségei.

A technológia adta lehetőségek kihasználásával nemcsak az integrált áramkörök teljesítményfelvétele, hanem bonyolultsága is évről-évre exponenciálisan nő, míg a logikai és a regiszter-átviteli szintű tervezőeszközök képessége ezt időben csak lineárisan haladva követi [72]. Ezért az integrált-áramkör tervezés másik fontos kérdése a tervező teljesítménye, valamint az elkészült terv minősége. A tervező csak azon az elvonatkoztatási szinten tud dolgozni, amely szinten a tervezési tárgyak száma viszonylag kicsi. Például néhány logikai egyenlet még megérthető, de pár száz már nem. Ekkor át kell térni magasabb elvonatkoztatási szintre, ahol a tervezési tárgyak száma kevesebb, például néhány algoritmus. Alacsonyabb elvonatkoztatási szinten a terv csak akkor kezelhető, ha azt kisebb darabokra osztjuk, vagy olyan önműködő tervezőeszközöket használunk, amelyek képesek az adott összetettséget kezelni.

A korszerű, nagy bonyolultságú berendezés-orientált integrált áramkörök tervezéséhez ezért elengedhetetlen a magas szintű szintézis eszközök használata, mivel a magasabb szintű tervezőeszközök használatával a tervezés is magasabb elvonatkoztatási szintről indítható. A hardverleíró nyelvek és a hozzá kapcsolódó szimulációs és szintézis eszközök elterjedése és általános használata magával hozta a VLSI áramkörök kívánt működésének magas szintű megadását.

Ez a magas szintű megadás leírja azt a feladatot, amit a tervezendő áramkörnek végre kell hajtani. Ez a feladat azonban általában túl összetett ahhoz, hogy egyetlen eszközzel szintetizálható legyen. Nem létezik ugyanis olyan általános eszköz, amely az összes elképzelhető alkalmazási területre megfelelő. Ehelyett, az egyes magas szintű szintézis eszközök egy jól meghatározott alkalmazási területre készülnek. Egy szintézis eszköz használhatóságának csak egy alkalmazási területre való korlátozása lehetővé teszi a szintézis folyamat összetettségének kézbentartását, és így a szintézis hatékonyságának elősegítését. A szintézis hatékonyságának biztosítása alapvető fontosságú, hiszen a magas-szinten hozott tervezési döntések sokkal nagyobb hatással vannak a terv minőségére, mint az alacsonyabb szinten hozott hasonló döntések [93]. Egy teljes rendszer magas szintű szintéziséhez a rendszermegadást fel kell osztani egymástól eltérő jellegű feladatkörökre, amely feladatkörök a felosztás után az adott alkalmazási területre jellemző szintézis eszközökkel már egymástól függetlenül szintetizálhatók.

Munkám célja a kis fogyasztású CMOS logikai áramkörök kutatása, beleértve új, a szakirodalomból ismerteknél még kisebb energiaveszteségű kapcsolástechnikai megoldások kidolgozását, valamint – ehhez kapcsolódóan – a digitális áramkör-tervezési módszerek fejlesztését. A statikus CMOS kapcsolások teljesítményfelvétel-csökkentésének kézenfekvő módja a tápfeszültség lecsökkentése [26], ennek azonban ára a gyenge teljesítőképesség. Ráadásul a tápfeszültségnek a küszöbfeszültséghez való közelítésével a dinamikus kapcsolások működésképtelenekké válnak, továbbá az elérhető legkisebb energiaveszteséget behatárolja a küszöbfeszültség és a zajtartalék értéke.

A teljesítményfelvétel további csökkentésére, és ugyanakkor a teljesítőképesség megtartására mutatnak utat az adiabatikus töltésvisszanyerő elven működő áramkörök [34]-[71]. Azonban a jelenlegi adiabatikus kapcsolások vagy nem megfelelőek a CMOS megvalósításra [42], vagy túl összetettek és nagy szilíciumterületet igényelnek [43][53], vagy működtetésükhöz túl sok fázisjel szükséges [55], vagy csak alacsony frekvencián működőképesek [45], egyes esetekben pedig nem-adiabatikus veszteségük csökkenti az energiahatékonyságot [38].

A fentiek alapján az adiabatikus töltésvisszanyerő elven működő áramkörök széleskörű elfogadásához és gyakorlati elterjedéséhez elengedhetetlenül szükséges a kapcsolás egyszerűsítése, a hatékony működési frekvenciatartomány és az átbocsátás növelése, valamint a teljesítményfelvételi tulajdonságok további javítása.

Az értekezés második fejezetében a kutatásaimhoz kapcsolódó szakirodalmat tekintem át, kezdve a CMOS áramkörök teljesítményfelvételének forrásaival, bemutatva a teljesítményfelvétel csökkentésére irányuló megoldásokat, részletesen kitérve az adiabatikus töltési elv lehetséges gyakorlati alkalmazásaira. A fejezet további részében a rendszer- és magas szintű tervezés problémakörét taglalom, bemutatva a magas szintű logikai szintézis során megoldandó feladatokat, valamint a kutatásomhoz kapcsolódó szintézis eljárások fő jellemzőit.

Elért kutatási eredményeimet a harmadik fejezet tartalmazza. Az alfejezetek sorrendje megfelel a téziseim sorrendjének, azok a tézispontjaimban megfogalmazott állításokat támasztják alá. Ez alól kivétel az utolsó tématerülethez átvezetést nyújtó 3.3 alfejezetet, amelyhez tézis nem kapcsolódik.

A 3.1 alfejezet egy olyan újszerű, javított tulajdonságú adiabatikus töltésvisszanyerő kapcsolást ismertet, amely a jelenlegi adiabatikus kapcsolások gyengeségein igyekszik enyhíteni mind bonyolultság, mind energia-hatékonyság szempontjából. Az adiabatikus áramkörök teljesítményfelvételének további csökkentési lehetőségeivel a 3.2 alfejezet foglalkozik. A következő alfejezet bemutat egy hardverleíró nyelven alapuló, nyelvi átalakításokat alkalmazó magas szintű logikai szintézis eljárást. Erre építve, a fejezet utolsó alfejezete a digitális áramkörök magas szintű logikai szintézisében használatos ütemező eljárásnak az adiabatikus áramkörökre történő alkalmazását részletezi, amellyel a szakirodalom mindezidáig nem foglalkozott.

Dolgozatom a munkám összefoglalása és a téziseim ismertetése után köszönetnyilvánítással, a saját tudományos közleményeim és a független idézettségeim felsorolásával, majd végül a felhasznált irodalmi hivatkozások jegyzékével zárul.

2. Irodalmi áttekintés

Ebben a fejezetben röviden áttekintem a munkámhoz szorosan kapcsolódó szakirodalmat, amellyel célom egyrészt a tématerületbe betekintést nyújtani, másrészt pedig meghatározok olyan fogalmakat, amelyek a későbbiek szükségesek az általam elért eredmények leírásához. Itt mutatom be azokat az áramköröket is, amelyek tulajdonságaival elért eredményeimet összevetettem, valamint röviden ismertetem azokat a tervezési eljárásokat, amelyeket munkám során felhasználtam.

2.1. CMOS áramkörök teljesítményfelvételének forrásai

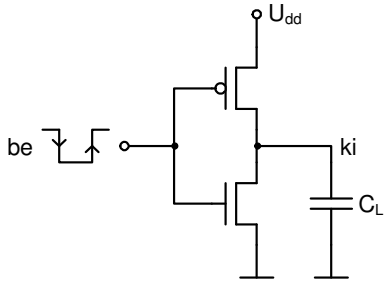
A statikus és az adiabatikus elven működő CMOS áramkörök teljesítményfelvétele eltérő okokra vezethető vissza, ezért e két áramkörtípus veszteségforrásait külön-külön alfejezetben mutatom be.

2.1.1. Statikus CMOS áramkörök teljesítményfelvételének forrásai

A statikus CMOS áramkörök teljesítményfelvétele a töltéspumpálásból, az egymásba-vezetési áramból, valamint a szivárgási és a küszöbalatti áramokból adódik [33]. A gyakorlatban a működésben lévő statikus CMOS áramkör teljes fogyasztását a töltéspumpálás a határozza meg, amely jól tervezett technológia esetén (azaz a többi tényező minimumra csökkentésével) a teljes fogyasztás több mint 90%-át adja [26].

A. Töltéspumpálás

A töltéspumpálás energiavesztesége a kapacitások feltöltésekor és kisütésekor fellépő Joule-veszteség. Ha a *2.1. ábrán* látható statikus CMOS inverter kimenetét a bemenete magas logikai szintre vezérli, akkor az U_{dd} tápfeszültséget „hirtelen” kapcsolják rá a kimeneti alacsony szintet tartó terhelőkapacitásra, aminek következtében a PMOS tranzisztoron jelentős feszültségesés lép fel.



2.1. ábra. Statikus CMOS inverter

Az átkapcsolás során a terhelőkapacitás $Q = C_L U_{dd}$ töltést, vagyis $E = 0.5QU_{dd} = 0.5C_L U_{dd}^2$ energiát tárol, míg a tápegységből felvett energia mennyiségét a (2.1) összefüggés adja.

$$E_{felvett} = C_L \cdot U_{dd}^2 \tag{2.1}$$

A tápegységből felvett energia felét a kimeneti C_L kapacitás tárolja, míg a másik fele hővé alakul a PMOS tranzisztor bekapcsolási ellenállásán. A későbbiekben, amikor a bemeneti logikai szint ismét megváltozik a terhelőkapacitásban tárolt energia az NMOS tranzisztoron hővé alakul, így a teljes folyamat teljesítményfelvétele a (2.1) összefüggés szerinti. A töltéspumpálásból adódó energiaveszteség független a tranzisztorok bekapcsolási ellenállásától és linearitásától, a küszöbfeszültségek értékétől, a töltési időtől és minden egyéb mástól [50].

B. Egymásba-vezetési áram

Az egymásba-vezetésből adódó energiaveszteség arányos az átkapcsolás alatt a tápegységből a földbe folyó töltések mennyiségével és a tápfeszültség értékével. Az egymásba-vezetési áramot a szakirodalomban általában elhanyagolják, tekintettel arra, hogy a jelek igen gyors fel- és lefutása következtében a keletkező áramtüske időtartama igen rövid, s így az erre jutó átlagos teljesítmény is igen alacsony a kapacitást töltő áramokhoz képest.

C. Szivárgási áram

A MOS tranzisztorok *source* és *drain* diffúziói a hordozó tömbbel parazita diódákat képeznek. Ezen diódák záróirányú árama statikus teljesítményfelvételt eredményez, amely teljesítmény az eszközméret csökkenéssel – egyrészt az alkalmazott nagyobb adalékoltság, valamint az eszközszám növelésből adódóan – növekszik. A szivárgási áram a *gate*-oxid-on is fellép, itt a veszteség az ugyancsak csökkenő *gate*-oxid vastagság függvényében exponenciálisan emelkedő értéket mutat.

D. Küszöbalatti áram

A lezárt tranzisztorokon statikus állapotban is folyó küszöbalatti áram a méretcsökkenéssel exponenciálisan növekszik. Ennek oka, hogy az alkalmazott egyre kisebb tápfeszültség következtében egyre kisebb küszöbfeszültségre van szükség. Továbbá a küszöbalatti áram a hőmérséklet függvényében is exponenciális növekedést mutat. Mindez előrevetíti, hogy a küszöbalatti áram a közeljövőben nemcsak készenléti állapotban lesz meghatározó, hanem a működésben lévő és ezért melegedő félvezetőlapka teljes fogyasztását is egyre nagyobb részben teszi majd ki [115].

2.1.2. Adiabatikus áramkörök teljesítményfelvételének forrásai

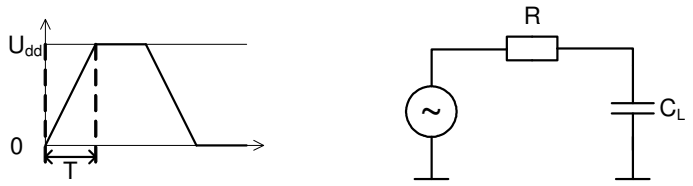
A hőtanból jól ismert adiabatikus energiaátvitel során az átvitel vesztesége az energiaáramlás sebességének korlátozásával tetszőlegesen kicsivé tehető. Ez a „közel” adiabatikus energiaátvitel a jelek lassú, lejtős fel és lefutásával a CMOS kapcsolásokban is megvalósítható.

Az adiabatikus elven működő CMOS áramkörök teljesítményfelvétele egyrészt az adiabatikus veszteségből, másrészt a nem-adiabatikus veszteségből, valamint a szivárgási és a küszöbalatti áramokból adódik [65]. Az áramköri kapcsolástól és a

működési frekvenciától függ, hogy az áramkör teljes fogyasztására nézve melyik a meghatározó, azonban nincs gyakorlati jelentősége annak a működési frekvenciatartománynak, amelyben a szivárgási és a küszöbalatti áramokból adódó teljesítményfelvétel van túlsúlyban.

A. Adiabatikus veszteség

Ha a *2.2. ábrán* látható C_L kapacitást az R ellenálláson keresztül – a szokásos egyenáramú táplálás helyett rezonáns tápegységet alkalmazva – az U_{dd} bemeneti feszültséget lineárisan emelve T ideig töltjük, úgy, hogy T sokkal nagyobb, mint a kapcsolás RC időállandója, akkor kapacitás feszültsége szorosan követi a bemeneti feszültséget, és a kapacitásba befolyó áramot a (2.2) képlettel közelítve közel állandónak tekinthetjük. Hasonlóképpen, ha a bemeneti feszültséget U_{dd} értékről T ideig lineárisan nullára csökkentve a kapacitást kisütjük, kapacitásból kifolyó áramot szintén közel állandónak tekinthetjük, amelynek abszolút értékét a (2.2) képlettel közelíthetjük. Feltételezve, hogy a rezonáns tápegység képes a feltöltött kapacitásban tárolt energia ismételt felhasználására, valamint a rezonáns tápegység belső veszteségeitől eltekintve a teljes (feltöltési és kisütési) folyamat energiaveszteségét a (2.3) egyenlettel közelíthetjük [66], amely abból adódik, hogy az R ellenállás P teljesítményt disszipál.



2.2. ábra. Adiabatikus töltés

$$I \approx C_L \frac{U_{dd}}{T} \tag{2.2}$$

$$E_{veszteség} = P \cdot 2T = I^2 R \cdot 2T = \left(C_L \frac{U_{dd}}{T} \right)^2 R \cdot 2T = 2 \frac{R \cdot C_L}{T} C_L \cdot U_{dd}^2 \tag{2.3}$$

A (2.3) egyenlet szerinti energiaveszteség az adiabatikus veszteség. Ez a CMOS áramkörökben mindig fellép, ha egy kinyitott tranzisztoron keresztül áram folyik és a töltésátvitel ideje lényegesen nagyobb, mint az adott elrendezés időállandója. Az utóbbi az a feltétel, amit az adiabatikus működéshez biztosítani kell.

Az adiabatikus energiaveszteség a (2.3) egyenlet alapján fordítottan arányos a töltési idővel, és ennek a végtelenhez való közelítésével az adiabatikus energiaveszteség értéke nullához tart. Tetszőlegesen kicsi adiabatikus teljesítményfelvétel elérhető a töltési idő megnövelésével, azonban ennek gátat szab többek között a működési sebesség lecsökkenése.

Ha a C_L terhelőkapacitást töltő áramútban N darab kinyitott tranzisztor sorosan helyezkedik el, akkor ezen kapcsolóelemek mindegyikét egy RC taggal modellezhetjük, ahol az R a nyitott tranzisztor bekapcsolási ellenállása, C pedig a

parazita kapacitása. Ha a töltési idő lényegesen nagyobb, mint a kapcsolás időállandója, akkor egy közbülső *k* tranzisztor áramának értéke a (2.4) képlettel közelíthető, míg a feltöltés során hővé alakult energia a (2.5) egyenlet szerinti.

$$I_k \approx (k \cdot C + C_L)\frac{U_{dd}}{T} \tag{2.4}$$

$$\begin{aligned} E_{\text{veszteség}} &= \sum_{k=1}^{N} I_k^2 R \cdot T = \sum_{k=1}^{N} (k \cdot C + C_L)^2 \frac{R \cdot U_{dd}^2}{T} = \\ &= \frac{N \cdot R \cdot U_{dd}^2}{T}\left(C_L^2 + (N+1)CC_L + \frac{(N+1)(2N+1)}{6}C^2\right) \end{aligned} \tag{2.5}$$

A (2.5) egyenletben felbontva a zárójeleket, az egyik tag értéke N^3-nal arányos, amiből az következik, hogy az adiabatikus veszteség alacsonyan tartásához az áramútban csak kis számú kapcsolóelem engedhető meg.

Ha az áramútban áteresztő kaput alkalmazunk és az U_{ds} *drain-source* feszültség kicsi (ami az adiabatikus működés feltétele), akkor a *p* és az *n* csatornás tranzisztor bekapcsolási ellenállását a (2.6) összefüggésekkel közelíthetjük. U_t a küszöbfeszültség értékét jelöli, C_p és C_n a *gate* kapacitás, K_p és K_n a *gate* feszültségtől és a *gate* kapacitástól független, az adott technológiára jellemző állandó. Ha egyenlő vezetési csatornahosszúság mellett a tranzisztorok W_p és W_n szélességét úgy állítjuk be, hogy $K_p/C_p = K_n/C_n$ akkor az áteresztő kapu bekapcsolási ellenállására a (2.7) egyenlet írható, így a feltöltés és kisütés energiaveszteségét a (2.8) összefüggés adja.

Az áteresztő kapu vezérlését statikus jelekkel végezve a vezérlés töltéspumpálásból adódó energiavesztesége $E_{\text{felvett}} = \alpha \cdot C_n \cdot U_{dd}^2$, ahol (egyéb parazita hatások elhanyagolásával) $\alpha = (C_n+C_p)/C_n$ közelítéssel élhetünk. Ha az áteresztő kapu tranzisztorainak W_p és W_n szélességét megnöveljük, akkor a kisebb bekapcsolási ellenállás a terhelőkapacitás töltése és kisütése során kisebb energiaveszteséget eredményez, azonban ekkor a vezérlésre fordított energiaveszteség a megnövekedett *gate* kapacitás miatt nagyobb lesz. A legkisebb teljes energiaveszteség értékét a (2.9) összefüggés adja, amit akkor kapunk, ha a vezérlés és a terhelőkapacitás töltésének illetve kisütésének energiavesztesége egyenlő [52].

$$R_p \approx \frac{K_p}{C_p(U_{ds} - U_t)} \qquad R_n \approx \frac{K_n}{C_n(U_{dd} - U_{ds} - U_t)} \tag{2.6}$$

$$R_{\text{áteresztő kapu}} = \frac{K_n}{C_n(U_{dd} - 2U_t)} \tag{2.7}$$

$$E_{\text{veszteség}} = \frac{2K_n \cdot C_L^2 \cdot U_{dd}^2}{T \cdot C_n(U_{dd} - 2U_t)} \tag{2.8}$$

$$E_{\text{legkisebb teljes veszteség}} = \sqrt{\frac{8\alpha \cdot K_n}{T \cdot (U_{dd} - 2U_t)}} C_L \cdot U_{dd}^2 \tag{2.9}$$

Megjegyzendő, hogy a gyakorlatban a terhelőkapacitás adiabatikus töltésére és kisütésére nem a lineárisan növekvő illetve csökkenő feszültség az ideális, mivel a logikai kapuk kimeneti – parazita – kapacitása a tranzisztorok *gate-drain* feszültségének függvénye, továbbá a gyakorlati rezonáns tápegységek a lineáris jelalakot szinuszos jellel közelítik [34]-[40]. A szinuszos jelalak hatását elméleti megfontolások alapján egy ξ alaktényezővel vehetjük figyelembe ($\xi = \pi^2/8$), gyakorlati eredmények azonban azt mutatják, hogy a lineáris helyett szinuszos jelalakkal való működtetésnek az adiabatikus energiaveszteségre nincs lényeges hatása [114].

B. Nem-adiabatikus veszteség

Az adiabatikus elven működő, de nem teljesen adiabatikus áramkörökben fellép nem-adiabatikus veszteség is, amikor egy bekapcsolt kapcsolóelem két végpontja közötti feszültségkülönbség nem nulla. A nem-adiabatikus veszteség független a működési frekvenciától, értékét a (2.10) összefüggés adja, ahol *U* a bekapcsolt kapcsolóelemen eső feszültséget jelöli.

$$E_{\text{veszteség}} = \frac{1}{2} C_L \cdot U^2 \tag{2.10}$$

C. Szivárgási és küszöbalatti áramok

A szivárgási és küszöbalatti áramokból adódó órajel-ciklusonkénti statikus teljesítményfelvétel arányos az órajel-ciklus hosszával, így nagyon alacsony működési frekvencia esetén ez a veszteség meghatározó lehet. A (2.4) és a (2.5) egyenleteket egy tranzisztorra felírva és kiegészítve az I_0 szivárgási és küszöbalatti árammal, a (2.11) és a (2.12) egyenletekhez jutunk. A teljesítményfelvétel szempontjából a kapacitások, valamint a tápfeszültség illetve a szivárgási és küszöbalatti áram függvényében egy legkedvezőbb *T* töltési idő kapható, amelynek értékét a (2.13) összefüggés adja.

$$I \approx (C + C_L)\frac{U_{dd}}{T} + I_0 \tag{2.11}$$

$$E_{\text{veszteség}} = I^2 R \cdot T = \frac{R \cdot (C + C_L)}{T}(C + C_L) \cdot U_{dd}^2 + I_0^2 R \cdot T \tag{2.12}$$

$$T_{\text{legkedvezőbb}} = \frac{(C + C_L)U_{dd}^2}{I_0} \tag{2.13}$$

A küszöbfeszültség értéke meghatározza mind a küszöbalatti áram, mind a bekapcsolási ellenállás értékét. Bár az alacsonyabb küszöbfeszültség mérsékli a bekapcsolási ellenállást, ugyanakkor nagyobb küszöbalatti áramot eredményez, ami akár összemérhető is lehet a bekapcsolási áram értékével. Mindez az eszközméretek csökkenésével egyre inkább előtérbe kerül. Gyakran a gyártási technológia – így a küszöbfeszültség értéke is – adott, amihez a legkedvezőbb töltési/kisütési időt a (2.13) összefüggés szerint lehet meghatározni.

2.2. A teljesítményfelvétel csökkentésének lehetőségei

A teljesítményfelvétel csökkentésére a digitális áramkör tervezés valamennyi szintjén kínálkozik lehetőség, azonban ezek megtalálása és kiaknázása ma még nem teljesen feltárt terület. Az eddig kidolgozott módszerek alkalmazási köre vagy nagyon szűk, vagy egy megoldás önmagában kevésbé hatékony, vagy alkalmazásával egyéb paraméterekben (szilícium terület, működési sebesség) kell súlyos árat fizetni. Kielégítő eredmény sokszor csak több módszer együttes alkalmazásával érhető el.

A következőkben – nem mindenre kiterjedően – röviden áttekintjük a digitális áramkör tervezés egyes szintjein a teljesítményfelvétel csökkentésére irányuló jelentősebb megoldásokat, részletesen kitérve azonban az adiabatikus elven működő kapcsolásokra, mivel ezek képezik munkám jelentős részét.

2.2.1. A teljesítményfelvétel csökkentésének tranzisztor-szintű lehetőségei

A teljesítményfelvétel csökkentésének főbb tranzisztor-szintű lehetőségei többek között a méretcsökkentés, a küszöbfeszültség változtatás, a tranzisztorméret változtatás, a tápfeszültség csökkentés és a szigetelőanyagú hordozó alkalmazása, amelyeket részletesebben is áttekintünk.

A. Méretcsökkentés

Az eszközméretek csökkentése értelemszerűen csökkenti a parazita kapacitások értékét és ezzel a töltőáramokat is. Ennek ellenére a korábban várt irányzat, nevezetesen, hogy a méretcsökkentés hatására a félvezetőlapkák fogyasztása csökkenni fog, nem következett be, sőt a méretcsökkentésből adódó nagyobb alkatrészsűrűség és gyorsabb működési sebesség kihasználása folytán a teljesítményfelvétel – és így a melegedés – még tovább fokozódik. Ráadásul 100 nm csíkszélesség alatt a parazita hatások már nem csökkennek a megszokott mértékben. Mindezek mellett a félvezetőlapkák méretének növekedése is rendkívül lelassult, szinte megállt, ami szintén rontja a hűtési viszonyok javulási esélyeit. A szakirodalom ezért magából a méretcsökkentésből adódó fogyasztáscsökkenéssel egyáltalán nem számol, a javítási lehetőségeket más utakon keresi.

B. Küszöbfeszültség változtatás

A küszöbfeszültség növelésének előnye, hogy ezzel párhuzamosan lecsökken a küszöbalatti áram, amely áram statikus állapotban is fogyasztást eredményez. Azonban a bekapcsolási ellenállás megnövekedése, és így a kapacitások lassabb töltődése és kisütése következtében az áramkör lelassul.

Statikus kapcsolásokban járható út az áramkör egyes jelutaiban lévő „időzítési hézag” eltüntetése. Ez azt jelenti, hogy azokban a jelutakban, ahol a késleltetési időkre vonatkozóan tartalék áll rendelkezésre (nem-kritikus jelutak), ott nagyobb küszöbfeszültségű, és így nagyobb késleltetésű kapuk alkalmazhatók anélkül, hogy a

teljes áramkör teljesítőképessége megváltozna [11]. Az egy logikai kapuhoz összefutó jelutak késleltetésének kiegyenlítése ezenkívül csökkentheti a hamis jelváltozások (glitch) számát, és így az ebből eredő fogyasztást is [29][92]. Ez a megoldás ugyanakkor megnöveli a kritikus jelutak számát, ami a gyártási technológia szórásából adódóan a kihozatal rovására mehet.

Adiabatikus áramkörök esetében az egyes kapcsolásoknál eltérő, és a működési frekvenciától függ, hogy a küszöbfeszültség csökkentésével nő-e vagy csökken az áramkör teljes vesztesége. Jellemző azonban, hogy a kisebb küszöbfeszültség kisebb nem-adiabatikus veszteséget eredményez [57].

C. Tranzisztorméret változtatás

A tranzisztorok mérete befolyásolja a logikai kapu a késleltetését és a teljesítményfelvételét.

Statikus áramkörökben a nagyobb tranzisztorméret csökkenti a késleltetést, ugyanakkor nagyobb energiaveszteséget eredményez. Megnöveli továbbá a meghajtó logikai kapu késleltetését, mivel a nagyobb tranzisztorok nagyobb kapacitív terhelést jelentenek. A tranzisztorméret csökkentésével járó késleltetés növekedés szintén felhasználható a nem-kritikus jelutak „időzítési hézagainak" eltüntetésére, és így változatlan teljesítőképesség mellett a fogyasztás csökkentésére. A minél jobb eredmény elérése érdekében gyakran a tranzisztorméret változtatást és a küszöbfeszültség változtatást együtt alkalmazzák [76][109].

Az adiabatikus kapcsolásokban a bekapcsolási ellenállás csökkenthető a tranzisztorméret növelésével, amivel az adiabatikus veszteség csökkenthető, ugyanakkor a nagyobb *gate* kapacitás miatt a tranzisztor vezérléséhez szükséges energia mennyisége is nagyobb. A tápfeszültség, a terhelőkapacitás és a töltési idő függvényében a tranzisztorok méretére (vezetési csatorna szélességére) egy elméleti optimum adható (lásd (2.6) - (2.9) összefüggések) [52], amely azonban csak egy sajátos elrendezésre érvényes, ezenkívül nem veszi figyelembe a kapcsolás elszórtan elhelyezkedő parazitáit.

D. Tápfeszültség csökkentés

A tápfeszültség csökkentés módszere lényegében hasonló a küszöbfeszültség változtatáshoz, de itt a fogyasztás csökkentését – és ezzel együtt a kapuk lassítását – további tápfeszültség-szintek alkalmazásával érik el [12]. Ez önmagában kézenfekvő megoldás a teljesítményfelvétel csökkentésére, azonban alkalmazásának határt szab a zajvédettség, továbbá hatékonyságát ronthatja a megnövekedett kapcsolási idő alatt hosszabb ideig folyó egymásba vezetési áram vagy a velejáró küszöbfeszültség-csökkentésből adódó küszöbalatti áram növekedés.

A tápfeszültség lecsökkentéséből adódó teljesítőképesség-romlás ellensúlyozása történhet párhuzamosítással és/vagy pipeline működés alkalmazásával, ahol az egymás után következő adatokat időben átlapolva dolgozzák fel [91]. Ez a megoldás azonban jelentős szilíciumterület többlettel járhat.

E. Szigetelőanyagú hordozó alkalmazása

A parazitakapacitások lényeges lecsökkenését eredményezi a zafír-alapú (SOI) hordozó alkalmazása, amelyet jelenleg nagyrészt csak professzionális (katonai, űrkutatási) célokra használnak. Az előrejelzések szerint azonban egyre szélesebb körben jelenik majd meg a közfogyasztású elektronikában is [113].

2.2.2. A teljesítményfelvétel csökkentésének kapcsolási szintű lehetőségei

A teljesítményfelvétel csökkentésére kapcsolási szinten számos lehetőség kínálkozik, többek között lehetséges a fogyasztás mérséklése különleges kialakítású dinamikus logikával [13], töltés-újrafelhasználáson alapuló differenciális logikával [14], áteresztő-tranzisztorokkal felépített kapcsolással [15], órajellel kapuzott, differenciális kaszkád áramkapcsolós logikával [16], és az adiabatikus töltési elv alkalmazásával [34]-[71].

Mivel munkámhoz szorosan csak a 2.1.2. alfejezetben bemutatott adiabatikus töltési elv alkalmazásával működő áramkörök kapcsolódnak, ezért a továbbiakban csak ezekkel foglalkozom. Az adiabatikus töltési elv gyakorlati alkalmazása a teljesítményfelvétel nagymértékű lecsökkentésének igen hatékony módja, amely a fogyasztáscsökkenést a működési sebesség rovására éri el. Bár a működési sebesség lecsökkentése az alkalmazások döntő többségében megengedhetetlen, az adiabatikus áramköröknek mégis van létjogosultsága, mivel a felépítésükből adódó pipeline működtetési lehetőség kihasználásával számos alkalmazás által igényelt átbocsátás az egyébként alacsony működési sebesség ellenére is tartható.

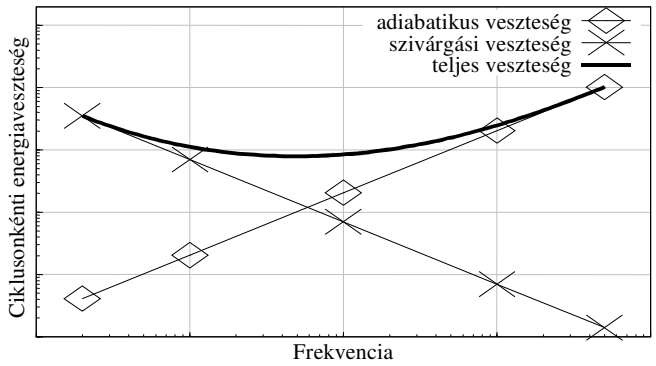
Az adiabatikus töltési elv lehetővé teszi, hogy a befektetett energiának csak egy kis része emésztődjön fel, míg nagy része a tápegységbe visszavezetve a következő órajel-ciklusban ismételten felhasználható. Az adiabatikus áramkörök táplálásához használt lejtős fel- és lefutású fázisjelek egyben a kapcsolás órajelei is. A fázisjelek alakjától eltekintve a megoldás hasonlít a régóta ismert tápfeszültség nélküli, úgynevezett fázis-táplálású dinamikus MOS áramkörökhöz [33]. A lejtős fázisjelekkel való tápláláshoz és a befektetett energia ismételt felhasználásához – a statikus CMOS áramköröknél megszokott egyenáramú táplálás helyett – úgynevezett rezonáns tápegységre van szükség, ami a gyakorlatban egy *RLC* kör, ahol a kapacitást maga az áramkör adja. A gyakorlati adiabatikus kapcsolásokat a komplementer felépítés jellemzi, vagyis mind a ponált, mind a negált logikát megvalósítják, amely azért szükséges, hogy az adiabatikus tápegység kapacitív terhelése, és így a rezgés frekvenciája független legyen az áramkör által feldolgozott adatoktól.

Az adiabatikus elven működő CMOS áramköri kapcsolásokban az adiabatikus veszteség valamint a szivárgási és a küszöbalatti áramokból adódó veszteség elkerülhetetlen, míg a nem-adiabatikus veszteség kiküszöbölhető. Energiaveszteség-típusaik alapján így az adiabatikus töltési elvet használó áramköri kapcsolásokat két csoportra oszthatjuk: teljesen adiabatikusan és kvázi-adiabatikusan működő áramkörökre.

A. Teljesen adiabatikus működésű áramkörök

A teljesen adiabatikus működésű áramkörök legfőbb tulajdonsága, hogy nincs nem-adiabatikus veszteségük. Ez elérhető egyrészt úgynevezett *töltésvisszarántó* elrendezéssel, ahol a bemenetek stabilan tartásával a kimeneti kapacitás a feltöltéshez is használt áramúton kisüthető. A logikai fokozatok száma így azonban jelentősen behatárolja a működési sebességet, mivel minden fokozatnak meg kell várnia az őt követő összes fokozat kiértékelését és visszahúzását. *Reverzibilis logikával* [52]-[54] ez elkerülhető, ahol a teljesen adiabatikus működést a következő logikai fokozat inverz logkai függvényével megvalósított kisütő áramút biztosítja.

A működési frekvenciatartományt a meghatározó veszteség szempontjából két részre oszthatjuk, amelyeket a *2.3. ábra* mutat [54]. Alacsony frekvenciákon a szivárgási és a küszöbalatti áramokból adódó veszteség az uralkodó, míg magasabb frekvenciákon a teljes veszteséget az adiabatikus veszteség határozza meg.

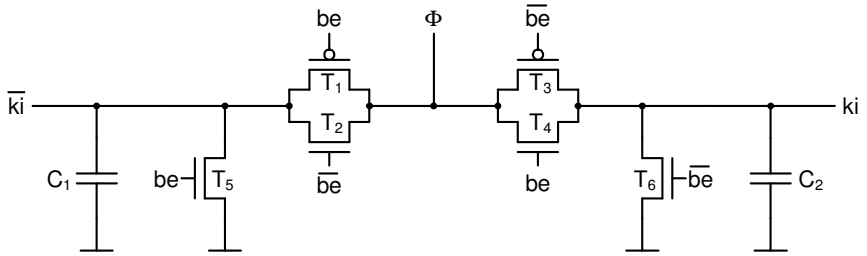


2.3. ábra. Teljesen adiabatikus működésű áramkörök energiaveszteség görbéi

Amint az ábrán is látszik, a teljes veszteségnek van egy legkisebb értéke, amikor a szivárgási áramokból adódó veszteség és az adiabatikus veszteség értéke egyenlő. Gyakorlati mérések alapján a legkisebb energiaveszteséghez tartozó frekvenciaérték reverzibilis logikát alkalmazó áramkörök esetén 50-200 kHz környékén van, az adott kapcsolástól függően. Ennél alacsonyabb frekvencián való működtetés gyakorlatilag értelmetlen, túl alacsony frekvencián lehetetlen is, mivel a hosszú töltési idő alatt a szivárgási áramokból adódó töltésfelhalmozás miatt a logikai szintek sérülnek. Magasabb frekvenciákon viszont a reverzibilis logikát alkalmazó áramkörök energiahatékonysága erősen romlik, mivel az áramutakban található kapcsolóelemek nagy száma miatt az adiabatikus veszteségük jelentőssé válik. A teljes reverzibilitásnak ugyanis ára van, ami a kapcsolás összetettségében és nagy szilíciumterület-igényében jelentkezik. Például egy reverzibilis logikát alkalmazó, teljesen adiabatikus működésű összeadó 20-szor annyi tranzisztort, és 32-szer annyi szilíciumterületet igényel, mint egy hagyományos összeadó [52]. Továbbá az áramutakban található kapcsolóelemek darabszáma egyben a működési frekvenciát is behatárolja, mivel a kapcsolás időállandója jelentősen megnő. Mindezeken túl, a szivárgási veszteségek az eszközméretek csökkenésével egyre jelentősebbé válnak, így a nagy bonyolultságú, sok elemet tartalmazó áramkörök egyre inkább jelentőségüket vesztik. Ezért a továbbiakban csak az egyszerű felépítésű, kevés számú elemet tartalmazó teljesen adiabatikus működésű kapcsolásokat tekintem át.

A.1. Adiabatikus buszmeghajtó áramkör

Egy adiabatikus elven működő buszmeghajtó áramkört [51] ismertet a *2.4. ábra.* A működés szemléltetésére tételezzük fel, hogy a Φ adiabatikus (azaz lejtősen fel- és lefutó) órajel kezdetben alacsony szintű, valamint a bemenetek az órajel felfutása, tartása és lefutása alatt végig változatlanok, és a *be* bemenet magas, így a $\overline{be}$ alacsony szintű. Ekkor a T_3 és a T_4 tranzisztorok nyitottak és a lassan felfutó Φ órajel adiabatikusan feltölti a C_2 kapacitást. A C_1 kapacitás feltöltetlen marad, és azért, hogy a szivárgási áramokból adódóan se változhasson a töltése, a T_5 tranzisztor földre kapcsolja. Az órajel tartása alatt a kimeneteken érvényes adat van, míg az órajel lefutásával együtt C_2 töltése – a feltöltéshez is használt útvonalon keresztül – adiabatikusan kisül. A kapcsolás hatékonyan használható nagy kapacitív terhelést jelentő buszok meghajtására, mivel ott a sebességi követelmények általában mérsékeltek.



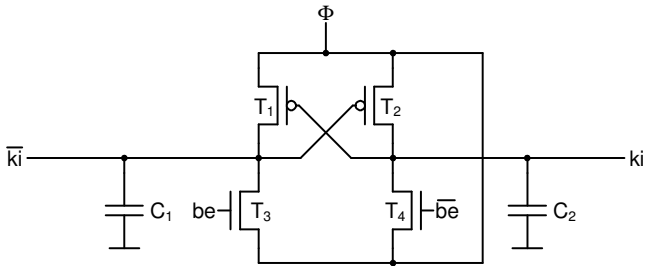
2.4. ábra. Adiabatikus buszmeghajtó áramkör

A.2. Továbbadótranzisztoros adiabatikus logika (PAL)

Az továbbadótranzisztoros adiabatikus logika [45], (*Pass-transistor Adiabatic Logic*, röviden *PAL*) felépítése a *2.5. ábrán* látható, amely nagyon hasonlít a kvázi-adiabatikus működésű áramkörökre jellemző keresztbekötött PMOS tranzisztorpárt alkalmazó kapcsolásra, mint amilyen például a későbbiekben ismertetendő *2N-2P* kapcsolás. A *PAL* kapcsolásban a T_3 és a T_4 tranzisztorok *source*-a a *2N-2P* kapcsolásbeli földpont helyett a fázisjelhez kapcsolódik, amivel a nem-adiabatikus veszteség – a működési sebesség rovására – teljesen kiküszöbölhető.

A *PAL* logikai kapuk pipeline működtethetők két, egymástól 180°-al eltolt fázisjel segítségével, ahol az egymás utáni fokozatokban elhelyezkedő kapuk felváltva az egyik, illetve a másik fázisjelhez kapcsolódnak. A működés szemléltetésére tételezzük fel, hogy a Φ fázisjel kezdetben alacsony szintű, valamint *be* bemenet magas, így a $\overline{be}$ alacsony szintű. Ekkor a T_3 tranzisztor nyitott, ezáltal a C_1 kapacitás Φ fázisjelhez kapcsolódik, a T_4 tranzisztor pedig zárt, így a C_2 kapacitás lebeg. A felfutó Φ fázisjel a C_1 kapacitást adiabatikusan tölti, a $\overline{ki}$ kimenet feszültsége követi a fázisjel értékét. A C_2 kapacitás feltöltetlen marad, mivel mind a T_2, mind a T_4 tranzisztor zárt, azonban lebeg, és mivel a fázisjel felfutásának kezdetén egy rövid ideig a T_2 tranzisztor is vezet, valamint a kapacitív töltésbecsatolás miatt feszültsége a küszöbfeszültség fölé nő.

A kimeneteken érvényes logikai szint van, amint a Φ fázisjel eléri a legnagyobb értékét, azonban az alacsony logikai szintű kimenet feszültsége nem nulla. A Φ fázisjel lefutásakor mind a C_1, mind a C_2 kapacitás adiabatikusan teljesen kisül a T_3, illetve a T_4 tranzisztoron keresztül, amit az előző fokozat küszöbfeszültségnél magasabb alacsony logikai szintje tesz lehetővé.

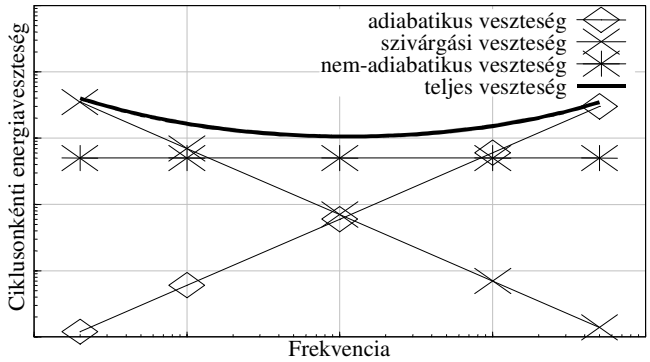


2.5. ábra. A *PAL* inverter

B. Kvázi-adiabatikus működésű áramkörök

A teljesen adiabatikus működésű áramkörök esetében a teljesítményfelvételt nagyon kis mértékűre csökkentették, azonban ezek az áramkörök nem megfelelőek akkor, ha a kis fogyasztás mellett a nagy működési sebesség is követelmény. A teljesítményfelvétel és a működési sebesség közötti egyezséget a kvázi-adiabatikus működésű áramkörök alkalmazása jelentheti.

A kvázi-adiabatikus működésű áramkörök legfőbb jellemzője, hogy a küszöbalatti és a szivárgási áramokból adódó valamint az adiabatikus veszteségük mellett nem-adiabatikus veszteségük is van. Eszerint a működési frekvenciatartományt a meghatározó veszteség szempontjából három részre oszthatjuk, amelyeket a *2.6. ábra* tüntet fel [54]. Alacsony frekvenciákon a küszöbalatti és a szivárgási áramokból adódó veszteség az uralkodó. Növelve a működési frekvenciát a teljes veszteség először lineárisan csökken, majd elérve a középső tartományt, ahol a nem-adiabatikus veszteség a döntő, a teljes veszteség a frekvenciától függetlenül közel állandó lesz. Tovább növelve működési frekvenciát, a teljes veszteséget az adiabatikus veszteség határozza meg, amely a működési frekvenciával lineárisan emelkedik.



2.6. ábra. Kvázi-adiabatikus működésű áramkörök energiaveszteség görbéi

Amint az ábrán is látszik, a teljes veszteségnek a legkisebb értéke abban a tartományban van, ahol a nem-adiabatikus veszteség a döntő. Mivel a kvázi-

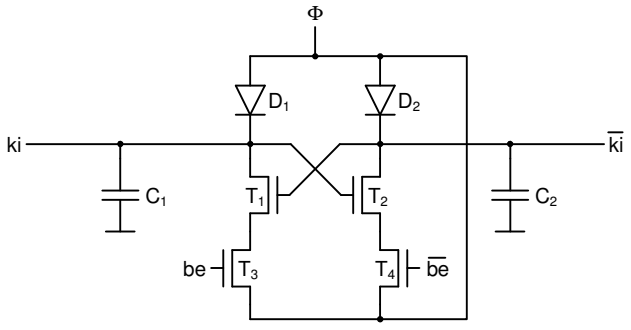
adiabatikus kapcsolásokban a nem-adiabatikus veszteség gyakorlatilag mindig nagyobb a teljesen adiabatikus kapcsolások legkisebb teljes veszteségénél, ezért kvázi-adiabatikus működésű áramkörökkel nem lehet olyan kis teljesítményfelvételt elérni, mint a teljesen adiabatikus működésű áramkörökkel. Azonban a kvázi-adiabatikus kapcsolások adiabatikus vesztesége ugyanazon működési frekvencián rendszerint jóval kisebb, mint a teljesen adiabatikus kapcsolásoké, ezért rendszerint energiahatékonyságuk is jóval kedvezőbb abban a működési tartományban, ahol az adiabatikus veszteség a meghatározó.

További előnye a kvázi-adiabatikus kapcsolásoknak az egyszerűségük, és így a kis szilíciumterület-igényük, valamint az, hogy teljesítményfelvételük kevésbé függ a terhelőkapacitás értékétől, és az áramutakban található kapcsolóelemek kis száma nagyobb működési frekvenciát is megenged. Természetesen a kvázi-adiabatikus működésű áramköröket is értelmetlen olyan frekvenciatartományban működtetni, ahol a küszöbalatti és a szivárgási áramokból adódó veszteség a döntő, illetve lehetetlen is, ha a túlságosan hosszú töltési idő alatt a szivárgási áramokból adódó töltésfelhalmozás miatt a logikai szintek sérülnek. Mindezek értelmében a kvázi-adiabatikus működésű áramkörök felhasználási köre sokkal tágabb, mint a teljesen adiabatikus működésűeké.

A kvázi-adiabatikus működésű áramkörök egy része [47][60] a *gate-csatorna* kapacitáson keresztüli feszültség utánhúzás (*bootstrap*) [21] elvét alkalmazza, hogy *n* csatornás tranzisztorral a logikai magas szintet átvigye. Az ilyen logikák nem, vagy csak csekély mértékű jelszint-helyreállító képességgel rendelkeznek, ami behatárolja a lehetséges működési frekvenciatartományt. Nagyobb frekvenciákon ugyanis a töltőáramok is nagyobbak, és az utánhúzott tranzisztorok bekapcsolási ellenállásain eső nagyobb feszültség csökkent kimeneti magas jelszintet eredményez, ami rontja a következő fokozatban elhelyezkedő logikai kapukban az utánhúzás hatékonyságát. Ez a folyamat önmagát gerjeszti, és így a magas logikai szint néhány fokozaton belül elvész, ami hibás logikai működéshez vezet. A következőkben ezért csak a gyors működésre alkalmas kvázi-adiabatikus kapcsolásokat ismertetem.

B.1. Dióda kapcsolt logika (2N-2N2D logika)

Valamennyi dióda kapcsolt logika működése az adiabatikus töltés *dinamikus* változatán alapul, ahol a kimeneti magas logikai értéket a feltöltött terhelőkapacitás tárolja. A terhelőkapacitás feltöltéséhez (előtöltés) a fázisjel átvezetését diódával (diódaként kapcsolt tranzisztorral) valósítják meg. A számos dióda kapcsolt elrendezés közül az egyik a *2N-2N2D* logika [56], amelynek invertere a *2.7. ábrán* látható.



2.7. ábra. A *2N-2N2D* inverter

A *2N-2N2D* logikai kapuk pipeline működtethetők két, egymástól 180°-al eltolt fázisjel segítségével, ahol az egymás utáni fokozatokban elhelyezkedő kapuk felváltva hol az egyik, hol a másik fázisjelhez kapcsolódnak. A logikai kapuk bemeneteinek az előtöltés alatt nem, csak fázisjel visszafutása alatt (kisütés, kiértékelés) kell érvényesnek lenniük.

A kapcsolás hátránya egyrészt – a diódán eső feszültség miatt – a csökkent kimeneti logikai magas jelszint, másrészt a zajérzékenység, mivel előtöltés után kimenetek lebegnek, harmadrészt az, hogy előtöltés alatt a nyitott diódán eső feszültség nem-adiabatikus veszteséget eredményez, ami alsó korlátot ad a kapcsolás energiaveszteségére. Ezen nem-adiabatikus veszteség értéke a (2.14) összefüggéssel közelíthető, ahol C_L a terhelőkapacitás értéke, U_f a dióda nyitófeszültsége, U_{dd} pedig a fázisjel legmagasabb értéke.

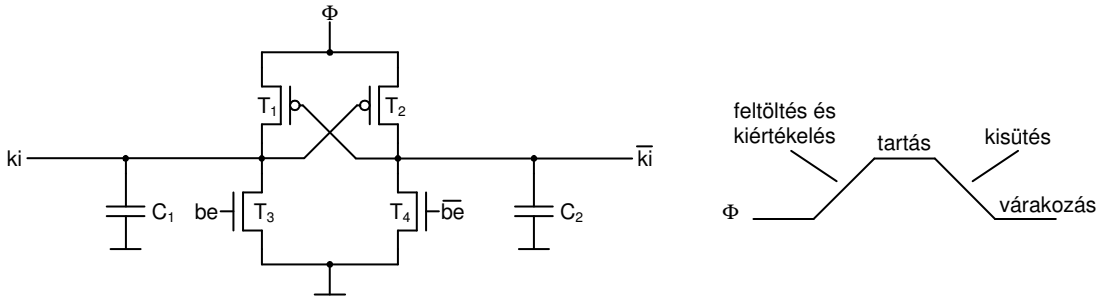
$$E_{\text{dióda}} \approx q \cdot U_f \approx C_L \cdot U_{dd} \cdot U_f \tag{2.14}$$

A nem-adiabatikus veszteség mellett természetesen még az adiabatikus valamint a küszöbalatti és a szivárgási áramokból adódó veszteség is fellép, azonban ezeket elhanyagolva a dióda kapcsolt logikák energiavesztesége a statikus CMOS kapcsoláshoz viszonyítva a (2.15) arány szerinti, ahol C_s a statikus, C_d pedig a dióda kapcsolt adiabatikus kapu terhelő kapacitása. Ez $U_{dd} = 5$ V és $U_f = 0.5$ V esetén a mért 1–2 kapacitás-arányokkal 10–20-szoros csökkenést jelent, figyelmen kívül hagyva még az adiabatikus fázisjelek előállításából adódó veszteségeket.

$$\frac{E_{\text{statikus}}}{E_{\text{dióda}}} \approx \frac{C_s U_{dd}^2}{C_d U_{dd} U_f} \approx \left(\frac{U_{dd}}{U_f}\right)\left(\frac{C_s}{C_d}\right) \tag{2.15}$$

B.2. Hatékony töltésvisszanyerő logika (2N-2P logika)

A hatékony töltésvisszanyerő logikai kapcsolás [57], amely felépítéséből adódóan a *2N-2P* néven is ismert, a *2.8. ábrán* látható. Az ábrán látható a működéséhez szükséges fázisjel elvi alakja is, amely a feltöltés és egyben kiértékelés, a tartás, a kisütés és a várakozás periódusokból áll.



2.8. ábra. A *2N-2P* inverter és négyperiódusú fázisjele

A működés szemléltetésére tételezzük fel, hogy a Φ adiabatikus fázisjel kezdetben alacsony szintű, valamint a bemenetek az órajel felfutása alatt változatlanok, és a *be* bemenet magas, így a $\overline{be}$ alacsony logikai szintű. Ekkor a

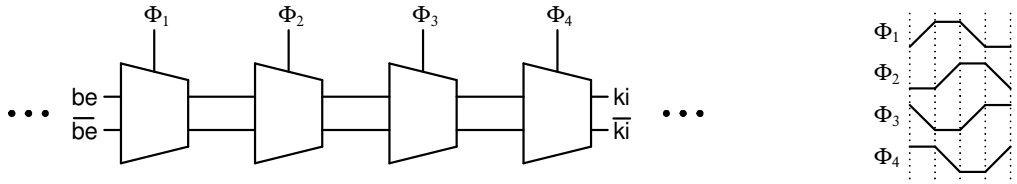
T_3 tranzisztor nyitott, így a C_1 kapacitás a földre kapcsolódik, a T_4 tranzisztor pedig zárt, így a C_2 kapacitás lebeg. Ha a felfutó Φ fázisjel értéke eléri a PMOS tranzisztor küszöbfeszültségének abszolút értékét, az $|U_{t,p}|$-t akkor a T_2 tranzisztor kinyit, és innentől kezdve a C_2 kapacitás adiabatikusan töltődik, így a $\overline{ki}$ kimenet feszültsége követi a fázisjel értékét. A T_2 tranzisztor kinyitásának pillanatában egy rövid időre a T_1 tranzisztor is kinyit, azonban a C_2 kapacitás feszültségének emelkedésével azonnal le is zár, így a C_1 kapacitás feltöltetlen marad, mivel a T_3 tranzisztor még mindig nyitott.

Amint a Φ fázisjel eléri a legnagyobb értékét, a kimeneteken érvényes logikai szint van (tartási periódus). A kisütési periódusban a C_2 kapacitás a feltöltéshez is használt útvonalon keresztül adiabatikusan kisül, egészen addig, amíg a lefutó Φ fázisjel értéke $|U_{t,p}|$ alá nem csökken, és a T_2 tranzisztor le nem zár. Ekkor a C_2 kapacitásban $|U_{t,p}|$ feszültség marad, ami nem gond, ha a következő fázisjel-ciklusban ugyancsak C_2 lesz feltöltve. Ha azonban a Φ fázisjel következő felfutásakor a kapu logikai állapota megváltozik, akkor a C_2 kapacitás töltése a T_4 tranzisztoron keresztül nem-adiabatikusan sül ki, a (2.10) összefüggés szerinti energiaveszteséget okozva. Ugyancsak nem-adiabatikus veszteséget okoz, hogy feltöltéskor a PMOS tranzisztor nem nyit ki azonnal, amelynek értékét úgyszintén a (2.10) összefüggés adja.

Ismételten elhanyagolva az adiabatikus, valamint a küszöbalatti és a szivárgási áramokból adódó veszteségeket és legrosszabb esetet feltételezve – amikor is a kapu logikai állapota minden fázisjel-ciklusban megváltozik – a *2N-2P* kapcsolás energiamérlegét a dióda kapcsolt logikához viszonyítva a (2.16) arány adja. Eszerint a *2N-2P* kapcsolással $U_{dd} = 5$ V, $U_{t,p} = -0.5$ V és $U_f = 0.5$ V esetén egyenlő kapacitásértékekkel számolva további tízszeres fogyasztáscsökkenés érhető el.

$$\frac{E_{\text{dióda}}}{E_{2N-2P}} \approx \frac{C \cdot U_{dd} \cdot U_f}{C \cdot U_{t,p}^2} \tag{2.16}$$

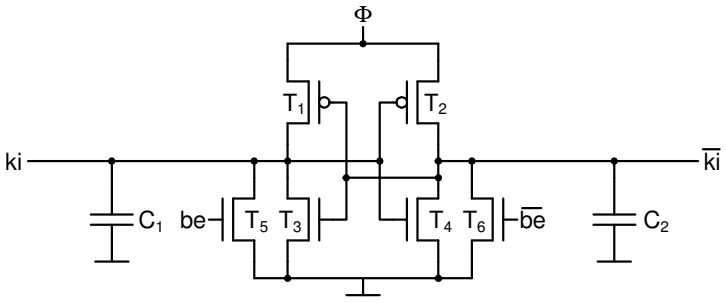
A *2N-2P* logikai kapuk pipeline működtetéséhez négy, egymástól 90°-al eltolt fázisjelre van szükség. Az egymás utáni fokozatokban elhelyezkedő logikai kapukat működtető fázisjelek sorrendjét, és ezek egymáshoz való viszonyát mutatja a *2.9. ábra*. Valamennyi logikai kapu bemenetén érvényes logikai szint van az adott kapuhoz bekötött fázisjel feltöltés és kiértékelés periódusa alatt, míg a logikai kapu kimenete egy periódussal eltolva, a tartás alatt lesz érvényes. Ugyanebben a periódusban a bemeneti logikai szintek érvénytelenné válnak, mivel az előző fokozatban lévő kapuk ekkor kisütés periódusban vannak, ezáltal a fel nem töltött terhelőkapacitás lebeg. A kisütés periódusban töltésvisszanyerés történik, míg a várakozás alatt az előző fokozatban lévő logikai kapuk az új bemeneti értékeket állítják elő, amelyek ekkor már a feltöltő periódusban vannak.



2.9. ábra. A *2N-2P* logikai kapuk pipeline kapcsolása és fázisjelei

B.3. 2N-2N2P logika

A *2N-2N2P* logika [58] a *2N-2P* logikának azt a hátrányát hivatott kiküszöbölni, hogy a fázisjel tartás periódusban a fel nem töltött kimenet lebeg. A *2N-2N2P* kapcsolás invertere a *2.10. ábrán* látható. Ez a kapcsolás hasonló jellemzőkkel rendelkezik, mint a *2N-2P* kapcsolás.

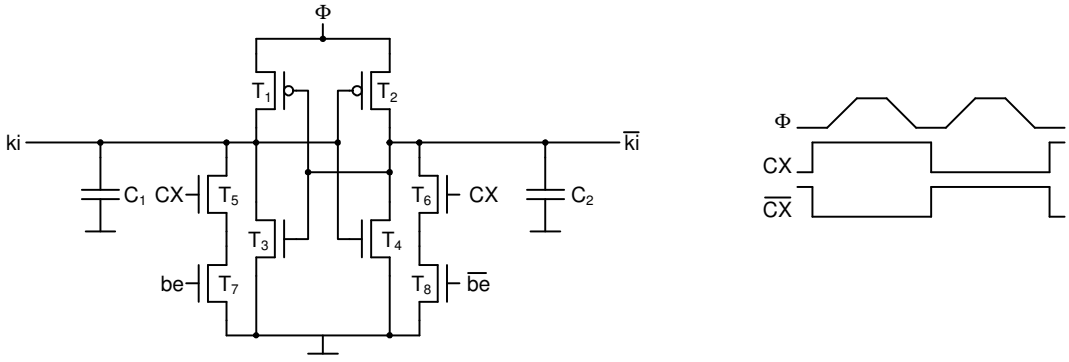


2.10. ábra. A *2N-2N2P* inverter

B.4. Órajeles adiabatikus logika (CAL)

Az órajeles adiabatikus logika [44] (*Clocked Adiabatic Logic*, röviden *CAL*), amelynek invertere a *2.11. ábrán* látható, hasonlít a *2N-2N2P* kapcsolásra. Pipeline működéséhez mindössze egyetlen adiabatikus fázisjel szükséges, azonban ezt kiegészíti két statikus segéd-fázisjel, amelyek felváltva engedélyezik a páros, illetve a páratlan fokozatokban elhelyezkedő logikai kapukat. A *2.11. ábrán* a *CX* statikus segéd-fázisjel a T_5 és a T_6 tranzisztorokat vezérli, a Φ adiabatikus-fázisjel minden második ciklusában engedélyezve csak a kapu működését.

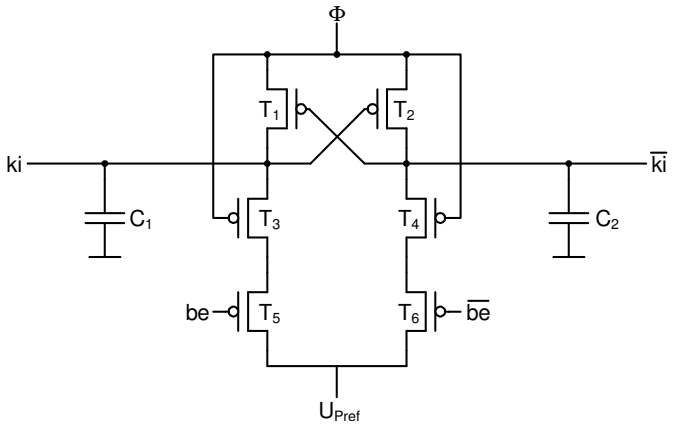
A következő fokozatban lévő kapuk működését a $\overline{CX}$ statikus segéd-fázisjel engedélyezi. Ezáltal – bár a *CAL* kapcsolás működtethető olyan nagy frekvencián, mint a *2N-2N2P* kapcsolás – átbocsátása csak fele annak. Abban a fázisjel-ciklusban, amikor a kapu működése nem engedélyezett, a kapu kimenete az előző fázisjel-ciklusban felvett kimeneti értékét ismétli, mivel a feltöltött terhelőkapacitás csak küszöbfeszültségig sül ki, és ez elég arra, hogy a lassan emelkedő Φ fázisjelre a kapu ismét az előző logikai állapotba billenjen. Mivel a statikus segéd-fázisjelek csak NMOS tranzisztorokat vezérelnek, a töltéspumpálásból adódó veszteség mérsékelhető a segéd-fázisjelek amplitúdójának csökkentésével.



2.11. ábra. A *CAL* inverter és fázisjelei

B.5. Igazi egyfázisú energia-visszanyerő logika (TSEL)

Az igazi egyfázisú energia-visszanyerő logika [46] (*True Single-phase Energy-recovering Logic*, röviden *TSEL*), amelynek kapcsolása a *2.12. ábrán* látható. Előnye, hogy pipeline működéséhez csak egyetlen szinuszos fázisjel szükséges. A *TSEL* kapcsolásban a páros, illetve a páratlan fokozatokban elhelyezkedő logikai kapuk rendre csak PMOS, illetve csak NMOS tranzisztorokból épülnek fel, amely kapuk a föld helyett rendre U_{Pref}, illetve U_{Nref} referencia feszültséghez kapcsolódnak. Az U_{Pref} elméleti legkisebb értéke $2 \cdot |U_{t,p}|$, míg az U_{Nref} elméleti legnagyobb értéke $U_{dd} - 2 \cdot U_{t,n}$. Az elméleti értékekhez közeli referencia feszültségek mellett a legkedvezőbb a kapcsolás teljesítményfelvétele, azonban a működési sebesség növeléséhez a referenciafeszültségek értékeit egymáshoz közelíteni kell.



2.12. ábra. A *TSEL* PMOS inverter

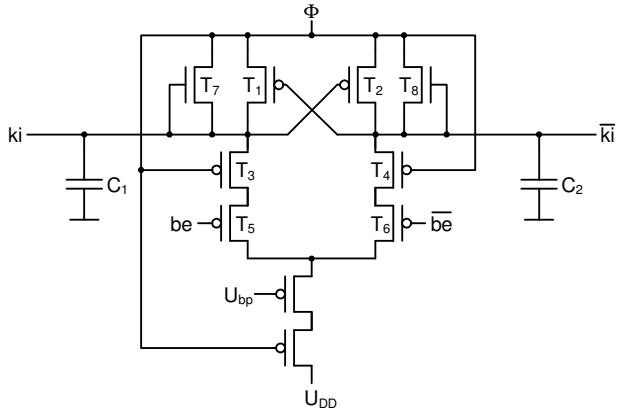
A *TSEL* PMOS kapu működése a kiértékelés és a kisütés periódusokra tagolódik. A kiértékelő periódus kezdetén a Φ fázisjel értéke alacsony, és tételezzük fel, hogy a *be* bemenet magas, így a $\overline{be}$ alacsony logikai szintű. Ahogy a Φ fázisjel értéke $|U_{t,p}|$ fölé emelkedik, a T_1 és a T_2 tranzisztorok kinyitnak. Amíg a Φ fázisjel értéke kisebb mint $U_{Pref} - |U_{t,p}|$, addig a T_3 és a T_4 tranzisztorok is vezetnek és így a C_2 kapacitás nem-adiabatikusan U_{Pref} értékére töltődik fel. A C_2 kapacitás feltöltődésével a T_1 tranzisztor lezár, majd innentől kezdve a C_2 kapacitás már adiabatikusan töltődik. Amikor a Φ órajel értéke meghaladja $U_{Pref} - |U_{t,p}|$ értékét, a T_3 és a T_4 tranzisztorok is lezárnak, így a bemeneteken történő jelváltozásnak nincs hatása a kimenetekre. A kimeneteken érvényes logikai szint van amikor a Φ órajel közel van a csúcsértékéhez. A kisütési periódusban a lefutó Φ fázisjel mind a C_1, mind a C_2 kapacitás feszültségét elméletileg $|U_{t,p}|$-ig lehúzza. Ez a kisütés adiabatikus mindaddig, amíg a Φ fázisjel értéke magasabb, mint $U_{Pref} - |U_{t,p}|$.

A *TSEL* NMOS kapu a *TSEL* PMOS kapu komplementere, működése az előtöltés és a kiértékelés periódusokra tagolódik. Az előtöltés alatt a kimenetek magas szintre töltődnek fel, míg kiértékeléskor az egyik kimenet először nem-adiabatikusan U_{Nref} feszültségre sül ki, majd a további kisütés már adiabatikusan történik.

B.6. Forráscsatolt adiabatikus logika (SCAL)

Az egyfázisú forráscsatolt adiabatikus logika [61], (*single-phase Source-Coupled Adiabatic Logic*, röviden *SCAL*), felépítésében és működésében a *TSEL* kapcsoláshoz hasonló, azonban az *SCAL* kapcsolás referencia feszültség helyett referencia áramokat alkalmaz, amelynek értékét logikai kapunként egy tranzisztor *W/L* arányával lehet beállítani. Ezáltal kiküszöbölhető a *TSEL* kapcsolás azon hátránya, hogy kisfrekvencián feleslegesen sokat fogyaszt, mivel a referencia feszültségek nem lehetnek kisebbek, illetve nagyobbak, mint a küszöbfeszültségek kétszerese.

Azonban mint a *TSEL*, mind az *SCAL* kapcsolás gondja, hogy gyakorlati megvalósításra közvetlenül nem alkalmas [69]. A PMOS kapu működéséhez ugyanis az szükséges, hogy kiértékelés előtt a kimenetek nagyjából $|U_{t,p}|$-ig kisütöttek legyenek. Azonban ha az egyik kimenet $|U_{t,p}|$-ig ki is sül, a következő ciklusban a feltöltött másik kimenet már csak $2|U_{t,p}|$-ig fog kisülni. Könnyen belátható, hogy ez hibás logikai működéshez vezet. Hasonlóképpen, az NMOS kapu működéséhez a kiértékelés előtt a kimenetek előtöltése szükséges, azonban a *2.12. ábrának* megfelelő NMOS tranzisztorokkal felépített elrendezés ezt nem biztosítja. A gyakorlatban is alkalmazható megoldáshoz a kapcsolást ki kell egészíteni a PMOS kapu kimeneteit kisütő, illetve az NMOS kapu kimeneteit előtöltő diódákkal. Egy ilyen kapcsolás az *SCAL-D* [68], amelyet a *2.13. ábra* mutat.



2.13. ábra. Az *SCAL-D* PMOS inverter

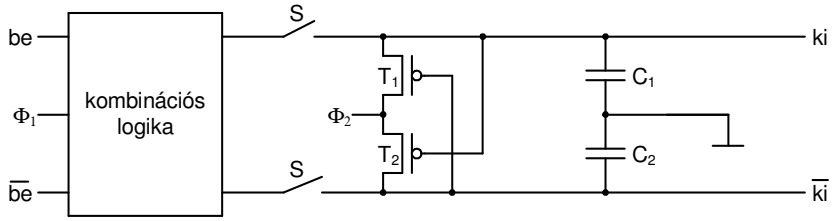
C. Adiabatikus tároló kapcsolások

A bemutatott adiabatikus logikai kapcsolásokkal bármely logikai függvény megvalósítható, sorrendi hálózatok építéséhez azonban tárolóelemek is szükségesek. Az adiabatikus tárolóelemek elkerülhetetlenül kvázi-adiabatikus működésűek, mivel a tárolt adat törléshez egy véges, le nem csökkenthető energiamennyiség szükséges [50].

A szakirodalom rendkívül szegény az adiabatikus tárolóelemek tekintetében. Az adiabatikus dinamikus tárolóelem [59] elméletileg is működésképtelen, ráadásul frissítése egy lassan működő adiabatikus logikával nem is volna feltétlenül lehetséges. Az adiabatikus regiszter [71] csak egy órajel-ciklusig képes adatot tárolni, ezért nem tekinthető tárolóelemnek. A szakirodalomban mindössze két használható kapcsolást találtam, amelyek közül az egyik nem is memória jellegű tároló, hanem inkább reteszelő áramkör.

C.1. Adiabatikus tárolóelem

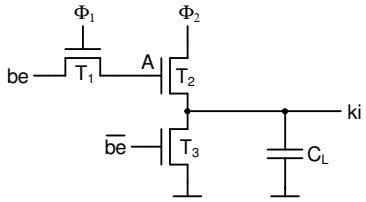
A *2.14. ábrán* látható adiabatikus tárolóelembe [50] új érték beírása a Φ_1 és a Φ_2 fázisjel alacsony szintje mellett az *S* kapcsolók zárásával kezdődik, majd felfutó Φ_1 fázisjel hatására a tároló kimenetén megjelenik a kombinációs logika által előállított érték. Ekkor a Φ_2 fázisjel is felfut, majd még a Φ_1 fázisjel lefutása előtt az *S* kapcsolók kinyitnak, leválasztva a tárolót a kombinációs hálózatról. A beírt adat tárolásához a Φ_2 fázisjel magas szintje szükséges. Új érték beírása előtt a tárolt adatot a lefutó Φ_2 fázisjel törli, amely adiabatikusan történik mindaddig, amíg a Φ_2 fázisjel értéke meghaladja $|U_{t,p}|$-t. Gond, hogy hosszú idejű tárolás nem lehetséges, mivel a szivárgási áramok hatására a kezdetben alacsony szintű kimenet feszültsége sérülhet, valamint, hogy a tároló működtetéséhez a logikáétól eltérő vezérlés szükséges.



2.14. ábra. Adiabatikus tárolóelem

C.2. Utánhúzott energia-visszanyerő dinamikus tárolóelem

Utánhúzott energia-visszanyerő dinamikus tárolóelem kapcsolása látható a *2.15. ábrán* [41], ahol kezdetben mind a Φ_1, mind a Φ_2 fázisjel alacsony szintű. Logikai magas érték beírásakor a Φ_1 fázisjellel együtt felfutó *be* bemenet az *A* pontot U_{dd} -$U_{t,n}$ feszültségre tölti, míg alacsony érték esetén az ellenkező bemenet a T_3 tranzisztor kinyitásával a C_L kapacitást leföldeli. Ha logikai magas értéket írunk be, akkor a Φ_1 fázisjel lefutása után a felfutó Φ_2 fázisjel az *A* pontot a T_2 tranzisztor *gate-csatorna* kapacitásán keresztül [21] legalább U_{dd} +$U_{t,n}$ feszültségre húzza, és így a C_L kapacitás adiabatikusan U_{dd} feszültségre töltődik. A kimeneten a beírt logikai magas érték mindaddig megőrződik, amíg a Φ_2 fázisjel magas értékű.



2.15. ábra. Utánhúzott energia-visszanyerő dinamikus tárolóelem

A kapcsolás hátránya az *A* pont diódaszerű feltöltése és nem-adiabatikus kisütése. Az ebből adódó veszteség jelentős lehet, mivel a feszültség utánhúzás hatékonysága érdekében a T_2 tranzisztort megfelelően nagy méretűre kell választani. A tároló energiahatékonysága tehát a C_L terhelőkapacitás és az *A* pont kapacitásának viszonyától függ. Például a visszanyert és a befektetett energia aránya egy összeadó bemenő adatainak reteszelésénél mindössze egyharmad [41], míg nagyméretű statikus RAM sor és oszlopcímeinek reteszelésénél ennél jóval kedvezőbb érték érhető el [63].

D. Adiabatikus fázisjel-generátorok

Az adiabatikus fázisjel-generátorokkal szemben támasztott két fő követelmény, hogy hatékonyan előállítsák az adiabatikus tápláláshoz szükséges lejtős fázisjeleket, valamint képesek legyenek a betáplált energia visszanyerésére és újbóli felhasználására. A lejtős jelalakok közelíthetők lépésenkénti töltéssel [74][75], ahol, ha a töltés *N* lépésben történik, akkor a (2.1) összefüggés szerinti energiaveszteség *N*-ed részére csökken. Az ilyen táplálás azonban a betáplált energiát kisütéskor, ha csak lépésekben is, de a földbe vezeti, ezért nem alkalmas annak újbóli felhasználására.

Egy jel értékének megváltozása energiaátalakítással jár, a betáplált energia visszanyeréséhez ezért ennek az energiaátalakításnak kétirányúnak kell lennie. A statikus CMOS kapcsolásokban az energiaátalakítás egyirányú, mivel az elektromos energia hővé alakítása nem visszafordítható folyamat. A kétirányú energiaátalakítás megvalósításához az elektromos energia mellett szükség van az energia még egy formájára, például a mágneses energiára. Ha a jelváltozást az elektromos energiának a mágneses energiára való átalakításával valósítjuk meg, akkor az energia visszanyerését is elérhetjük, mivel a mágneses energia visszaalakítható elektromos energiává, tehát a folyamat kétirányú. A mágneses energia tárolásához induktivitás szükséges, amely azonban az adiabatikus áramkörök alkalmazási frekvenciatartományában – a kívánt értékben és jóságban – a félvezetőlapkán nem valósítható meg. Ezért a gyakorlati adiabatikus fázisjel-generátorok többnyire külső induktivitást alkalmaznak [34]-[39], illetve nagyon nagy frekvencián elegendő a kikötővezetékek induktivitása is [40].

2.2.3. A teljesítményfelvétel csökkentésének rendszerszintű lehetőségei

A teljesítményfelvétel csökkentésének rendszer-szinten lehetőségei közé tartozik az egyidejű átkapcsolások során keletkező nagy áramlökések mérséklése az órajelek időbeli eltolásával [17], a digitális szűrők fokszámának változtatása a feldolgozandó jel (várható) spektrális eloszlása, illetve az adott pillanatban szükséges átviteli karakterisztika függvényében [18], a buszokon történő jelváltozások számának csökkentése inverzióval [83], átrendezéssel [19], vagy a busz feldarabolásával [84].

A magas szintű logikai szintézis során szóba jöhet az összeköttetést megvalósító elemek veszteségi teljesítményének csökkentése [85], a kis fogyasztásra történő ütemezés [86], a műveletvégző egységek felesleges működésének kiküszöbölése [87], a műveletvégző egységek [88] és a regiszterek [89] kiosztása a várható teljesítményfelvétel figyelembevételével, továbbá különböző tápfeszültség-szintek alkalmazása az ütemezésben [90].

Általánosságban elmondható, hogy a teljesítményfelvétel csökkentésére irányuló lépések annál inkább eredményesek, minél magasabb elvonatkoztatási szinten történnek. Míg tranzisztor szinten rendszerint 10-30%, addig kapcsolási szinten tipikusan (az adiabatikus áramköröket nem számítva) 30-50% javulás érhető el. Rendszerszinten akár 50-90% teljesítményfelvétel csökkentés is lehetséges. Természetesen a magas szinten történő optimálás nem zárja ki az alacsonyabb szinteken történő további energiamegtakarítás lehetőségét.

2.3. A várható fogyasztás meghatározásának módszerei

A várható fogyasztás meghatározására számos módszer létezik, köztük a magas szintű, kifejezetten digitális jelfeldolgozó áramkörök teljesítményfelvételének becslésére kidolgozott technikák [93][94], továbbá mind az alulról-felfelé [95], mind a felülről-lefelé [96] tervezésben alkalmazható megoldások.

Adiabatikus áramkörök energiaveszteségének meghatározására a fentiekhez hasonló technikák jelenleg nincsenek. Annak ellenére, hogy az adiabatikus áramkörök is digitális logikát valósítanak meg, mégis főként analóg működésűek. Ezért az adiabatikus áramkörök energiavesztesége egyszerű digitális szimulátorokkal nem határozható meg, hanem jóval bonyolultabb, analóg szimulátorok szükségesek, esetleg nemcsak az energiaveszteség, hanem – szélső esetben – a helyes működés ellenőrzésére is.

2.4. Hardverleíró nyelvek

Szabványos hardverleíró nyelvek alkalmazhatók arra a célra, hogy az „áramköri elképzelésünket”, a műszaki leírást hűen tükröző áramkörmegadást, a számítógép számára érthető alakban adhassuk meg. Számos alkalmazási terület modellezési igényeinek megfelelő, kiemelkedő jelentőségű és széles körben használt szabványosított nyelv a VHDL [20], amely elsősorban digitális áramkörök egységes leírására alkalmas. Segítségével a viselkedési megadástól a kapu szintű leírásig számos elvonatkoztatási szinten megadhatjuk a tervet, továbbá a különböző elvonatkoztatási szinteknek megfelelő leírások keverten is alkalmazhatók, támogatva a felülről-lefelé történő tervezés módszerét, valamint az időzítési viszonyok valósághű modellezését.

A. Viselkedési leírás

A viselkedési leírás számítási és vezérlési lépések megadásából áll. Magas szintű műveletek és vezérlő szerkezetek írják le az egymás után következő be-, kimeneti és számítási lépéseket. Egy-egy számítási lépés elemi műveletek halmazát, valamint szinkronizációs pontokat tartalmaz. Az időzítés eseményvezérelt, ahol egy-egy számítási lépés több órajel-cikluson keresztül is tarthat. A VHDL viselkedési modell alapját a *folyamatok* adják, amelyek között az adatátvitelt *átfogó jelek* valósítják meg. A folyamatok önmagukban sorrendi, de egymáshoz képest együttfutó programrészek. Egy folyamat működése bármikor felfüggeszthető valamilyen feltétel bekövetkeztéig, vagy akár vételen hosszú ideig. A viselkedési szintézis fő feladata, hogy a számítási lépéseket órajel-ciklusok sorozatára bontsa, és órajel-ciklusonként meghatározza, hogy mikor mely elemi műveletek kerüljenek végrehajtásra, és ezeket mely műveletvégző egységek végezzék el.

B. Regiszter-átviteli szintű leírás

A regiszter-átviteli szintű leírás során felhasznált áramköri elemek az elemi műveletvégző egységek, regiszterek, valamint az ezek közötti adatáramlást biztosító

egyéb elemek (pl. multiplexerek és buszok). A fő tervezési lépés a regiszter-átviteli szintről indulva a logikai optimálás, az órajel ciklusidejének és a logikai kapuk típusainak és ezek darabszámának meghatározása.

C. Kapuszintű leírás

A legalacsonyabb szintű áramkörmegadás a kapuszintű leírás. Ez a logikai kapuk és a közöttük lévő összeköttetések megadását jelenti. Az alapvető időzítés itt a logikai kapuk és az összeköttetések késleltetési ideje, az órajel periódusideje pedig legalább olyan nagy kell, hogy legyen, mint a legnagyobb késleltetési idő két tárolóelem között.

2.5. Magas szintű logikai szintézis

A megvalósítandó áramkör összetettségének és bonyolultságának növekedése újfajta szintézis eszközök használatára kényszeríti a tervezőket, amely szintézis eszközökkel a tervezés magasabb elvonatkoztatási szinten indítható. Mindehhez az alacsonyabb szintű tervezési lépések automatizálása teremt lehetőséget, mivel egy magasabb elvonatkoztatási szintről a regiszter-átviteli szintre eljutva már önműködő tervezőeszközök segítségével előállíthatók a gyártáshoz szükséges dokumentációk, vagy valamely programozható logikai eszközbe letölthető bitsorozat.

A viselkedési leírás elkészítése és esetleges változtatása – az elvonatkoztatási szint emelése révén – lényegesen egyszerűbb, mint a regiszter-átviteli, vagy esetleg még alacsonyabb szintű tervé, ezáltal használatával jelentősen rövidíthető a tervezési idő. Mindezeken túl a viselkedési szinten történt optimális lépések a terv minőségét is nagyban javítják.

A magas szintű logikai szintézis feladata, hogy a viselkedési leírásból a tervezési korlátoknak megfelelő regiszter-átviteli szintű tervet alakítson ki. A szintézis eredményeként előállított terv két különválasztható, de összetartozó részből áll: az *adatútból* és a *vezérlő részből* [1]. Az adatút elemi műveletvégző és adattároló egységek halmazából valamint a köztük lévő összeköttetésekből áll. A vezérlő rész, amely – általában egy központosított véges állapotú automata – az adatútban szereplő egységek működését vezérli. Minél egyszerűbbek az adatútban felhasznált elemi műveletvégző egységek, annál több szabadságfoka marad a tervezőnek, illetve a magas szintű szintézis eszköznek, hogy ezeket milyen módon kapcsolja össze. A gyakorlati magas szintű szintézis rendszerek – a vezérlő bemeneteket nem számolva – emiatt jellemzően két bemenetű, egy kimenetű műveletvégző egységeket alkalmaznak. A szintézis eredményeként regiszter-átviteli szintű kód áll elő, rendszerint valamely hardverleíró nyelven megadva.

2.5.1. A magas szintű logikai szintézis lépései

A viselkedési leírás elkészítésekor feltételezzük, hogy az egyes műveleteket megvalósító elemi műveletvégző egységek rendelkezésre állnak, így a viselkedési leírás ezen elemi műveletvégző egységek által megvalósított műveletekre való

hivatkozásokkal adható meg. A megoldandó feladat szinte minden esetben többféleképpen is leírható kétbemenetű elemi műveletekkel. A legmegfelelőbb leírás legtöbbször csak heurisztikus eszközökkel, próbálgatással érhető el. A szintézis fő feladata a műveletvégző egységek valamely feltételrendszernek megfelelő, optimális elrendezése, amelynek eredményessége függhet a kezdeti viselkedési leírás megadásától [106].

A magas szintű logikai szintézis fő lépései a következők: belső ábrázolási forma előállítása, ütemezés, erőforrás kiosztás, műveletek hozzárendelése a műveletvégző egységekhez, összeköttetés kiosztás, és regiszter-átviteli szintű leírás elkészítése. Az egyes lépések sorrendje nem általános, a magas szintű szintézis rendszerek különböző megvalósításaiban eltérő lehet.

A. Belső ábrázolási forma előállítása

A belső ábrázolási forma előállítása a számítógépes feldolgozáshoz szükséges, a tervező által adott magas szintű leíráson fordítás-szerű átalakításokat jelent [97]; például a megjegyzések eltávolítását, a ciklusok és függvényhívások feloldását, stb. A szakirodalomban található viselkedési szintézis rendszerek [99]-[103] a kiinduló leírást rendszerint vezérlési- és adatfolyam gráf [98] formára alakítják, és a szintézis további lépéseiben ezt a gráfot használják.

B. Ütemezés

Az ütemezés a terv időbeli felosztását jelenti, ahol az időzítést rendszerint órajelben vagy vezérlési lépésben mérik. Az ütemezés rögzíti, hogy a viselkedési leírásban szereplő egy-egy elemi művelet végrehajtását mely vezérlési lépésben kell indítani. Egy művelet indításának ideje egy legkorábbi és egy legkésőbbi időpont között változhat. Ez alapján két szélsőséges ütemezést különböztetünk meg, az *ASAP* (As Soon As Possible) és az *ALAP* (As Late As Possible) ütemezéseket [6]. Egy művelet *ALAP* és *ASAP* indítási időpontjainak különbsége jelenti az adott művelet *mozgékonyságát*.

Egy vezérlési lépésben természetesen több művelet végrehajtása is indítható és/vagy folyhat párhuzamosan, feltéve, hogy ehhez elegendő erőforrás áll rendelkezésre. Ennek alapján az ütemezés lehet *végrehajtási idő korlátozott* [4], *erőforrás korlátozott* [3], illetve *erőforrás és végrehajtási idő korlátozott* [5].

A végrehajtási idő korlátozott ütemezéshez nagyszámú erőforrás szükséges, időben több művelet végrehajtása párhuzamosan folyik, a vezérlő rész kevés állapotátmenettel megvalósítható. Az erőforrás korlátozott ütemezés takarékoskodik műveletvégző egységek számával: csak a műveletvégző egységek előírt legnagyobb számának erejéig engedi az azonos típusú műveletek időben átfedő végrehajtását, ezen felül azokat időben egymás utáni végrehajtásra ütemezi. Ekkor rendszerint a vezérlő rész bonyolultabbá válik a több állapotátmenet miatt, valamint jelentősen nőhet az adatút *lappangási ideje* (a bemenő adat beléptetésének időpontjától a kimeneten a hozzá tartozó eredmény megjelenésének időpontja közötti időtartam).

C. *Erőforrás kiosztás*

Az erőforrás kiosztás rögzíti az ütemezés eredményeként előállt viselkedési leírás végrehajtásához a szükséges elemi műveletvégző egységek mennyiségét és típusát, valamint a közbülső adatok tárolásához szükséges regiszterek darabszámát. Az erőforrás kiosztó algoritmus meghatározza az egyes műveletek *foglaltsági idejét*, vagyis azt az időtartamot, amíg a művelet végzése folyik, és amíg az eredményt valamely tárolóelem tárolja. Az azonos típusú, egymást át nem fedő foglaltsági idejű műveletek számára elegendő egy műveletvégző egységet kiosztani. A mozgékonysági időtartamon belüli mozgatással további lehetőségek adódhatnak az át nem fedő foglaltsági idejű műveletek felderítésére.

Az ütemezés és az erőforrás kiosztás nem független egymástól, azonban a két feladat párhuzamos megoldása esetén a műveletvégző egységek számának növekedésével a lehetséges variációk száma exponenciálisan nő [10]. Valamely szempont szerinti optimum eléréséhez nem minden esetben kerülhető el a tervezői beavatkozás, szóba jöhetnek heurisztikus megoldások és próbálgatások.

D. *Műveletek hozzárendelése a műveletvégző egységekhez*

Ez a lépés dönt arról, hogy a kiosztott műveletvégző egységek közül melyik mely elemi műveleteket hajtson végre. Sok rendszerben ezt a lépést az erőforrás kiosztás részeként valósítják meg.

E. *Összeköttetés kiosztás*

Az összeköttetés kiosztás az adatút összetevői (műveletvégző egységek és regiszterek) közötti adatátviteli utakat építi ki. Alapvetően kétfajta összeköttetési elrendezés létezik: a pont-pont összeköttetés és a megosztott erőforrásokon keresztüli összeköttetés. Az utóbbi a busz megoldást jelenti, amely kevesebb erőforrást igényel, azonban nem teszi lehetővé a párhuzamos adatátvitelt (hacsak nem osztjuk a buszt részekre), továbbá a vezérlése is bonyolult. A pont-pont összeköttetés bármely két egység között adatcsatornát alkalmaz, ahol ez szükséges. Ha egy műveletvégző egység több helyről is kap bemenő adatot, akkor az éppen szükségeset multiplexer választja ki. Az összeköttetés kiosztás eredményeként az adatút minden olyan egységet tartalmaz, amely a bemenettől a műveletvégző egységeken és a belső tárolóelemeken keresztül a kimeneti kapukig történő adatáramláshoz szükséges.

F. *Regiszter-átviteli szintű leírás elkészítése*

Ez a lépés a kezdeti viselkedési leírásnak megfelelő regiszter-átviteli szintű leírást állítja elő. A korábbi szintézis lépések eredményére támaszkodva történik az adatút, valamint az adatútnak megfelelő vezérlő rész kialakítása, amely egymás után következő állapotátmenetek során aktivizálja az adatút megfelelő egységeit.

Általánosságban ez a lépés a belső ábrázolási forma átalakítását jelenti más rendszerek, és/vagy a tervező által értelmezhető formává. Ez a forma lehet például valamely hardverleíró nyelvi kód, vagy könyvtári elemek összeköttetési listája.

2.6. Ütemező algoritmusok áttekintése

A magas szintű logikai szintézis egyik legfontosabb lépése – az erőforrás kiosztás és a műveleteknek a műveletvégző egységekhez való hozzárendelése mellett – az ütemezés [2]. Az ütemezéshez szükség van olyan adatokra, hogy mely típusú műveletvégző egységek milyen típusú műveleteket képesek végrehajtani (típus-megfeleltetés), és ehhez mennyi idő szükséges, ezen kívül további optimálási szempontokból fontos lehet az egyes műveletvégző egységek által elfoglalt szilícium-terület, a felvett teljesítmény, stb.

Mindezekből látszik, hogy ütemezéskor olyan tervezési döntések születnek, mint például a szükséges műveletvégző egységek típusai és az egyes típusok darabszáma, amelyek szorosabban véve az erőforrás kiosztáshoz, illetve a műveleteknek a műveletvégző egységekhez való hozzárendeléséhez tartoznak. Ezért nemegyszer e három lépést közös algoritmusban valósítják meg [110]-[112].

Az ütemezés fontos kérdése az ütemezett adatút *átbocsátása*, ami alatt az időegység alatt feldolgozott adatok számát értjük. Az átbocsátás növelésének egyik módja a pipeline elv alkalmazása, vagyis amikor az adatút új bemeneti adatot fogad, még mielőtt az előző számítás eredménye a kimeneti kapukon megjelenik. A digitális jelfeldolgozó alkalmazásokban az adatút rendszerint periodikusan ismétlődő adatokon ugyanazon műveletsort hajtja végre, ekkor a bemenő (és egyben a kimenő) adatok ismétlődési idejét *újraindítási időnek* nevezzük. Pipeline működésről eszerint akkor beszélünk, ha az újraindítási idő kisebb, mint a lappangási idő.

A pipeline elv alkalmazása két szinten is lehetséges [1]. Az első szintet nevezik *adatút* vagy *működési pipeline*-nek, ez a „hagyományos” értelemben vett pipeline. A második szintű pipeline az *összetevő* vagy *szerkezeti pipeline*, amikor nemcsak az adatút, hanem a benne szereplő műveletvégző egységek is pipeline működésűek, így az elemi műveletek végrehajtása is átlapolva történik. Első szintű pipeline ütemezésre valamennyi pipeline ütemező algoritmus képes [77]-[80], második szintű pipeline ütemező algoritmusra azonban csak kevés kivétel akad [7][8].

Ütemezéskor a leggyakrabban alkalmazott tervezési megkötések az újraindítási idő, a szilícium-terület, és a lappangási idő. Mindezek azonban sokszor egymásnak ellentmondó szempontok, közöttük az optimum csak a *tervezési tér* (lehetséges megoldások halmaza) teljes átvizsgálásával található meg. Mivel a gyakorlati feladatok olyan nagy méretűek és oly összetettek, hogy a *tervezési tér* teljes átvizsgálása számítási teljesítmény és futási idő korlátok miatt nem lehetséges, nagy szerepet kapnak a heurisztikus megoldások, próbálgatások. A gyakorlatban ezért nem feltétlenül törekednek a lehető legjobb megoldásra, ha egy kielégítő megoldás könnyen elérhető.

A heurisztikus ütemezési algoritmusok *konstruktív* jellegűek, a műveleteket egymás után kiválasztva, egyszerre csak egy műveletet ütemeznek, szemben a nem heurisztikus, átfogóan optimális megoldást adó *egészértékű lineáris programozáson* alapuló ütemezéssel, amely valamennyi műveletet egyidejűleg ütemezi.

A két legegyszerűbb heurisztikus ütemezési algoritmus a már említett *ASAP* és *ALAP* ütemezés, amelyek végtelen számú erőforrást tételeznek fel [6], azonban az algoritmus módosításával erőforrás korlátok is figyelembe vehetők [82]. Fő ütemező

algoritmusként *ASAP* és *ALAP* ütemezéseket csak a korai szintézis rendszerek alkalmaztak, azonban eljárásaik más ütemező algoritmus részeként szinte minden rendszerben előfordulnak. Két további, általánosan használt heurisztikus ütemezési algoritmus a *lista ütemezés* és az *erőirányított ütemezés*.

A. Lista ütemezés

A lista ütemezés [3] az erőforrás korlátozott ütemezési probléma megoldására alkalmas, különböző formában számos szintézis rendszer alkalmazza. A lista ütemező vezérlési lépésről vezérlési lépésre haladva – vezérlési lépésenként, vagy minden egyes művelet ütemezése előtt – az ütemezésre kész műveletekből egy listát állít össze. Az ütemezésre kész műveletek azok a műveletek, amelyek az adott vezérlési lépésbe ütemezhetők, mivel valamennyi megelőző művelet ütemezése már megtörtént és a végrehajtásukhoz szükséges erőforrások is rendelkezésre állnak. Az algoritmus az ütemezésre kész műveletekből egy elsőbbségi függvény alapján egyszerre csak egy műveletet választ. Az elsőbbségi függvény nagyon sokféle lehet, szerepet kaphat benne a műveletek mozgékonysága, az adott műveletet közvetlenül követő műveletek darabszáma, stb.

B. Erőirányított ütemezés

Az erőirányított ütemezés [4] az azonos típusú műveleteket igyekszik egyenlően elosztani a vezérlési lépések között, így a végrehajtási idő korlátozott ütemezési probléma megoldására használható. A műveletek egyenletes elosztása növeli a műveletvégző egységek kihasználását, és ezzel lecsökkenti a szükséges erőforrások számát.

C. Egészértékű lineáris programozáson alapuló ütemezés

Az ütemezési feladatot úgy is meg lehet oldani, hogy a viselkedési leírás műveleteinek egymástól való függését, valamint a tervezési korlátokat matematikai egyenlőségek, illetve egyenlőtlenségek alakjában írjuk le [5]. Ez a gyakorlatban lineáris egyenlőségeket, illetve egyenlőtlenségeket jelent, amelyek változói nem-negatív egész számok. A tényleges ütemezés ezen egyenlőségek és egyenlőtlenségek megoldását jelenti, amely általában valamely kifejezetten erre a célra kifejlesztett szoftvercsomaggal történik.

Az egészértékű lineáris programozáson alapuló ütemezés előnye, hogy optimális megoldást ad mind az erőforrás korlátozott, mind a végrehajtási idő korlátozott, továbbá mind az erőforrás és végrehajtási idő korlátozott ütemezési problémákra. Ezt az biztosítja, hogy az előbb említett egyenlőségekhez és egyenlőtlenségekhez célfüggvényeket adhatunk, és megoldáskor a célfüggvények legnagyobb vagy legkisebb értékét írjuk elő. Az optimális megoldásnak azonban ára van, ami az algoritmus nagy számítási-teljesítmény igényében és hosszú futási idejében jelentkezik. Ezért az egészértékű lineáris programozáson alapuló ütemezés a gyakorlatban csak kis- és középméretű problémák megoldására alkalmazható.

3. Kisfogyasztású CMOS logikai áramkörök vizsgálata és tervezési eljárásuk kidolgozása

Az elért tudományos eredményeimet a tézispontok köré csoportosított egy-egy alfejezetben mutatom be.

A 3.1. alfejezetben a javított tulajdonságú adiabatikus kapcsolásomat ismertetem, egy inverter segítségével bemutatom a működését, továbbá megmutatom az általános logikai kapu felépítését. A kapcsolásom tulajdonságait összehasonlítom a szakirodalomban található egyéb adiabatikus kapcsolások, valamint a statikus CMOS kapcsolás megfelelő tulajdonságaival. Ismertetem továbbá az általam kidolgozott adiabatikus tárolóelem felépítését és működését, és egy lehetséges alkalmazási példát mutatok.

A következő alfejezetben az adiabatikus kapcsolások áramútjaiban elhelyezkedő tranzisztorok energiaveszteség szempontjából történő méretoptimálásával foglalkozom. Ezen tranzisztorok vezetési csatornaszélességének helyes megválasztása az energiaveszteség adott működési frekvencián való minimalizálását teszi lehetővé. Az itt bemutatott eredményeim – bár az általam kidolgozott kapcsolásra épülnek – könnyen általánosíthatóak, így nemcsak arra, hanem általában az adiabatikus kapcsolásokra érvényesek.

Dolgozatom 3.3. alfejezete átvezetést nyújt az utolsó tématerülethez, tézis hozzá nem kapcsolódik. Témája a digitális jelfeldolgozó áramkörök magas szintű szintézise. Itt egy olyan VHDL alapú magas szintű szintézis eljárást ismertetek, amely a szintézis során a nyelvi szimulációs lehetőségek és a szimulációs környezet megtartása érdekében VHDL nyelvi átalakításokat alkalmaz.

E fejezet utolsó alfejezete visszakanyarodik az adiabatikus áramkörökhöz, pontosabban ezek magas szintű szintéziséhez. Ezen belül is a magas szintű logikai szintézis egyik legfontosabb problémakörével, az ütemezéssel foglalkozom. Az adiabatikus áramkörök ütemezése több szempontból is eltér az irodalomban található, „hagyományos” ütemezéstől, ezért ezek ütemezéséhez új eljárások kidolgozása szükséges.

3.1. Javított tulajdonságú adiabatikus kapcsolás

A szakirodalomban található teljesen adiabatikus működésű kapcsolásokkal nagyon kis teljesítményfelvételt el lehet érni, azonban ehhez a kis teljesítményfelvételhez tartozó működési sebesség is nagyon alacsony. Általánosságban ezek a kapcsolások csak alacsony frekvencián hatékonyak, magasabb frekvenciákon pedig már nem is működnek helyesen.

A kvázi-adiabatikus működésű áramkörök legkisebb teljesítményfelvétele ugyan nagyobb, mint a teljesen adiabatikus működésű áramköröké, azonban az a működési sebesség is nagyobb, amin az ilyen kapcsolások még helyesen működnek. Általánosságban ezek a kapcsolások széles frekvenciatartományban használhatók, bár nagyon kis frekvenciákon kevésbé hatékonyak.

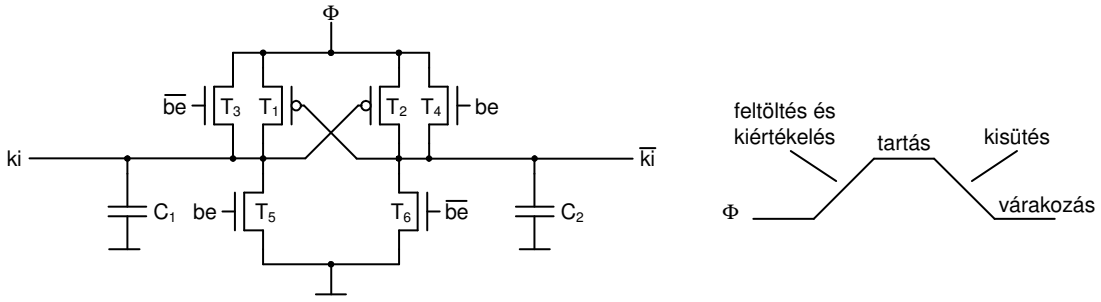
Célom egy olyan javított tulajdonságú adiabatikus kapcsolás kidolgozása volt, amely ötvözi a teljesen adiabatikus működésű kapcsolások kis energiaveszteségét a kvázi-adiabatikus működésű áramkörök nagyobb működési sebességével. A szakirodalomban található – a 2. fejezetben áttekintett – adiabatikus kapcsolásokban a legkisebb kapcsolási idő és az időállandó arányának vizsgálatával megállapítottam, hogy a keresztbekötött PMOS tranzisztorpárt alkalmazó kapcsolások működési sebességben felülmúlnak minden más elrendezést.

3.1.1. A kidolgozott javított tulajdonságú adiabatikus kapcsolás

A többi adiabatikus kapcsoláshoz mérten kiváló működési-sebesség tulajdonságai alapján a javított tulajdonságú adiabatikus kapcsolásom alapjául én is a keresztbekötött PMOS tranzisztorpárt alkalmazó elrendezésből indultam ki. Ez a keresztbekötött elrendezés az alapja a statikus áramkörökben is használt reteszelő kapcsolásnak [33], továbbá a *PAL* [45], a *2N-2P* [57], a *2N-2N2P* [58], a *CAL* [44], valamint a *TSEL* [46] és az *SCAL* [61] logikáknak.

A felsorolt áramkörök vizsgálatával megállapítottam, hogy a reteszelő kapcsolás adiabatikus aktiválása önmagában nem teszi lehetővé a magasabb frekvenciákon történő működtetést, ehhez a fel nem töltendő kimenet földelése, vagy a feltöltendő kimenet részben nem-adiabatikus feltöltése szükséges. A javított tulajdonságú adiabatikus kapcsolásomat mindezen szempontok figyelembevételével dolgoztam ki.

A kapcsolásom alapinvertere, valamint a vezérlését és egyben a tápellátását is biztosító fázisjele a *3.1. ábrán* látható [S24]. A *be* bemenetnek mind a ponált, mind a negált logikai szintjére szükség van a helyes működéshez, és az inverter *ki* kimenete is mind a ponált, mind a negált logikai szintet szolgáltatja, ezáltal kimenetei a következő fokozatban elhelyezkedő logikai kapu vezérlésére közvetlenül felhasználhatók, továbbá az inverternek a fázisjel felé mutatott kapacitív terhelése független a logikai állapotától. Ezen túlmenően a magas logikai szintű kimenet a tápfeszültség értékét veszi fel, míg az alacsony logikai szintű kimenet nulla értékű lesz. A fel nem töltött kimenet a fázisjel ciklusidejének egy kis részében lebeg, azonban ez nem okoz gondot, mivel egyrészt feszültség utánhúzás (*bootstrap*) nem történik, másrészt a kapcsolást alapvetően gyors működésre készítettem, így ez a lebegési idő sokkal rövidebb annál, hogy a logikai szintek sérüljenek.



3.1. ábra. A kidolgozott javított tulajdonságú adiabatikus kapcsolás invertere és fázisjele

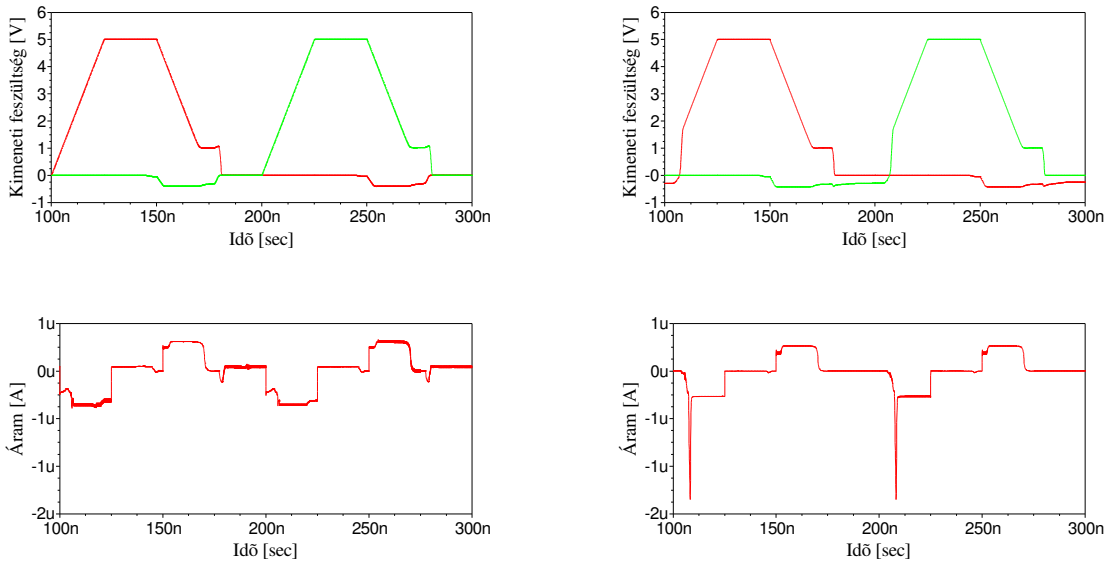
A. A javított tulajdonságú adiabatikus inverter működése

A *3.1. ábrán* látható inverter működésének szemléltetésére tételezzük fel, hogy a Φ adiabatikus fázisjel kezdetben alacsony szintű, a C_1 és a C_2 kapacitások töltetlenek, valamint a bemenetek a fázisjel felfutása alatt változatlanok, és a *be* bemenet magas, így a $\overline{be}$ alacsony logikai szintű. Ekkor a T_4 és a T_5 tranzisztorok nyitottak, így a C_2 kapacitás a Φ fázisjelhez, míg a C_1 kapacitás földre kapcsolódik. A T_3 és a T_6 tranzisztorok zártak. A felfutó Φ fázisjel a nyitott T_4 tranzisztoron keresztül adiabatikusan tölteni kezdi a C_2 kapacitást, így a $\overline{ki}$ kimenet feszültsége követi a Φ fázisjel értékét. A C_1 kapacitás feltöltetlen marad, mivel a T_1 és a T_3 tranzisztorok zártak, a T_5 tranzisztor pedig még mindig nyitott. Ha a felfutó Φ fázisjel értéke eléri a PMOS tranzisztor küszöbfeszültségének abszolút értékét, azaz $|U_{t,p}|$-t akkor a T_2 tranzisztor is kinyit, és innentől kezdve a Φ órajeltől a C_2 kapacitás, azaz a $\overline{ki}$ kimenet felé két áramút van. Az átvitelt végül egyedül a T_2 tranzisztor fejezi be, mivel a T_4 tranzisztor lezár amikor a Φ fázisjel értéke $U_{t,n}$ feszültségnyire megközelíti a *be* bemenet feszültségét.

Amint a Φ fázisjel eléri a legnagyobb értékét, a kimeneteken érvényes logikai szint van. Mivel a kapcsolás gyakorlati alkalmazásában a Φ fázisjel tartása alatt a bemeneteken lévő logikai szintek érvénytelenekké válnak (mind a ponált, mind a negált bemenet alacsony logikai szintű lesz), a T_4 tranzisztor a továbbiakban zárva marad és a lefutó Φ fázisjel a C_2 kapacitást a – részben a feltöltéshez is használt – T_2 tranzisztoron keresztül adiabatikusan $|U_{t,p}|$ feszültségig süti ki, mivel a T_2 tranzisztor is lezár, amikor Φ értéke $|U_{t,p}|$ alá csökken. A C_2 kapacitásban maradt $|U_{t,p}|$ feszültség, ami a következő fázisjel-ciklusban – az inverter új logikai állapotától függően – vagy a T_4 tranzisztoron keresztül a még alacsony Φ fázisjel felé, vagy a T_6 tranzisztoron keresztül a föld felé nem-adiabatikusan sül ki, mindkét esetben a (2.10) összefüggés szerinti energiaveszteséget okoz.

A *3.2/a ábra* felső részén a javított tulajdonságú adiabatikus kapcsolás inverterének kimeneti feszültség jelalakjai láthatók (mind a ponált, mind a negált kimenetet feltüntetve). A jelalakokat szimulációval kaptam, trapéz alakú, egyenlő feltöltés, tartás, kisütés és várakozás időkkel rendelkező tápláló fázisjelet, valamint periódusonkét változó bemeneti logikai értéket feltételezve. Lentebb az invertert tápláló fázisjelből ki- illeve befolyó áram jelleggörbéjét tekinthetjük meg. Az ábrán az inverterbe befolyó áram negatív előjelű. A *3.2/b ábrán* – összehasonlításképpen – a *2N-2P* kapcsolás hasonló jelalakjait tüntettem fel, a javított tulajdonságú adiabatikus kapcsolásával megegyező léptékben.

A *3.2 ábráról* leolvasható, hogy a javított tulajdonságú adiabatikus kapcsolás inverterébe befolyó áram közel állandó, ami az adiabatikus töltés egyik feltétele. A *2N-2P* kapcsolás esetén ugyanez csak a PMOS tranzisztor kinyitásától kezdve igaz.



a) b)

3.2. ábra. a) A javított tulajdonságú adiabatikus kapcsolás inverterének kimeneti feszültség, valamint az iverterbe befolyó áram jelalakjai. b) A *2N-2P* kapcsolás inverterének kimeneti feszültség, valamint az iverterbe befolyó áram jelleggörbéi.

B. A javított tulajdonságú adiabatikus logikai kapuk pipeline működtetése

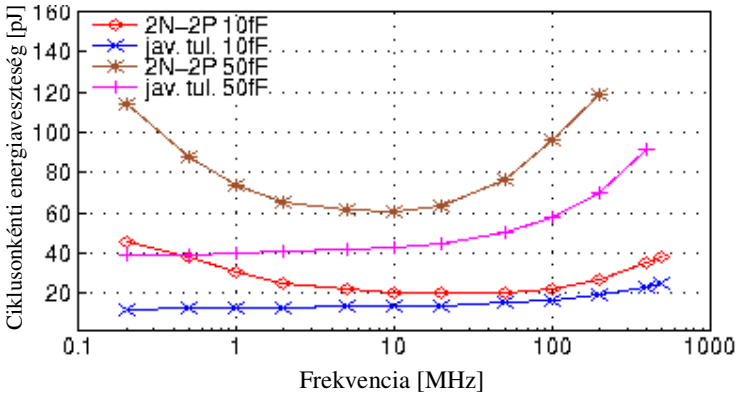
A javított tulajdonságú adiabatikus kapcsolás négyfázisú vezérléssel pipeline működtethető, a *2.9. ábrával* megegyező elrendezésben. A *3.1. ábra* szerinti elrendezéssel teljesen kiküszöböltem a terhelőkapacitás feltöltése alatt a nem-adiabatikus veszteséget, azonban ez az inverter a *2N-2P* és a többi rokon kapcsolás inverteréhez képest – a vezetékezés kapacitását nem számítva – kétszeres kapacitív terhelést jelent a bemeneteit meghajtó áramkörre nézve. Tehát egy inverter láncban minden kapu terhelése kétszeres, ami nagyobb adiabatikus veszteséget jelent, azonban ezt ellensúlyozza egyrészt a kettős áramút által lecsökkent ellenállás, másrészt a megszüntetett nem-adiabatikus veszteség.

További előnye a kapcsolásnak, hogy – a *3.1. ábrára* visszatérve – a C_2 kapacitás feszültsége kezdettől fogva követi a fázisjel értékét, így a T_1 tranzisztor egyáltalán nem nyit ki, és így – a szivárgó áramoktól eltekintve – nem folyik keresztbe áram, ezért nem keletkezik ebből adódó veszteség sem. Továbbá ugyanezen okból, valamint a C_2 kapacitás teljesen adiabatikus töltése következtében nincs áramlökés sem a tápláló fázisjelen, ami a vezetékezés ellenállásán feszültségesést, valamint a fázisjel jelalakjában torzulást és ezáltal veszteséget okozna.

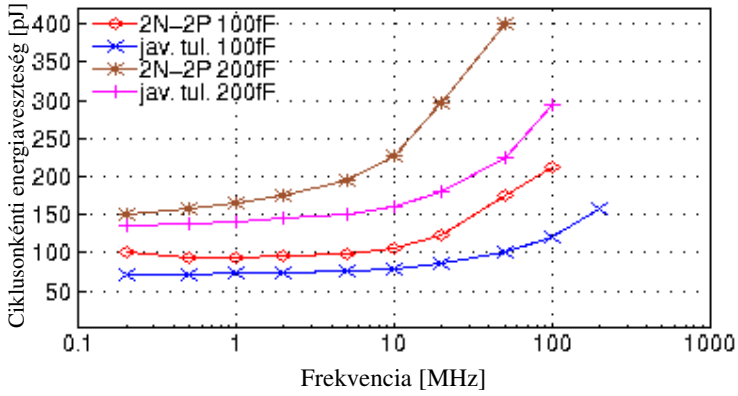
Összességében tehát az energiaveszteség a *3.1. ábra* szerinti elrendezéssel csökkenthető, ezen állításomat szimulációkkal igazoltam. A szimulációkat 100 darab láncba kapcsolt inverteren végeztem, ahol a kidolgozott kapcsolásom és a legegyszerűbb rokon kapcsolás, a hatékony töltés-visszanyerő logika, azaz a *2N-2P* kapcsolás órajel-ciklusonkénti energiaveszteségét vetettem össze a frekvencia függvényében, különböző terhelőkapacitás értékek mellett. A szimulációkat ideális

fázisjel-generátort feltételezve trapéz alakú fázisjelekkel végeztem, egyenlő feltöltés, tartás, kisütés és várakozás időkkel. A szimulációk során az Osztrák Microsystems cég 0.8 µm-es technológiáját használva, minden esetben legkisebb méretű tranzisztorokat alkalmaztam.

A szimulációs eredmények 10 fF és 50 fF terhelőkapacitások mellett a *3.3. ábrán*, 100 fF és 200 fF terhelőkapacitásokkal pedig a *3.4. ábrán* láthatók. A szimulációs eredmények szerint a *3.1. ábra* szerinti elrendezéssel az inverter láncon a teljes energiaveszteség közel a felére-negyedére csökkenthető.



3.3. ábra. 100 láncba kapcsolt inverter energiavesztesége10 fF és 50 fF terhelőkapacitások mellett



3.4. ábra. 100 láncba kapcsolt inverter energiavesztesége100 fF és 200 fF terhelőkapacitások mellett

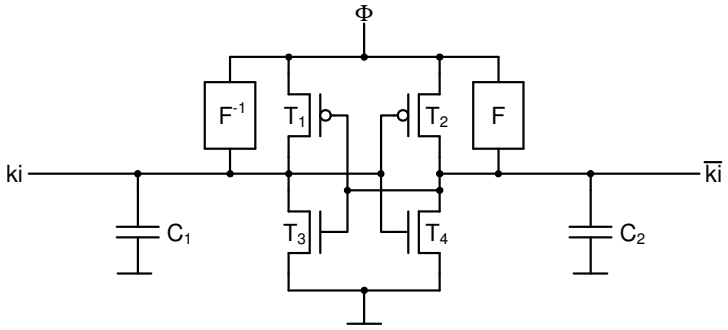
A *2N-2P* kapcsolás energiavesztesége kis frekvenciákon és kis terhelőkapacitás mellett igen jelentős, ami elsősorban nem a szivárgási áramokból, hanem a lassú kapcsolási idő miatt a hosszabb ideig keresztbefolyó áramokból adódik. Mivel ekkor a keresztbefolyó áramokból adódó veszteség meghaladja a terhelőkapacitások töltéséből adódó veszteségeket, ezért ennek kiküszöbölésével az elérhető energiamegtakarítás itt különösen látványos. A magasabb frekvenciatartományban mindkét kapcsolás vesztesége növekszik, azonban a kidolgozott kapcsolásé mindvégig a *2N-2P* kapcsolás vesztesége alatt marad, mindamellett, hogy a működési határfrekvenciája is magasabb.

Mindkét kapcsolásra jellemző, hogy a működés felső határfrekvenciájához közeledve az energiaveszteségnek a működési frekvenciától való függése az addigi

lineáris helyett négyzetes lesz, mivel az adiabatikus töltési feltétel már csak kismértékben teljesül, és a tranzisztorok bekapcsolási ellenállásán eső feszültség növekvő nem-adiabatikus veszteséget eredményez. Azonban a *3.1. ábra* szerinti elrendezéssel az elérhető felső határfrekvencia minden esetben nagyobb, mint a *2N-2P* kapcsolás legnagyobb működési frekvenciája. A szimulációk során az ideális generátor mellett a vezetékezés is ideális volt, ezáltal az áramlökésekből adódó veszteséget ez a szimuláció nem vette figyelembe. Ezért a valóságban az itt közöltnél még jobb eredmény várható.

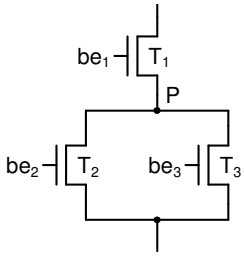
C. Általános logikai kapu kialakítása

A javított tulajdonságú adiabatikus kapcsolással felépített általános logikai kapu elrendezése a *3.5. ábrán* látható. Az *F* és az F^{-1} inverz logikai függvények kizárólag NMOS tranzisztorokból épülnek fel, és csak az egyik logikai függvény igaz értékű, ha a bemeneti logikai szintek érvényesek, illetve egyik sem, ha a bemeneti logikai szintek nem érvényesek. A *3.5. ábrán* látható elrendezésben a logikai függvények megkettőzése a gyakorlatban nem jelenti a szükséges tranzisztorszám kétszereződését, mivel ez a logikai függvények tényleges megvalósításakor többkimenetű függvényegyszerűsítéssel mérsékelhető.



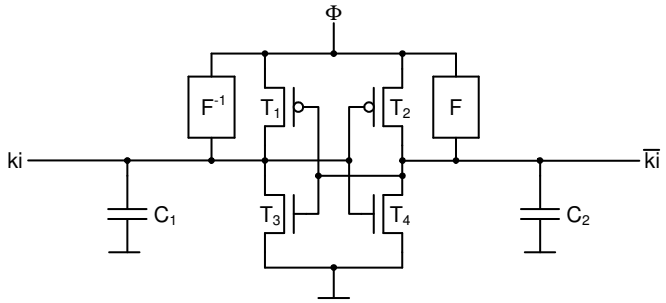
3.5. ábra. A kidolgozott javított tulajdonságú adiabatikus kapcsolással felépített általános logikai kapu elrendezése

A javított tulajdonságú adiabatikus kapcsolás inverterének minden pontja – a jelen esetben csak $|U_{t,p}|$ feszültségig – adiabatikusan kisüthető, azonban ez – akárcsak az összes többi adiabatikus logika esetén – nem igaz egy általános logikai kapura. Egy általános logikai kapu nem reverzibilis olyan értelemben, hogy a kapcsolás nem minden belső – parazita – kapacitása süthető ki adiabatikusan. Példaként tekintsük a *3.6. ábrán* található elrendezést, és tegyük fel, hogy ez a *3.5. ábra F* logikai függvényének egy része.



3.6. ábra. Az F logikai függvény egy része

Mivel a bemenetek még a Φ órajel visszafutása előtt érvénytelenekké válnak (azaz mind a ponált, mind a negált bemenet alacsony logikai szintű lesz), ezért az *F* logikai függvény belső, adott esetben feltöltött *P* pontja nem tud adiabatikusan kisülni. Ez a *source*- és *drain*-kapacitásokban tárolt töltés a következő fázisjel-ciklusban az új bemeneti kombináció előálltakor vagy a föld, vagy az ekkor még alacsony Φ fázisjel felé nem-adiabatikusan sül ki. Az ebből adódó energiaveszteség természetesen függ az egymás után következő bemeneti kombinációktól, tehát nem határozható meg előre. Általánosan elmondható azonban, hogy egy bonyolult *F* és F^{-1} inverz logikai függvényekkel rendelkező összetett kapu esetén a belső pontok nem-adiabatikus kisütéséből adódó veszteség a kapu összetettségével nő. Ugyancsak a kapu összetettségével nő járulékos elemek szivárgási és küszöbalatti áramaiból adódó veszteség, ezért a kapu összetettsége egy adott határon túl nem növelhető. A logikai kapu összetettségének csökkentésére megoldás lehet a logikai szintek számának növelése és az elvégzendő műveletek elosztása az egyes fokozatok között, vagy ha ez – például a lappangási idő megnövekedése miatt – nem járható út, akkor az összetettség mérsékelhető a *3.7. ábrán* látható elrendezéssel.



3.7. ábra. A kidolgozott javított tulajdonságú adiabatikus kapcsolással felépített általános logikai kapu elrendezése bonyolult összetett kapu esetére (amennyiben gyors működésre nincs szükség)

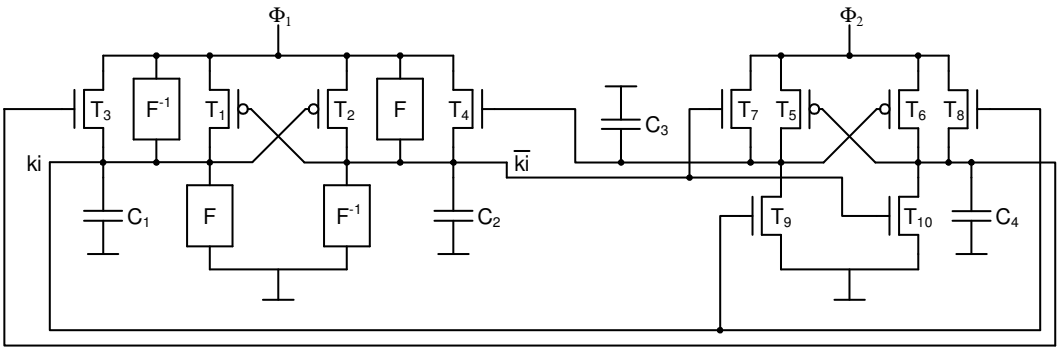
A választást a *3.5.* és a *3.7. ábrán* látható elrendezés között a tényleges kapcsolás és parazitái ismeretében – gondosan választott bemeneti kombináció sorozat mellett – szimulációval kell eldönteni. A *3.7. ábra* szerinti elrendezést azonban nem javaslom, mert használatával az elérhető működési sebességben kell árat fizetni. A kezdetben nem földelt fel nem töltendő kimenet ugyanis nem teszi lehetővé a reteszelő rész gyors billenését, ami magasabb frekvencián hibás működéshez vezet. Megjegyzendő, hogy a *3.7. ábra* szerinti elrendezésre a legtöbb esetben nincs is szükség, és ezért ezt a továbbiakban nem is alkalmazom. Az adiabatikus áramkörök jellemző alkalmazása az aritmetikai egységek megvalósítása, ahol az olyan műveletek, mint a *kizáró-vagy* és az *és-vagy*, amelyek az összeadók és a szorzók alapjai, elég egyszerűek ahhoz, hogy a *3.5. ábra* szerinti elrendezéssel egy összetett kapuban hatékonyan megvalósíthatók legyenek, és a *3.7. ábra* szerinti elrendezés alkalmazására ne legyen szükség.

D. Teljesen adiabatikus működés

A *3.5. ábrán* látható elrendezés a feltöltés és kiértékelés periódus alatt teljesen adiabatikus működést tesz lehetővé, mindez azonban nem biztosított a kisütési

periódusban, amikor a feltöltött terhelőkapacitásban $|U_{t,p}|$ feszültség adiabatikusan kisütetlen marad. Az ebből adódó, a (2.10) képlet szerinti energiaveszteség is kiküszöbölhető azonban, ha az adiabatikus kisütést lehetővé tesszük egy új – a feltöltéshez használt áramúttól különböző – kisütő áramút biztosításával [S24].

A kisütő áramúttal szembeni követelmények, hogy a terhelőkapacitást nulla feszültségig ki tudja sütni, valamint hogy az ellenállása – és így a kisütés adiabatikus vesztesége – kicsi legyen. Mindezen követelmények figyelembevételével egy olyan kapcsolást dolgoztam ki, amely a kisütő áramútban – a meglévő PMOS tranzisztor mellett – csak egyetlen NMOS tranzisztort alkalmaz, amelynek fázishelyes vezérlését a következő fokozatban elhelyezett inverter adja [S20]. A javított tulajdonságú adiabatikus kapcsolás teljesen adiabatikus működésű elrendezését a *3.8. ábra* mutatja, ahol a T_3 és a T_4 tranzisztorok alkotják a kisütő áramutakat. Az ábrán feltüntettem a következő fokozatban elhelyezett – a kisütő áramút vezérléséhez szükséges – invertert is.



3.8. ábra. A kidolgozott javított tulajdonságú adiabatikus kapcsolás teljesen adiabatikus működésű logikai kapujának elrendezése és a kisütő áramút fázishelyes vezérlése a következő fokozatban elhelyezett inverterrel

Az inverter a következő fokozatban azért szükséges, mert mindig csak a feltöltött kimenethez tartozó kisütő áramutat szabad engedélyezni, és mivel az adiabatikus inverter bemeneti logikai állapotát egy az egyben képzi le a kimenetére, felhasználható arra, hogy az előző fokozatban lévő logikai kapu kimeneteit ismételje. Ha a kisütési periódusban mindkét kisütő áramút engedélyezve lenne, akkor a fel nem töltött terhelőkapacitás is feltöltődne, tönkretéve ezzel a kimeneti logikai szinteket, ráadásul nem-adiabatikusan töltődne fel, nagy energiaveszteséget okozva.

A *3.8. ábra* szerinti elrendezés működése a következő. Tegyük fel, hogy az aktuális bemeneti kombinációra az F logikai függvény igaz logikai értékű, ekkor az F^{-1} inverz logikai függvény értéke hamis, valamint hogy a Φ_1 fázisjel alacsony szintű és éppen kezd felfutni (feltöltés és kiértékelés periódus). Ekkor a Φ_2 fázisjel alacsony szintű (várakozási periódus). A felfutó Φ_1 fázisjel a C_2 kapacitást adiabatikusan tölti, míg a C_1 kapacitás feltöltetlen marad. Amikor a Φ_1 fázisjel eléri a legnagyobb értékét (tartási periódus), akkor a Φ_2 fázisjel éppen kezd felfutni.

A Φ_1 fázisjel tartási periódusa alatt a felfutó Φ_2 fázisjel a C_3 kapacitást adiabatikusan tölti, míg a C_4 kapacitás feltöltetlen marad. Ugyancsak a Φ_1 fázisjel tartási periódusa alatt a logikai kapu bemenetei érvénytelenekké válnak, mivel a *2.9. ábrának* megfelelő pipeline működés szerint – a *3.8. ábrán* nem szereplő –

bemeneti kombinációt előállító fokozat Φ_4 fázisjele ekkor lefut. Az F logikai függvényen keresztül tehát a Φ_1 fázisjel kisütési periódusában nem lesz áramút.

A lefutó Φ_1 fázisjel a nyitott T_2 tranzisztoron keresztül megkezdi a C_2 kapacitás adiabatikus kisütését. Amikor a Φ_1 fázisjel – és vele együtt a C_2 kapacitás feszültsége – U_{dd} -$U_{t,n}$ alá csökken, akkor a T_4 tranzisztor is kinyit, mivel a C_3 kapacitás tápfeszültségre töltődött, és innentől kezdve a C_2 kapacitástól Φ_1 fázisjel felé két áramút van. Az átvitelt végül egyedül a T_4 tranzisztor fejezi be, mivel a T_2 tranzisztor lezár, amikor a Φ_1 fázisjel értéke $|U_{t,p}|$ feszültség alá csökken. A teljes ciklusban tehát, a C_2 kapacitásnak mind a feltöltése, mind a kisütése teljesen adiabatikusan történt.

Hasonló lenne a helyzet, ha a C_1 kapacitást töltöttük volna fel, így a *3.8. ábra* szerinti elrendezés biztosítja az ábrán a Φ_1 fázisjelhez kapcsolt logikai kapu teljesen adiabatikus működését. Nem megoldott azonban a C_3, illetve a C_4 kapacitások teljesen adiabatikus kisütése, így az ábrán a Φ_2 fázisjelre kapcsolt inverter teljesen adiabatikus működése. A probléma hasonló a pipeline-elven működő logika [52], valamint a *tRERL* [53] és az *nRERL* [54] kapcsolásokban utolsó fokozat adiabatikus kisütésének kérdésével, mivel az inverter-lánc folytatásával a C_3 és a C_4 kapacitások is teljesen adiabatikusan kisüthetőek lennének, csak a legutolsó fokozat kimenete nem.

A *3.8. ábra* szerinti elrendezés gyakorlati alkalmazhatóságának megítélésekor két különböző esetet kell figyelembe venni. Az első eset az, amikor a logikai kapu után következő fokozatban az inverter adott, míg a második esetben a logikai kaput nem követi inverter.

D.1. A következő fokozatban meglévő inverter felhasználása

Ha a logikai kapu után következő fokozatban az inverter adott, akkor a kisütő áramutak két NMOS tranzisztor hozzáadásával megvalósíthatók, és a gyakorlati alkalmazásokban – az energiaveszteség csökkentésére törekedve – mindig megéri ezt megtenni.

Külön megjegyezném, hogy a következő fokozatban meglévő inverter az adiabatikus kapcsolásokban egyáltalán nem ritka. Ha egy logikai kapu bemeneti értékeit különböző fokozatokban elhelyezkedő egyéb logikai kapuk állítják elő, akkor ezeket a bemeneti értékeket össze kell szinkronizálni [70], hogy fázishelyesen rendelkezésre álljanak. A szinkronizáláshoz inverterek szükségesek, amelyek fáziskéséssel ismétlik az előző fokozat kimeneti állapotát. Ugyanígy, a pipeline elven működő kapcsolásokban a pipeline fenntartására – ha az adott fokozatban logikai műveletre nincs szükség – invertereket alkalmaznak. Erre mutat példát a *3.14. ábrán* bemutatott számláló kapcsolás, amely bitenként három invertert tartalmaz, valamint a *3.22. ábrán* található összeadó kapcsolás, ahol látható, hogy a pipeline fenntartására használt inverterek száma jelentős.

D.2. Inverter hozzáadása a következő fokozathoz

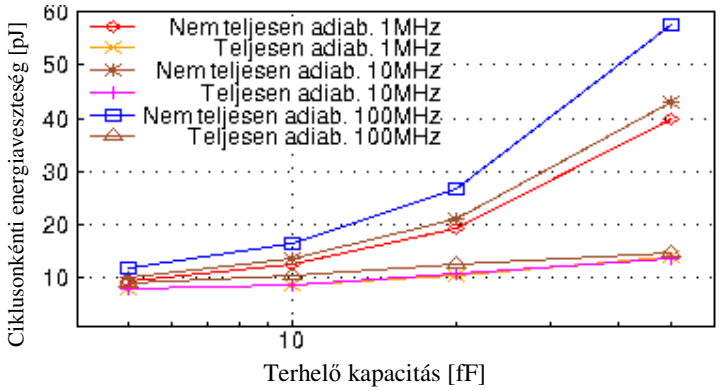
Ha a logikai kaput nem követi inverter, akkor *3.8. ábra* szerinti elrendezés alkalmazása megfontolást igényel, mivel a logikai kapu ugyan teljesen adiabatikus

működésűvé válik, azonban az így visszaforgatott energiát össze kell vetni azzal az energiaveszteséggel, amit a logikai működés szempontjából feleslegesen beszúrt kisütő inverter okoz. Előnyös azonban, hogy a kisütő inverter terhelőkapacitása alacsony értéken tartható, mivel a kisütő inverter a félvezetőlapkán fizikailag közel helyezhető a logikai kapuhoz és kimenetei csak egy-egy NMOS tranzisztort vezérelnek, így mind az adiabatikus, mind a nem-adiabatikus vesztesége, továbbá – mivel kevés számú elemből épül fel – a szivárgási áramokból adódó vesztesége is alacsony.

Elhanyagolva azokat a parazitákat, amelyeket a kisütő áramút tranzisztorai okoznak, azt lehet mondani, hogy logikai kapu teljesen adiabatikus kisütése a (2.10) összefüggéssel megadott nem-adiabatikus veszteséget küszöböli ki. Ugyanakkor a többletként beszúrt inverter szintén rendelkezik a (2.10) összefüggéssel megadott nem-adiabatikus veszteséggel, valamint a (2.11) képlettel közelített adiabatikus és a szivárgási áramokból adódó veszteséggel is. Ezeket az összefüggéseket a (3.1) szerint arányba állítva látható, hogy az energiaveszteségek viszonya a C_l és a C_i kapacitások arányától, illetve a működési frekvenciától függ, ahol a C_l a logikai kapu, míg a C_i az inverter terhelőkapacitása.

$$\frac{\frac{1}{2}C_l U_{t,p}^2}{\frac{1}{2}C_i U_{t,p}^2 + C_i \frac{U_{dd}}{T} + I_0} \tag{3.1}$$

Mivel a (3.1) arány csak közelítés, és a benne szereplő változók egy részének értékei is ismeretlenek, ezért a kérdés megválaszolására szimulációt végeztem. A szimulációval nyert adatok alapján a *3.9. ábrán* a nem teljesen adiabatikusan kisütött 100 tagú inverter lánc, valamint a teljesen adiabatikusan kisütött 100 tagú inverter lánc és a 100 kisütő inverter együttes energiaveszteségét ábrázoltam különböző működési frekvenciákon az inverter lánc egyes invertereinek kisütő inverteren felüli terhelőkapacitásának függvényében.

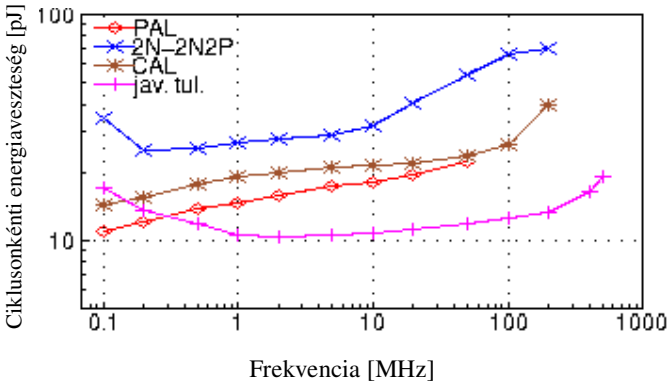


3.9. ábra. 100 nem teljesen adiabatikusan kisütött, valamint 100 teljesen adiabatikusan kisütött inverter és a 100 kisütő inverter együttes ciklusonkénti energia-vesztesége 1, 10 és 100 MHz működési frekvenciákon az inverter lánc kisütő invertereken felüli terhelőkapacitásának függvényében

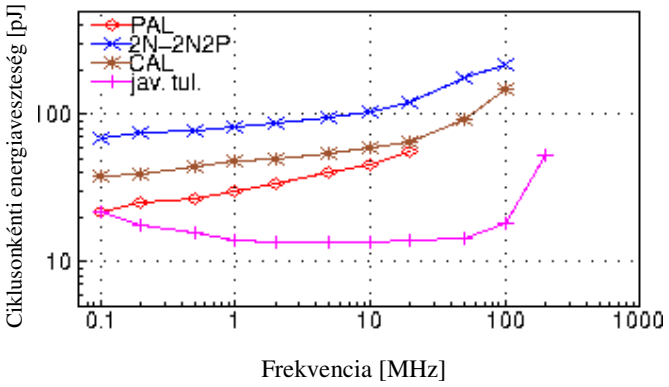
A *3.9. ábráról* leolvasható, hogy a kisütő inverter alkalmazása már alacsony terhelő kapacitás és kis működési frekvencia együttese esetén is energiamegtakarítást eredményez, amely energiamegtakarítás akár a terhelő kapacitás, akár a működési frekvencia növelésével egyre jelentősebbé válik.

E. További szimulációs eredmények

A *3.10.* és a *3.11. ábrán* a teljesen adiabatikusan működő javított tulajdonságú adiabatikus kapcsolást hasonlítottam össze működési frekvencia és ciklusonkénti energiaveszteség szempontjából a gyors működésre képes a *2N-2N2P* és *CAL*, valamint az energia-hatékony, teljesen adiabatikus működésű *PAL* kapcsolásokkal 20 fF illetve 100 fF terelő kapacitások mellett.



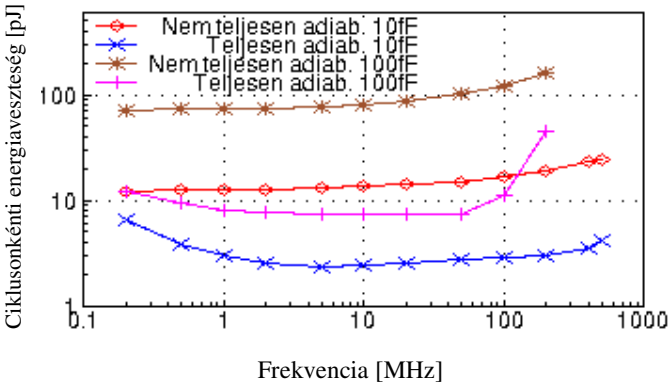
3.10. ábra. 100 láncba kapcsolt *PAL*, *2N-2N2P* és *CAL* inverter, valamint a teljesen adiabatikusan működő javított tulajdonságú adiabatikus kapcsolás ciklusonkénti energiavesztesége 20 fF terhelőkapacitás mellett a frekvencia függvényében



3.11. ábra. 100 láncba kapcsolt *PAL*, *2N-2N2P* és *CAL* inverter, valamint a teljesen adiabatikusan működő javított tulajdonságú adiabatikus kapcsolás ciklusonkénti energiavesztesége 100 fF terhelőkapacitás mellett a frekvencia függvényében

Az *3.10.* és a *3.11. ábrákról* leolvasható, hogy – kis terhelő kapacitás és alacsony működési frekvencától eltekintve – a teljesen adiabatikusan működő javított tulajdonságú adiabatikus kapcsolás mind működési határfrekvencia, mint ciklusonkénti energiaveszteség szepontjából felülmúlja az irodalomban található leghatékonyabb adiabatikus kapcsolásokat.

A *3.12. ábrán* a javított tulajdonságú adiabatikus kapcsolást hasonlítottam a következő fokozatban meglévő inverter felhasználásával teljesen adiabatikus működésűvé tett javított tulajdonságú adiabatikus kapcsoláshoz. Az ábráról leolvasható, hogy a következő fokozatban meglévő inverter felhasználásával az energai-hatékonyság – terhelő kapacitástól függően – akár egy nagyságrenddel is javítható.



3.12. ábra. A javított tulajdonságú adiabatikus kapcsolás, valamint a következő fokozatban elhelyezkedő inverter felhasználásával teljesen adiabatikus működésűvé tett javított tulajdonságú adiabatikus kapcsolás energiaveszteségе 100 tagú inverter láncon 10 fF és 100 fF terhelőkapacitás mellett a frekvencia függvényében

3.1.2. Adiabatikus elven működő tárolóelem kialakítása

Adiabatikus elven működő kapcsolással statikus tárolóelem kialakítása nem lehetséges az adiabatikus töltés dinamikus jellege miatt. Az adattárolás csak dinamikusan valósulhat meg, mivel a vezérlő fázisjel egyben a kapcsolás tápvezetékét is jelenti, így ennek alacsony szintje alatt az áramkör nem táplált. A szakirodalomban található adiabatikus tárolóelemek a logikai kapuktól elviekben is különböző felépítésűek, így be- és kimeneti jelalakjaik nem feltétlenül egyeznek meg a logikai kapuk hasonló jelalakjaival, vagy működésük egy részében statikus jeleket is igényelnek.

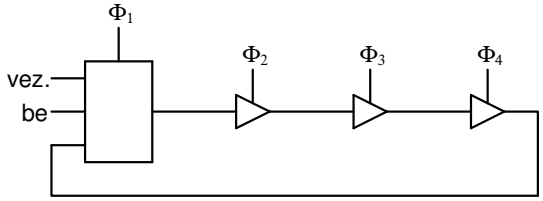
A reverzibilis működési elvből adódóan az adiabatikus elven történő adattárolás mindig – ha csak nagyon pici is, de – vissza nem forgatható energiaveszteség árán valósítható meg [50].

A. A kidolgozott adiabatikus elven működő tárolóelem

Adattárolás megvalósítható úgy is, hogy a tárolandó adatot nem a hagyományos értelemben, fizikailag ugyanazon a helyen „tároljuk”, hanem folyamatosan, újra és újra lemásoljuk. Mivel egy adat lemásolása tetszőlegesen kis energiaveszteség árán megvalósítható, így hatékony adiabatikus tárolóelem készíthető. Az adiabatikus elven működő tárolóelem kidolgozásakor ezt a módszert alkalmaztam.

Az adatok lemásolásához invertereket használhatunk. Ez alapján egy végtelen hosszú inverter lánc végtelen idejű adattárolásra képes. Gyakorlati okokból azonban a végtelen hosszú lánc nem megengedhető, a szükséges építőelemek számát csökkenteni kell. A javított tulajdonságú adiabatikus kapcsolás négyfázisú működésű, ahol egy logikai kapu kimenete a bemenetéhez képest negyed periódussal késik, ezáltal a negyedik inverter kimenete fázishelyesen visszavezethető az első inverter bemenetére. Az invertereket ily módon gyűrűbe fűzve az adattárolás a tárolt adat folyamatos másolásával megoldható.

A fentiek alapján kidolgozott adiabatikus elven működő tároló felépítése a *3.13. ábra* szerinti [S26], ahol a Φ_1 fázisjelhez kapcsolódó logikai kapu egy összetett kapu, míg a többi három kapu csak inverter. Az inverterek pufferként működnek, mivel azonban az adiabatikus kapcsolás mind a ponált, mind a negált logikai értéket előállítja, inverter és a puffer között nincs különbség, csak a kimeneteiket cseréljük fel.



3.13. ábra. A kidolgozott adiabatikus elven működő tárolóelem felépítése

Az összetett logikai kapu – ami tulajdonképpen egy kétbemenetű multiplexer – szolgál az új adat beírására, ami a *3.13. ábra* szerinti elrendezésben a felfutó Φ_1 fázisjelre történhet meg. Bekapcsoláskor célszerű a tárolónak új bemeneti értéket szolgáltatni, hogy biztosan a megfelelő logikai állapotba billenjen, azonban az áramköri elrendezés okozta nem tökéletes szimmetria következtében egy átmeneti állapot után enélkül is beáll valamelyik logikai állapotba.

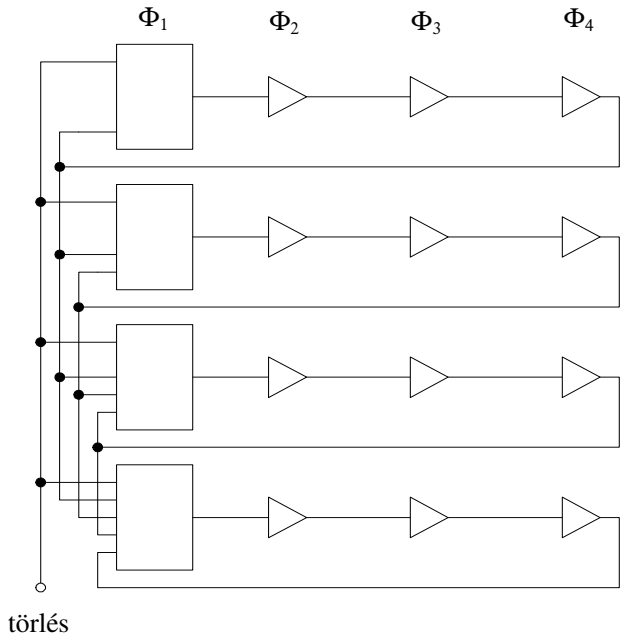
A kidolgozott tárolóelem működésének szemléltetésére tegyük fel, hogy a Φ_1 fázisjel felfutásakor a multiplexer olyan vezérlést kap, hogy a bemenő adat számára legyen átlátszó. Amikor a Φ_1 fázisjel eléri a legmagasabb szintjét, akkor a bemenő adat másolata megjelenik a multiplexer kimenetén. Ugyanezt az adatot másolja az első inverter a Φ_2 fázisjel felfutásakor, majd a második, illetve a harmadik inverter a Φ_3, illetve a Φ_4 fázisjelek felfutásakor. A Φ_1 fázisjel következő felfutásakor az előzőleg beírt adat másolata megjelenik a multiplexer megfelelő adatbemenetén. Ha a multiplexer most ellenkező vezérlést kap, és így a tárolt adat számára átlátszó, akkor ennek a másolata a kapuk között ismételten körbejár, egészen addig, amíg egy új adat felül nem írja.

Az adat tárolása tehát dinamikus, a kapuk a fázisjelek által megszabott ütemben a beírt adatot folyamatosan másolva egymásnak adják körbe. A tárolt adat bármelyik kapu kimenetéről levehető, de mindig csak az adott kapu helyzetének megfelelő fázisjel magas szintje alatt. A tároló gyakorlati megvalósításában minden olyan logikai kapu teljesen adiabatikus működésűvé tehető, amelyet inverter követ. Eszerint nem-adiabatikus vesztesége csak a negyedik inverternek van, azonban az előző alfejezetben bemutatott megoldással ez is kiküszöbölhető, ha ezt annak kapacitív terhelése indokolttá teszi.

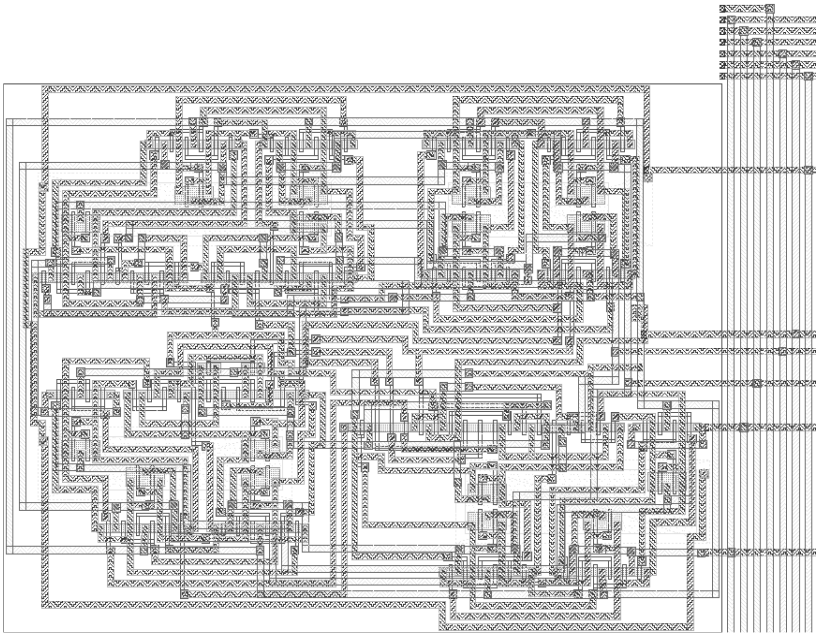
A kidolgozott, logikai kapukból felépített tárolóelem előnye, hogy – mivel csak logikai kapukból épül fel – mind a vezérléséhez szükséges jelalakok, mind a tároló kimeneti jelalakjai megegyeznek a logikai kapukéval. Ezáltal a kidolgozott javított tulajdonságú adiabatikus logika alkalmazásával kis fogyasztású kombinációs hálózatok mellett kis fogyasztású sorrendi hálózatok is megvalósíthatók anélkül, hogy statikus tárolóelemekre szükség lenne.

B. Számláló megvalósítása adiabatikus elven működő tárolóelemekkel

A kidolgozott adiabatikus elven működő tárolóelem gyakorlati alkalmazhatóságának szemléltetésére, annak felhasználásával egy adiabatikus elven működő négy-bites bináris számlálót terveztem, amelynek elvi kapcsolása a *3.14.*, szilícium rajzolata pedig a *3.15. ábrán* látható. Az elvi kapcsoláson látható, hogy minden egyes bit tárolására egy tárolóelem szolgál, azonban az első bit kivételével a tárolóelemekben a két-bemenetű multiplexer helyett egy-egy összetettebb logikai függvényt megvalósító kapu szerepel, konkrétan több bemenetű *kizáró-vagy* kapu, a számláló működésének megfelelően.



3.14. ábra. A kidolgozott adiabatikus elven működő négy-bites bináris számláló elvi kapcsolása



3.15. ábra. A kidolgozott adiabatikus elven működő négy-bites bináris számláló szilícium-rajzolata.

Az áramkör szilícium-rajzolatának elkészítése alapján lehetővé vált a javított tulajdonságú adiabatikus kapcsolás és a statikus CMOS kapcsolás által elfoglalt szilíciumterületek összevetése. Az adiabatikus számláló és az ugyanerre a technológiára standard cellás tervezőrendszerrel elkészített négy-bites bináris statikus számláló által elfoglalt szilíciumterületeket a *3.1. táblázat* tartalmazza. Látható az adatokból, hogy a javított tulajdonságú adiabatikus kapcsolás szilíciumterület-igénye a statikus kapcsoláshoz képest körülbelül másfélszeres.

	Magasság [μm]	*Szélesség [μm]*	*Terület [$\mu m * \mu m$]*
Adiabatikus	122,3	165,6	20240,65
Statikus	118,2	116,6	13782,12

3.1. táblázat. A kidolgozott adiabatikus elven működő négy-bites bináris számláló és a statikus megfelelője által elfoglalt szilíciumterületek összevetése

A nagyobb területigény oka a megduplázott logika és az ezzel együtt járó megduplázott vezetékezés, hiszen az adiabatikus kapcsolásban mind a ponált, mind a negált logika megvalósítása szükséges. Azonban a területigény kevesebb, mint kétszeres, amivel a javított tulajdonságú adiabatikus kapcsolás logikai kapunként csak két PMOS tranzisztort igényel, és a logikai függvényt megvalósító NMOS tranzisztorok a technológia adta legkisebb méreten tarthatók.

3.2. Adiabatikus kapcsolások optimálási lehetőségei energiaveszteség szempontjából

A gyakorlati áramkörökben a kapcsolóelemeket tranzisztorokkal valósítjuk meg, amelyek nem ideális volta rontja az energiaveszteségekre kapott elméleti értékeket. Az energiaveszteség csökkentésére ezért különböző megoldásokat alkalmazhatunk. Adiabatikus kapcsolások esetén is szóba jöhetnek az olyan – a statikus CMOS áramkörök kapcsán már jól ismert – módszerek, mint a tápfeszültség csökkentés, a terhelőkapacitás mérséklése, ezenkívül – egy bizonyos határig – a működési frekvencia lecsökkentése.

Az adiabatikus kapcsolások alkalmazása ezen technikáknak nem helyettesítője, hanem kiegészítője. Tehát például egy alacsonyabb tápfeszültségről működtetett – és így kedvezőbb teljesítményfelvételű – statikus CMOS áramkört adiabatikus kapcsolással helyettesítve további energiamegtakarítás érhető el. Hasonlóképpen, egy adiabatikus áramkört kisebb feszültséggel táplálva az energia-hatékonysága nő. A kisebb feszültséggel való táplálás adiabatikus áramköröknél a tápláló fázisjel amplitúdójának csökkentését jelenti. Azonban ennek – akárcsak a statikus CMOS áramkörök esetén – elsősorban a velejáró küszöbfeszültség-csökkentés, és az így megnövekedő küszöbalatti áram szab határt. Megjegyzendő, hogy a tranzisztor kikapcsolt állapotában is folyó áram értéke a statikus és az adiabatikus áramkörökben ugyan összemérhető, az így adódó teljesítményfelvétel azonban adiabatikus áramkörökben mégis kisebb értékű, mivel ez az áram csak a fázisjel tartási periódusa alatt folyik; ami, például négyfázisú adiabatikus rendszerben a teljes periódus idő negyede.

Alacsonyabb küszöbfeszültség alkalmazása a kvázi-adiabatikusan működő áramkörök esetén előnyösnek látszik abból a szempontból, hogy mérsékli a kapcsolás nem-adiabatikus veszteségét. Ugyanakkor a folyamatos eszközméret- és küszöbfeszültség csökkenéssel járó szivárgási és küszöbalatti áramok exponenciális növekedése egyre inkább a figyelem középpontjába kerül. Ezen áramok hatása akár olyannyira számottevővé válhat, hogy a belőlük adódó teljesítményfelvétel már túlszárnyalja az alacsonyabb küszöbfeszültséggel elérhető nem-adiabatikus teljesítményveszteség csökkenést. Ilyen esetben tehát nem érdemes túlzottan alacsony küszöbfeszültségre törekedni, sokkal inkább kifizetődő a szivárgási és a küszöbalatti áramok minimalizálása mellett a nem-adiabatikus veszteséget egyéb módon, például kapcsolási szinten megszüntetni. Az előző alfejezeben bemutatott – nem-adiabatikus veszteség kiküszöbölésére szolgáló – megoldások (kettős feltöltő áramút, új kisütő áramút) mind mind ezt a célt szolgálják.

Mindemellett az eszközméret csökkentése a teljesítményfelvétel csökkenését vonja maga után. Minél kisebb méretű ugyanis egy tranzisztor, annál kisebb mértékű parazita hatásokkal rendelkezik, ideértve a *gate* kapacitást is. Kisebb eszközmérettel tehát elérhető az egyes logikai kapukat terhelő kapacitás mérséklése, ami által a teljesítményfelvétel csökkenthető és/vagy a működési frekvencia növelhető.

Nagy működési frekvencia esetén, és amennyiben a tápfeszültség (a tápláló fázisjel amplitúdója) valamint a gyártási technológia (és ezáltal az egységnyi felületű tranziszor *gate* kapacitása) adott, akkor az energiaveszteség további csökkentése a

feltöltő/kisütő áramút ellenállásának csökkentésével lehetséges. A feltöltő/kisütő áramút ellenállásának csökkentése az adiabatikus veszteség mérséklését veszi célba.

Egy MOS tranzisztor vezetési csatornájának *W* szélessége, valamint ehhez kapcsolódóan a bekapcsolási ellenállása megszabja a rajta átfolyó áram értékét, ezáltal a lehetséges töltési időt. Egy túl keskeny tranzisztoron átfolyó nagy áram ugyanis egyrészt nagy adiabatikus veszteséget eredményez, másrészt alsó korlátot ad a töltési időre, és így felső korlátot a működési sebességre. Minél kisebb adiabatikus veszteséget, illetve minél nagyobb működési sebességet szeretnénk elérni, annál kisebb ellenállású – vagyis nagyobb *W* csatorna szélességű tranzisztorokból felépített – feltöltő/kisütő áramútra van szükség.

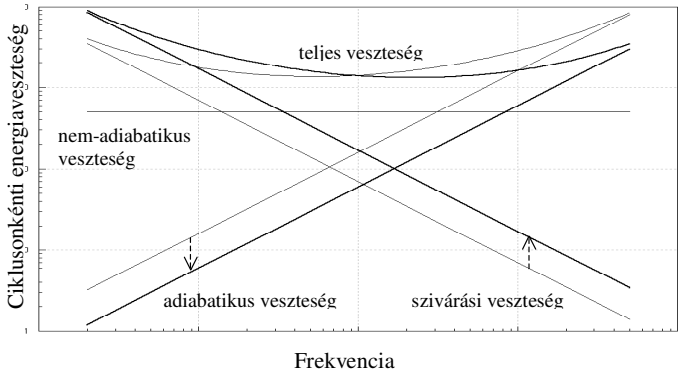
3.2.1. Adiabatikus kapcsolások energiaveszteségének csökkentése a tranzisztorok vezetési csatornaszélességének optimálásával

Láthattuk tehát, hogy lehetőség van az adiabatikus áramkörök – a 2.1.2 fejezetben felsorolt – valamennyi teljesítményfelvételi forrásából adódó veszteségének csökkentésére. A szivárgási és a küszöbalatti áramok magasabb küszöbfeszültséggel mérsékelhetők, míg a nem-adiabatikus veszteség például kisütő áramúttal kiegészített javított tulajdonságú adiabatikus kapcsolással küszöbölhető ki. Végül pedig az adiabatikus veszteség a feltöltő/kisütő áramút ellenállásának csökkentésével mérsékelhető.

A valóságban azonban a helyzet nem ennyire egyértelmű. Nem lehet ugyanis az egyes teljesítményfelvételi forrásokból adódó veszteségek egymástól független optimálásával kellő energia-hatékonyságot elérni. Csak az egyik veszteségtípus önmagában való lecsökkentése, vagy esetleg teljes megszüntetése kedvezőtlenül befolyásolhatja más teljesítményfelvételi forrásokból adódó veszteségek értékeit, így az áramkör eredő teljesítményfelvétele adott esetben nemhogy nem csökken, hanem még nőhet is. Továbbá nem szabad figyelmen kívül hagyni a teljesítményfelvétel optimálásának az elérhető működési frekvenciára gyakorolt hatását sem.

A tervezés azon szakaszában, amikor a működési frekvencia, a tápláló fázisjel amplitúdója és a gyártási technológia már meghatározott, a tervezőnek még lehetősége van feltöltő/kisütő áramútban elhelyezkedő tranzisztorok vezetési csatornáinak *W* szélességének megválasztására.

Amennyiben egy tranzisztor vezetési csatornaszélességét növeljük, akkor bekapcsolási ellenállása csökken. A kisebb bekapcsolási ellenállás kisebb adiabatikus veszteséget eredményez, amit szemléletesen úgy képzelhetünk el, hogy a *2.6. ábra* „adiabatikus veszteség” görbéje lefelé tólódik el. A szélesebb vezetési csatorna ugyanakkor megnöveli a tranzisztor szivárgási és küszöbalatti áramait, és így az ezekből adódó veszteségét, amit a „szivárgási veszteség” görbe felfelé tólódása jelez. Mindezekből következik, hogy a vezetési csatornaszélesség növelésével a teljes energiaveszteség még az adiabatikus veszteség maradéktalan kiküszöbölése mellett sem csökkenthető egy legkisebb érték alá. A vezetési csatornaszélesség növelésének hatására az adiabatikus- és a szivárgási veszteséggörbék eltolódását, valamint ezáltal a teljes energiaveszteség változását a *3.16. ábra* szemlélteti.



3.*16. ábra.* A tranzisztor vezetési csatornaszélesség növelésének hatása az egyes energiaveszteség típusokra

Az energiaveszteség - működési frekvencia görbék eltolódása azt eredményezi, hogy a vezetési csatornaszélességet növelve a legkisebb teljes energiaveszteség a magasabb működési frekvenciák felé mozdul el. Ugyanez az elgondolás fordítva is igaz. Amennyiben a működési frekvenciát a tervezés korábbi szakaszában már rögzítették, a vezetési csatornaszélességet az adott működési frekvenciához tartozó legkisebb teljes energiaveszteséghez lehet igazítani.

Egy adott működési frekvencián – amennyiben ez a működési frekvencia az adiabatikus veszteség által meghatározott frekvenciatartományon belül van – azt tapasztaljuk, hogy növelve a vezetési csatornaszélességet, az energiaveszteség először csökken, majd ismét növekszik, mivel a tranzisztor túl széles vezetési csatornájának nagyobb szivárgási és küszöbalatti áramai, valamint a megnövekedett parazita kapacitásainak töltésére és kisütésére fordított energiatöbblet felülmúlja a lecsökkent bekapcsolási ellenállásból adódó energiamegtakarítást [S18].

Ennek értelmében tehát a legkisebb energiaveszteség egy olyan vezetési csatornaszélesség mellett adódik, amelynél a bekapcsolási ellenállásból adódó energiaveszteség valamint a tranzisztor szivárgási és küszöbalatti áramainak vesztesége és a parazita kapacitásainak töltésére és kisütésére fordított energia összege a legkisebb.

3.2.2. A javított tulajdonságú adiabatikus kapcsolás tranzisztorainak méretoptimálása

Az adiabatikus kapcsolásokban általában csak a terhelőkapacitást töltő és kisütő áramútban lévő tranzisztorok vezetési csatornaszélességének méretoptimálása szükséges, mivel egy logikai kapu belső pontjainak kapacitása többnyire olyan kismértékű, hogy a töltéséhez és kisütéséhez szükséges optimális tranzisztorméret a technológiai legkisebb méret alatt lenne.

A javított tulajdonságú adiabatikus kapcsolás logikai kapujában a kimeneti kapacitást töltő áram két úton folyik. Az egyik a PMOS tranzisztor, a másik pedig a logikai függvényt megvalósító NMOS tranzisztorhálózat. Az NMOS tranzisztorhálózat a megvalósítandó logikai függvénynek megfelelően sorosan és/vagy párhuzamosan

kapcsolt NMOS tranzisztorokból áll, ahol az egyes tranzisztorok az éppen érvényes bemeneti logikai kombinációnak megfelelően nyitottak vagy zártak. Ezáltal a tranzisztorhálózaton belül akár több párhuzamos áramút is lehetséges, de ugyanígy létrejöhet több tranzisztoron sorosan átvezető egyetlen áramút is. Mindebből következik, hogy az NMOS tranzisztorhálózat ellenállása az adiabatikus veszteség csökkentése szempontjából lehet igen kedvező, ugyanakkor meglehetősen kedvezőtlen eset is előfordulhat. Ráadásul így a feltöltő áramút ellenállása nemcsak attól függ, hogy az adott logikai kapu milyen logikai függvényt valósít meg, hanem függ még az éppen érvényes bemeneti logikai kombinációtól is.

A kimeneti kapacitást kisütő áram, amennyiben az adott logikai kapu rendelkezik kisütő áramúttal, szintén két úton folyik, amelynek egyike a kisütő áramút NMOS tranzisztora, a másik pedig a feltöltésben is részt vevő PMOS tranzisztor. Kisütő áramút hiányában a kisütést egyedül a PMOS tranzisztor végzi.

Mindebből látszik, hogy a PMOS tranzisztor bekapcsolási ellenállása mind a feltöltő, mind a kisütő áramút ellenállását nagyban meghatározza. Ezért a bekapcsolási ellenállás csökkentéséhez szükséges, szilíciumterület-igény növekedéssel járó tranzisztorméret növelés leginkább akkor térül meg, ha az áramút ellenállásának csökkentéséhez a PMOS tranzisztor vezetési csatornáját választjuk szélesebbre. Az optimálást az NMOS tranzisztorhálózattal kezdeni márcsak azért sem érdemes, mert így egyrészt több tranzisztor szivárási veszteségét növelnénk, másrészt ezen tranzisztorok *gate*-kapacitásai egyben az előző logikai fokozat terhelőkapacitásai is, továbbá a kisütő áramút ellenállását az NMOS tranzisztorhálózat optimálása érintetlenül hagyja, annak javításáról külön kellene gondoskodni.

A PMOS tranzisztor vezetési csatorna szélességének optimálása ezzel szemben az áramútban csupán egyetlen tranzisztor szivárási veszteségét érinti, valamint segítsével mind a feltöltő, mind a kisütő áramút ellenállása anélkül csökkenthető, hogy az lényeges hatással lenne az előző logikai fokozat által látott kapacitív terhelésre. Közvetlen hatással van ugyanakkor a PMOS tranzisztor mérete az adott logikai kapu saját belső, parazita kapacitására.

A vezetékezés kapacitása és a következő logikai fokozat által mutatott bemeneti kapacitás, összességében tehát a C_L terhelőkapacitáson túl az adott logikai kapunak a saját belső, parazita kapacitásait is fel kell töltenie, illetve ki kell sütnie működése során. A belső, parazita kapacitást a C_g *gate* kapacitások, a C_{gs} *gate-source* kapacitások és a C_{gd} *gate-drain* kapacitások adják, amelyek közvetlenül arányosak az egyes tranziszorok vezetési csatornáinak fizikai méretével.

Feltételezve, hogy az adiabatikus töltési feltétel teljesül, és így az U_{ds} *drain-source* feszültség kicsi, ekkor a tranzisztorok bekapcsolási ellenállását a (2.6) összefüggésekkel közelítettük. Bevezetve a C_{ox} egységnyi oxid-kapacitást, a korábbi összefüggést a PMOS tranzisztorra a (3.2) képlet szerint írhatjuk át, ahol K_p az adott technológiára jellemző állandó, W_p a vezetési csatorna szélessége, L a hosszúsága, U_{ds} a *drain-source* feszültség és U_t a küszöbfeszültség.

$$R \approx \frac{K_p}{\frac{W_p}{L} C_{ox} (U_{ds} - U_t)} \tag{3.2}$$

A (3.3) képlet a W_P vezetési csatorna szélességű PMOS tranzisztor adiabatikus energiaveszteségét adja a *3.1. ábrán* látható inverter kisütési folyamatára, amikor is csak a PMOS tranzisztor vezet. Feltételezve, hogy a C_2 terhelőkapacitás volt feltöltve, a T_2 tranzisztor a terhelőkapacitáson kívül kisüti a T_1 tranzisztor *gate* kapacitását, továbbá a saját *drain*, a T_4 tranzisztor *source*, valamint a T_6 tranzisztor *drain* kapacitásait is.

$$E_{\text{adiabatikus veszteség}} = \frac{K_p \cdot L \cdot (C_{g(T_1)} + C_{gd(T_2)} + C_{gs(T_4)} + C_{gd(T_6)} + C_2)^2 \cdot U_{dd}^2}{W_p \cdot C_{ox} \cdot (U_{ds} - U_t) \cdot T} \tag{3.3}$$

Kis *drain-source* feszültség mellett a *gate* kapacitás megközelítőleg duplája *source* illetve a *drain* kapacitásoknak, tehát $C_{gs} \approx C_{gd} \approx 0.5C_g$ írható. A tranzisztorok között egyenlő *L* vezetési csatornahosszúságot feltételezve az egyes kapacitás értékek a csatornaszélességgel arányosak, így a *gate* kapacitást eggyel súlyozva, az egyes kapacitás értékeket összegezve, a következő eredő csatornaszélességet kapjuk: $W_{\text{eredő}} = W_p + 0.5W_p + 0.5W_n + 0.5W_n$, ahol W_n az NMOS tranzisztorok csatornaszélességét jelöli. Az egységnyi csatornaszélesség *gate* kapacitását C_G-vel jelölve a (3.3) képletet a (3.4) szerinti formába írhatjuk át [S37].

$$E_{\text{adiabatikus veszteség}} = \frac{K_p \cdot L \cdot ((1.5W_p + W_n) \cdot C_G + C_2)^2 \cdot U_{dd}^2}{W_p \cdot C_{ox} \cdot (U_{ds} - U_t) \cdot T} \tag{3.4}$$

Ebből a PMOS tranzisztor legkisebb adiabatikus energiaveszteséget eredményező $W_{p,\ optimális}$ szélességét a (3.5) összefüggés adja.

$$W_{p,\ \text{optimális}} = \frac{2}{3} W_n + \frac{2}{3} \cdot \frac{C_2}{C_G} \tag{3.5}$$

Összetett logikai kapu esetén a legkisebb adiabatikus veszteséget adó PMOS tranzisztorszélességet hasonló gondolatmenet alapján kaphatjuk meg. Ilyenkor a kisütött kapacitások összegzésekor az NMOS tranzisztorhálózatban lévő valamennyi párhuzamos, a kimenethez csatlakozó *gate-source* kapacitást figyelembe kell venni. Tekintettel arra, hogy a kisütés fázisban az NMOS tranzisztorhálózat tranzisztorai zártak, az NMOS hálózat közbülső parazita kapacitásai nem sülnek ki a PMOS tranzisztoron keresztül. A vázolt összefüggés azonban a PMOS tranzisztor optimális szélességére csak közelítő értéket ad, továbbá nem veszi figyelembe az optimális vezetési csatorna szélesség működési frekvenciától való függését sem. Közelítés helyett, az adott működési frekvenciához tartozó optimális csatornaszélességet szimulációval határozhatjuk meg [S18].

A *3.2. táblázatban* a kidolgozott adiabatikus elven működő négy-bites bináris számláló ciklusonkénti energiavesztesége látható a működési frekvencia függvényében. A technológia adta legkisebb méretű tranzisztorokkal felépített kapcsolás energiavesztesége a „Legkisebb" oszlopban található. Az „Optimális" oszlopban a tranzisztorméreteket szimuláció alapján úgy állítottam be, hogy a számláló az adott működési frekvencián a legkisebb energiaveszteséget nyújtsa. Az utolsó oszloban közölt eredményeket kisütő áramút alkalmazásával kaptam, ahol a számlálóban egyébként is meglévő invertereket használtam fel az előző fokozatban elhelyezkedő logikai kapu teljesen adiabatikus működésűvé tételére.

Frekvencia	Energiaveszteség/ciklus [pJ]		
[MHz]	Legkisebb	Optimális	Teljesen adiabatikus
10	2.29	2.29	0.87
50	2.77	2.77	1.06
100	3.42	3.20	1.26
200	4.42	3.56	1.67
400	5.73	4.47	2.79
500		5.21	3.67

3.2. táblázat. A kidolgozott adiabatikus elven működő négy-bites bináris számláló egy számlálási ciklusának energiaveszteségе a működési frekvencia függvényében

Általánosságban elmondható, hogy a legkisebb energiaveszteségre – a *3.16. ábrán* bemutatott elméleti megfontolásnak megfelelően – egy jól meghatározható tranzisztorméret található, amely függvénye a terhelőkapacitásnak és a működési frekvenciának [S18]. A *3.2. táblázatból* ugyanakkor kiolvasható, hogy egy bizonyos működési frekvencia (illetve terhelőkapacitás) alatt ilyen optimum nem kapható, hanem az energiaveszteség közel lineáris függvénye a működési frekvenciának (illetve a terhelőkapacitásnak). Ennek az a magyarázata, hogy az optimális tranzisztorméret a technológiai legkisebb méret alatt lenne. A *3.2. táblázatban* az is látszik, hogy a tranzisztorméret megfelelő beállításával a működési frekvencia növelésére is lehetőség nyílik.

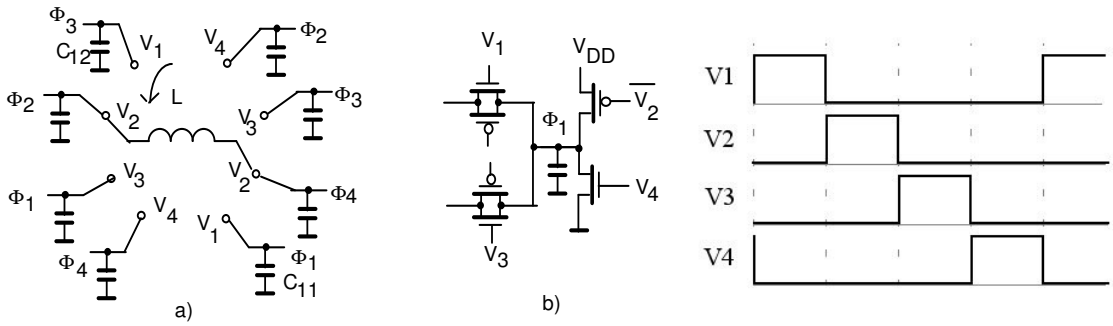
3.2.3. Adiabatikus logika és fázisjel generátor együttes optimálása

Többfázisú adiabatikus rendszerek gyakorlati megvalósításánál előállhat az a probléma, hogy az áramkörnek a generátor felé mutatott kapacitív terhelése fázisonként eltérő. Mivel a tápláló fázisvezetékek hálózata a rezonáns tápegység része, ezért az eltérő kapacitást mutató fázisvezetékeknek az induktivitáshoz való kapcsolásával a generátor működési frekvenciája megváltozik.

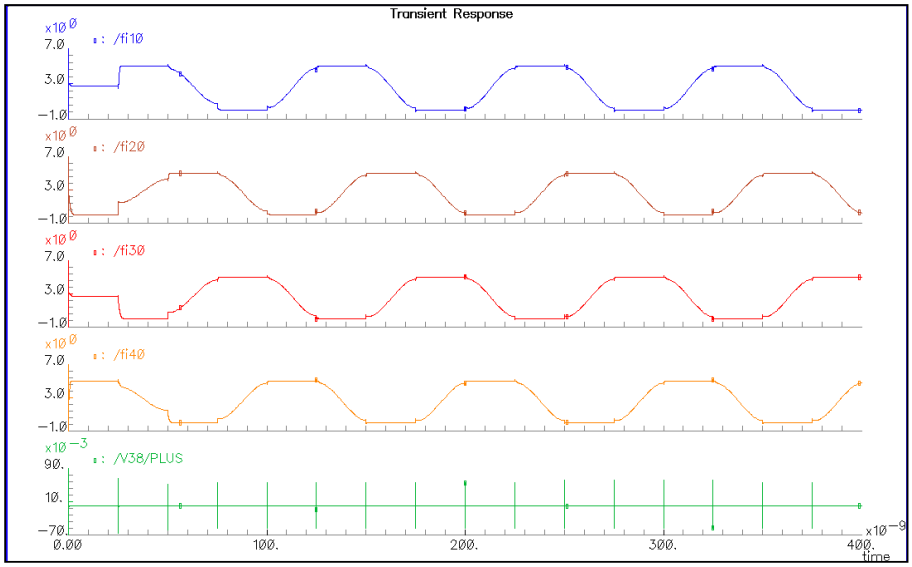
A fázisvezeték kapacitása egy adatfüggő és egy adatfüggetlen részre osztható. Az adatfüggő kapacitás abból adódik, hogy az adott fázisvezetékhez csatlakozó logikai kapuk kapacitív terhelése eltérő lehet a kapuk igaz vagy hamis logikai állapotától függően, ami elsősorban a nem feltétlenül szimmetrikus vezetékezésnek tudható be. A fázisvezeték adatfüggetlen kapacitása maga az elosztó hálózat kapacitása, amihez hozzáadódik a rá csatlakozó – fázisvezetékenként esetleg eltérő darabszámú – logikai kapu terhelőkapacitása. A gyakorlatban általában az elosztó hálózat kapacitása összemérhető az adott fázisvezetékhez csatlakozó logikai kapuk terhelőkapacitásával, valamint az adatfüggő kapacitás csak igen kismértékű, így összességében ennek hatása jelentéktelen.

A fázisonként eltérő kapacitív terhelés problémája megszüntethető több fázisjel-generátor alkalmazásával, ahol az egyes generátorok egymástól függetlenül külön-külön az adott kapacitáshoz hangolhatók. Azonban a teljes rendszer energiahatékonysága, és az integrált áramköri kivezetések számának minimalizálása szempontjából előnyösebb csupán egyetlen generátor használata [S31][S37].

Kutatócsapatommal számos adiabatikus generátorkapcsolást megvizsgáltunk, az egyes generátorok részletes összehasonlítását publikáltuk [S21]. Négyfázisú áramkörökhöz az általunk gyakorlati megvalósításra legjobbnak ítélt úgynevezett forgótekercses elrendezés a *3.17. ábrán* látható. A generátor szimulációval nyert jelalakjai a *3.18. ábrán* láthatóak.



3.17. ábra. a) Forgótekercses elrendezésű négyfázisú adiabatikus fázisjel generátor elrendezése. b) A fázisjelek bekötése és statikus vezérlése

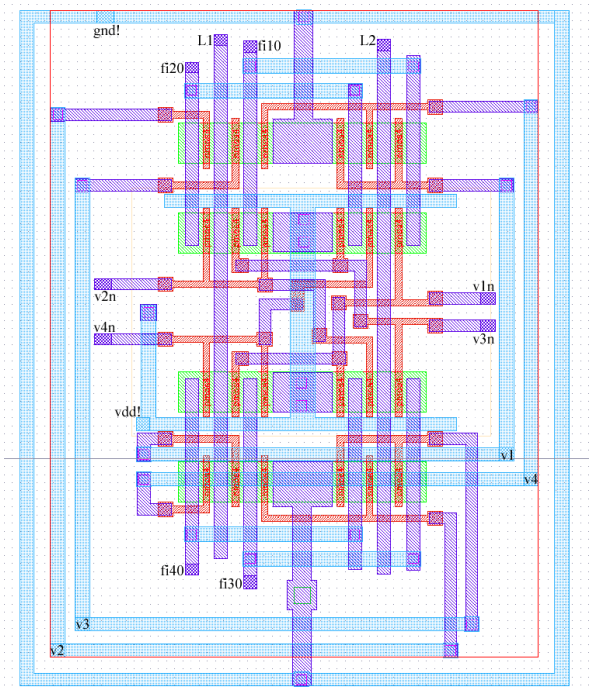


3.18. ábra. Forgótekercses elrendezésű négyfázisú adiabatikus fázisjel generátor jelalakjai

Többfázisú adiabatikus rendszerekben egyetlen generátor használata esetén – illetve minden olyan esetben, amikor egy generátor több fázisvezetéket is meghajt – az egyes fázisjelek közel egyenlő kapacitív terhelést kell, hogy mutassanak. Ezt a gyakorlatban úgy érik el, hogy a kisebb kapacitív terhelést mutató fázisvezetékek kapacitív terhelését külső, állítható kapacitással megnövelik [57].

Előnyösebb azonban – amennyiben a fázisvezetékek közötti kapacitáskülönbség már a tervezéskor meghatározható – a különbségek kiegyenlítéséhez a kis kapacitást mutató fázisvezetékekhez csatlakozó logikai kapukban az áramutakban lévő tranzisztorokat a csak a kapu energiaveszteségéből adódó optimumnál nagyobb méretűre választani [S31].

A *3.19. ábrán* a forgótekercses elrendezésű négyfázisú adiabatikus fázisjel generátor szilícium-rajzolata látható.



3.19. ábra. A forgótekercses elrendezésű négyfázisú adiabatikus fázisjel generátor szilícium-rajzolata

A fázisjel generátor és a *3.15. ábrán* bemutatott adiabatikus elven működő négy-bites bináris számláló rajzolattervének együttes ciklusonkénti energiaveszteségét a *3.3. táblázat* foglalja össze.

Önmagában az adott működési frekvenciára optimalizált számláló a fázisjel generátorral összekapcsolva nem nyújtja a legkedvezőbb ciklusonkénti energiaveszteséget. Ennek eléréshez a teljes rendszer szempontjából optimális tranzisztorméretekre van szükség, amely figyelembe veszi az egyes fázisjelek eltérő kapacitív terhelését. A közölt példában a fázisvezetékek közötti kapacitáskülönbségek tranzisztorméret-növeléssel való kiegyenlítésével további 16%-os energiamegtakarításra volt lehetőség.

	Energiafelvétel [µW]
Optimális számláló	71,56
Optimális rendszer	60,62

3.3. táblázat. A négy-bites adiabatikus bináris számláló és a forgótekercses elrendezésű négyfázisú adiabatikus fázisjel generátor együttes energiavesztesége csak a logika méretoptimálásával, illetve a teljes rendszer méretoptimálásával

3.3. VHDL nyelvi átalakításokkal megvalósított tervezés

Az integrált-áramkör tervezés hagyományos módja a tömbvázlat formájában történő megadás, ahol az egyes tömbök rendszerösszetevőket jelölnek, amelyekből a rendszermegadást csak az után készítik el, hogy már bizonyos tervezési döntéseket meghoztak. A tervezési döntések, amelyek egy adott tömbvázlathoz vezetnek, a tervező gyakorlatán múlnak és nem a *tervezési tér* (a lehetséges megoldások halmaza) átvizsgálásán. Ezen kívül az egyes tömbök működési leírása a tervezés folyamán módosulhat, mivel belső ellentmondásokra derülhet fény, és így időigényes újratervezés válik szükségessé. Kívánatosabb lenne a rendszermegadást már a tervezés kezdeti szakaszában rögzíteni, még mielőtt az első tervezési döntések megszületnek, így elkerülhetők lennének az újratervezési ciklusok.

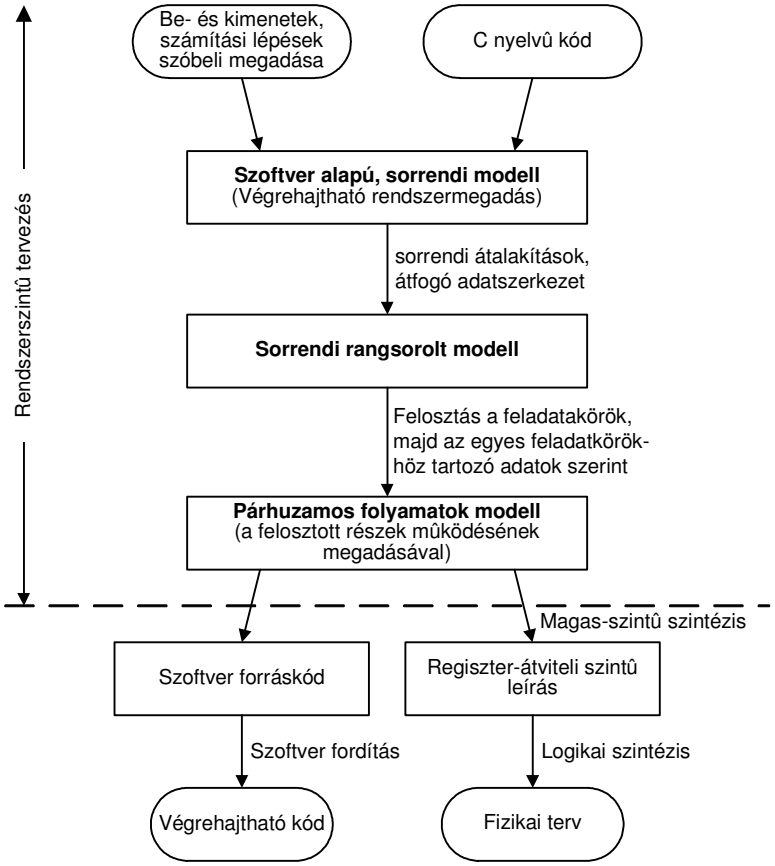
Az előbbiekben vázolt probléma megoldása lehet a teljes rendszer által végrehajtandó feladat megadása valamely leíró nyelven, amely leíró nyelv megvalósítás-független. Számos alkalmazási területen általánosan használható az úgynevezett *végrehajtható rendszermegadás*, amely tervezési döntések meghozatala nélkül rögzíti, hogy fog a rendszer működni, hogyan fog viselkedni.

A végrehajtható rendszermegadásnak számos előnye van, nemcsak az, hogy megadja a rendszer viselkedését, hanem ez szimulációval ellenőrizhető is, és ezen túlmenően dokumentációként szolgál a további tervezési lépésekhez. A végrehajtható rendszermegadás a rendszer működését tulajdonképpen futtatható program formájában írja le. A kiinduló modell egyben referenciaként is szolgál a további tervezési lépésekhez, ugyanis egy későbbi részletesebb modell helyessége a szimulációs eredményeinek a referenciaként szolgáló, kiindulási modell szimulációs eredményeivel való összehasonlításával igazolható.

A végrehajtható rendszermegadásból induló rendszertervezési eljárás végeredményként egymással adatátviteli csatornákon keresztül összekapcsolt rendszerösszetevők készletét adja, minden egyes összetevő működési leírásával. Az egyes összetevők ezután már egymástól függetlenül, különböző szintézis eszközök használatával megvalósíthatók. Például egy processzor összetevő adott esetben a rajta futó program elkészítését jelenti, míg egy nagysebességű jelfeldolgozást végző céláramkör összetevőnek a tervezése magas szintű logikai szintézist jelenthet a viselkedési szintről a regiszter-átviteli szintre, majd onnan további, technológiafüggő szintézist a kiválasztott céleszközre.

3.3.1. A végrehajtható rendszermegadásból induló, VHDL alapú tervezés modellezési szintjei

A VHDL alapú rendszertervezésben alkalmazható, három különböző modellezési szintet dolgoztam ki, amelyek egymásra épülnek úgy, hogy mindegyik az előző szint valamely szempontból történő finomítása [S13]. Az egyes modellek közötti alapvető különbség az elvonatkoztatási szintjükben van és az elvontabb modellhez többlet információt kell adni, hogy egy kevésbé elvont modellhez jussunk. A kidolgozott modellezési szinteket a *3.20. ábra* mutatja.



3.20. ábra. A kidolgozott rendszerszintű tervezési eljárás egymásra épülő modellezési szintjei

Az első modellezési szint, amelynek neve *szoftver alapú, sorrendi modell* a megvalósítandó algoritmust írja le számítások és vezérlések sorozataként. Ez a modell független a későbbi tervezési döntésektől, nem tartalmaz sem a felosztásra (kisebb, kezelhető részekre osztás), sem az időzítésre vonatkozó adatokat, és csupán egyetlen folyamatból áll (ezért is lett sorrendi a neve), valamint helyi adatszerkezetet használ, amelyek ott kerülnek meghatározásra, ahol először értéket kapnak.

A második modellezési szint, név szerint a *sorrendi rangsorolt modell*, ugyancsak sorrendi végrehajtású, amelyben a szoftver alapú modell helyi adatai átfogó, rangsorolt rekordszerkezetbe kerülnek, összefogva egy rekordszerkezetbe az egy feladatkörhöz tartozó adatokat. Ez a modell sorrendi átalakításokkal finomítja és közelebb viszi a tervet a tényleges megvalósításhoz azáltal, hogy előrevetíti a terv lehetséges felosztását, valamint keretet biztosít a következő modellezési lépésnek, amely tulajdonképpen már a tervezendő rendszer tömbvázlatának felel meg.

A harmadik modellezési szinten, amelyet *párhuzamos folyamatok modellnek* neveztem, a rendszertervet felosztjuk párhuzamosan működő folyamatokra úgy, hogy az eltérő jellegű feladatköröket ellátó részek elkülönüljenek, lehetővé téve a későbbiekben az adott feladatkörtől függő további finomítást [S7]. A párhuzamosan futó folyamatok közötti kapcsolattartás elvonatkoztatott adatcsatornák segítségével történik, majd a rendszerterv további finomításával, az órajel alapú időzítés bevezetésével már közvetlenül a szintézishez vezető lépések tehetők [S13].

3.3.2. VHDL nyelvi átalakításokkal megvalósított magas szintű logikai szintézis

A *párhuzamosan működő folyamatok* modellezési szintre eljutva a tervezés további lépéseiben – a magas szintű szintézis eszközkészletét felhasználva, egymás után következő nyelvi átalakításokkal – regiszter-átviteli szintű modellt állítunk elő, mindvégig megtartva a VHDL szabványos, szimulálható nyelvi elemeit.

A VHDL adta szimulációs lehetőség igen jelentős, mivel a gyakorlatban a terv érvényesítése a legidőigényesebb feladat [22]. A nyelvi átalakításokkal megvalósított szintézis eljárás – felhasználva a VHDL több elvonatkoztatási szintű leíró képességét – elérhetővé teszi a szimulációs lehetőségek magas szintű logikai szintézisben történő kihasználását. Ennek eredményeként a digitális részrendszer terve szimulálható mind a viselkedési, mind a regiszter-átviteli szintű modellel, amelyben a felhasznált műveletvégző egységek szintén VHDL nyelven adottak. Ezáltal a szintetizált modell magától érthetődően ellenőrizhetővé válik, amely egy nem VHDL alapú megközelítésben egyébként komoly tervezési feladatot jelent [108].

A nyelvi átalakításokkal megvalósított magas szintű logikai szintézist az angol nevének szavaiból SYLANT-nak (SYnthesis based on LANguage Transform) [S3][S33] neveztem el. A SYLANT megközelítésnek a fő előnyeit a következőképpen foglalhatom össze:

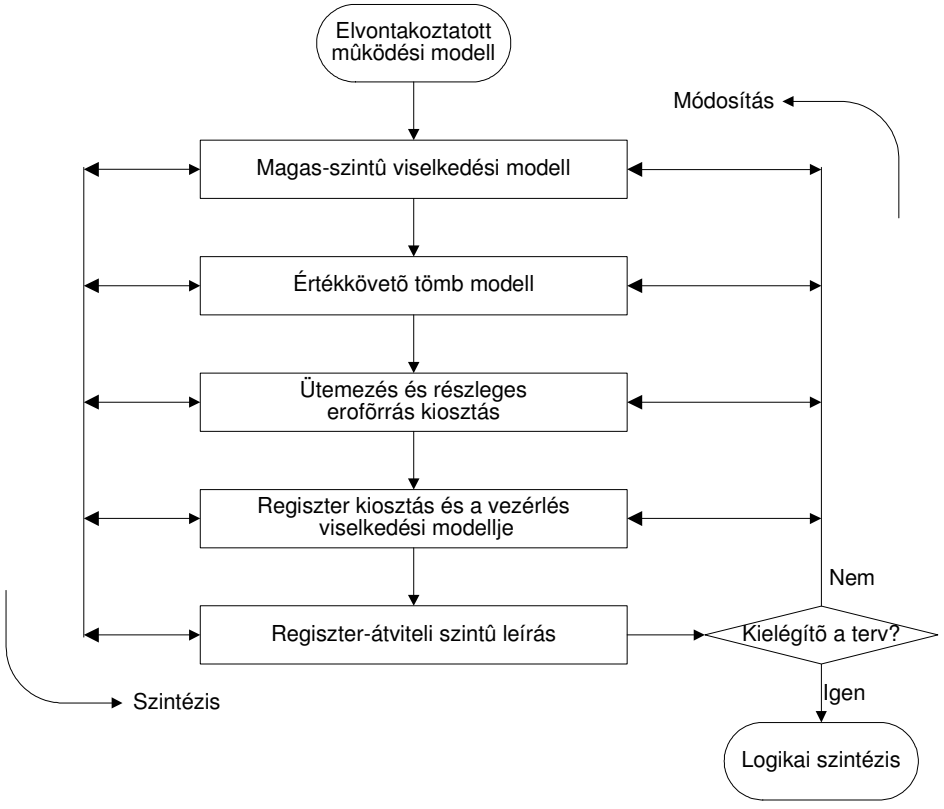
1. A kezdeti tiszta VHDL viselkedési leírás jól átlátható és könnyen megérthető, valamint jól kifejezi a megoldandó feladatban rejlő párhuzamos végrehajtási lehetőségeket.
2. A szintézis folyamat során használt, a mindenkori állapotot leíró ábrázolási forma a VHDL nyelv. Ez a VHDL ábrázolási forma hozzáférhető a tervező számára és szimulálható. A tervezendő áramkör viselkedése ellenőrizhető már a tervezés korai szakaszában is, valamint a tervezés folyamatában bármikor. A szimuláció eredménye felhasználható a további tervezési lépések irányítására.
3. A szimuláció minden elvonatkoztatási szinten futtatható. A kezdeti viselkedési leírás szimulációja, a szimuláció a szintézis közbenső lépéseiben és a szintézis után előállt regiszter-átviteli szintű kód szimulációja közvetlenül végrehajtható. Ugyanazon eszköz, ugyanazon környezet használható a több elvonatkoztatási szintet átívelő szimulációkhoz. A szimulálandó terv az eredeti próbapadba illeszthető anélkül, hogy a próbapadon változtatni kellene. Néhány esetben szükséges azonban a próbapad időzítési viszonyait módosítani.
4. A nyelvi átalakításokkal megvalósított eljárás hatékony eszközkészlet a tervező kezében, közbeavatkozhatósága és a szimulációs eredmények kiértékelése révén lehetővé teszi az alkalmazott szintézis algoritmusok irányítását, meghagyva a lehetőséget a *tervezési tér* hatékony felderítésére.

A. A nyelvi átalakítás alkalmazása

A szintézishez kétfajta ismeret szükséges: a viselkedési megadás és a műveletvégző egységek könyvtára. A VHDL nyelvi átalakítás alkalmazása a kezdeti viselkedési megadást VHDL nyelven igényli [S2], amely jóval magasabb elvonatkoztatási szintű, a szerkezeti leírásnál, vagyis annál, hogy logikai szintézis eszközök bemenete lehessen. A műveletvégző egységek könyvtára a rendelkezésre álló elemi műveletvégző egységek leírását tartalmazza, ezenkívül tartalmazhat még kiegészítő adatokat, paramétereket a műveletvégző egységekről [104].

A VHDL nyelvi átalakításhoz használt műveletvégző-egység könyvtárban minden egyes műveletvégző egység egy VHDL egyed. Az egyedmegadó rész fejléce leírja az egyed nevét, valamint megadja a be- és a kimeneti kapuit, esetleg paraméterekkel kiegészítve. Az egyedmegadó rész törzse pedig az egyed működését és az időzítési viszonyait tartalmazza.

Néhány alapvető feltételezéssel kell élni a kezdeti megadással kapcsolatban. A szintetizálandó áramkör egyetlen VHDL folyamatból áll, amelyet egy adatút és egy vezérlő feladatot végrehajtó véges állapotú automata formájában képzünk le a hardverre [105]. Amennyiben a megoldandó feladat több folyamatból áll, akkor ezeket a folyamatokat egymástól függetlenül szintetizáljuk. A szintézis folyamat lépéseinek összefoglalását a *3.21. ábra* szemlélteti, az egyes lépéseket következő alpontban ismertetem.



3.21. ábra. VHDL nyelvi átalakításokat alkalmazó magas szintű logikai szintézis eljárás fő lépései

B. A szintézis folyamat lépései

A VHDL viselkedési leírás egy tisztán algoritmikus megadás, amely magas szintű műveleteket tartalmaz. A legtöbb algoritmus közös abban az értelemben, hogy bemenő adatokat fogadnak, ezeken az adatokon műveleteket hajtanak végre, ennek során előállítják a kimenő adatokat, majd az egészet kezdik elölről.

A viselkedési leírás számos, egymástól különböző jellegű kialakításban megadható, amely kialakítások jelentéstanilag azonosak, azonban jelöléstanilag eltérőek. Ez alapján szintetizált tervek között jelentős minőségi eltérés lehet [106]. Követve néhány, a kezdeti megadás kialakítására vonatkozó szabályt, olyan kiinduló modellek készíthetők, amelyek hatékonyan szintetizálhatóak hardverre.

A viselkedési leírás teljes szerkezete tartalmazza az egyedmegadó rész fejlécét és törzsét. Az egyedmegadó rész törzsében a folyamat magában foglalja – a szintézisre megfelelő formában – az adat- és a vezérlő folyamot. Az áramkör egyedszintű időzítését az elvonatkoztatott „*start*” és „*ready*” jelek adják.

A tervező írhat ilyen VHDL modellt, vagy használhat egy kiegészített, VHDL-szerű formát, amelyből egyszerűen nyerhető a szimulálható VHDL felépítmény. Ez az *elvonatkoztatott működési modell*, amelyből a kezdeti megadás kialakítására vonatkozó szabályoknak megfelelő szimulálható VHDL modellhez jutunk.

B.1. Elvonatkoztatott működési modell

A nyelvi átalakításokkal történő szintézis folyamatot legjobban egy példa alapján tudom bemutatni. Vegyük kiindulásként szóbeli megadásnak a (3.6) kifejezést, amely egy matematikai leírás, a lehető legelvontabb, hiszen az alapvető logikai feladatkörön kívül semmilyen más információt nem tartalmaz. Az *a*, *b*, *c* és *d* a bemenetek, az *y* pedig a kimenet:

$$y = (a + b) \cdot \mathrm{abs}(c - d) \tag{3.6}$$

A fenti szóbeli megadásnak megfelelő *elvonatkoztatott működési modell*, amit a továbbiakban használok, a *3.1. listán* látható.

```
program example (a, b, c, d : in integer;
   y : out integer) is
variable va, vb : integer;
begin
   va := a + b;
   if c > d then
      vb := c - d;
   else
      vb := d - c;
   end if;
   va := va * vb;
   y := va;
end;
end program;
```

3.1. lista. Elvonatkoztatott működési modell

A magas szintű műveletek a műveletvégző-egység könyvtárban található elemekre vonatkoznak olyan értelemben, hogy a könyvtárban kell lenni legalább egy olyan műveletvégző egységnek, amely az adott műveletet képes végrehajtani.

Kifejlesztettem egy szoftver-eszközt, amely az elvonatkoztatott működési modellt szimulálható VHDL modellé alakítja. Ez a szoftver-eszköz az említett átalakítással párhuzamosan elkészíti a szimulációhoz szükséges próbapad vázát is. A próbapad-egyed szerkezete tartalmazza a tervnek megfelelő jelösszetevőket, amelyeken keresztül a tervezendő áramkör a próbapadhoz csatlakozik, valamint ezen jelösszetevők szolgának a szimulációs adatok figyelésére is. A tervező a próbapad kimeneti kapuira bármilyen jelértékeket előírhat, így tetszőleges tesztvektorok használhatók a megkívánt működés ellenőrzésére. Az önműködően elkészített próbapad-váz a fentieken kívül tartalmaz még egy órajel-generátort és a szimulációs folyamat időzítésére szolgáló a „*start*” és „*ready*” vezérlőjel-szerkezetet is.

B.2. Magas szintű viselkedési modell

A kezdeti VHDL modell, amelynek neve *magas szintű viselkedési modell*, önműködően előállítható az elvonatkoztatott működési modellből. Az előző alpontban megadott elvonatkoztatott működési modellnek megfelelő *magas szintű viselkedési modell* a *3.2. listán* található.

```
entity example is
   port (pa, pb, pc, pd : in integer;
         py : out integer;
         start : in bit;
         ready : out bit);
end example;

architecture HBM of example is
begin process
variable a, b, c, d: integer;
variable va, vb : integer;
begin
   wait until start = '1';
   ready <= '0';
   a := pa; b := pb; c := pc; d := pd;
   wait until start = '0';
   va := a + b;
   if c > d then
      vb := c - d;
   else
      vb := d - c;
   end if;
   va := va * vb;
   py <= va;
   ready <= '1';
end process;
end HBM;
```

3.2. lista. Magas szintű viselkedési modell

A modellben létrehoztuk a be- és kimeneti kapuknak megfelelő változókat, majd magát a folyamatot egy feltételes *wait* utasítással kezdtük. Ezután a bemeneti kapukon lévő adatok beolvasása következik, majd az algoritmus végrehajtása a

változókat használva, valamint az eredmények eltárolásra a kimeneti kapuknak megfelelő változókban. Végül a változókból a kimeneti adatokat a kimeneti kapukra irányítjuk.

Ez a VHDL kód szimulálható, a tervező a működési szimulációval még a szintézis megkezdése előtt ellenőrizheti a terv működését. A szintézis során a magas szintű viselkedési modellt egymás utáni nyelvi átalakításokon keresztül hardverre képezzük le – egy adatút és egy vezérlő feladatot végrehajtó véges állapotú automata formájában. Minden egyes átalakítási lépés fokozatosan gazdagítja a modellt szerkezeti adatokkal, amíg a teljes regiszter-átviteli szintű leírás kibontakozik.

B.3. Értékkövető tömb modell

Az első átalakítási lépésben az adatút és a vezérlő rész különválasztása történik. Az átalakítási lépésnek megfelelő modell a *3.3 listán* látható.

```
architecture VTB of example is
   signal a0, b0, c0, d0 : integer;
   signal va0, va1, vb0 : integer;
   signal cnt0 : bit;
   type pltype is (empty, token);
   signal df0 : pltype;
   constant t1 : time := 50 ns;
begin process
variable a ,b ,c ,d : integer;
variable va : integer;
begin
   wait until start = '1';
   ready <= '0';
   a := pa; b := pb; c := pc; d := pd;
   wait until start = '0';
   a0  <= a; b0 <= b; c0 <= c; d0 <= d;
   df0 <= token; wait for t1; df0 <= empty;
   va  := va1; py <= va;
   READY <= '1';
end process;

dfb0 : block (df0 = token)
begin
   va0  <= guarded add (a0, b0);
   cnt0 <= guarded comp (c0, d0);
   vb0  <= guarded sub (c0, d0) when cnt0 = '1'
      else sub (d0, c0);
   va1  <= guarded mul (va0, vb0);
end block;
end VTB;
```

3.3. lista. Értékkövető tömb modell

Az adatút és a vezérlő rész szétválasztása lehetővé teszi a viselkedési szintű elemi műveletek algoritmikus eszközökkel történő feldolgozását. Az időzítés vezérlése továbbra is a folyamaton belül marad, azonban a szűkebb értelemben vett adatút a folyamaton kívülre, egy tömbbe kerül. Ez a tömb úgynevezett *őrzött tömb*, amelyen belül őrzött együttfutó (azaz egyidejű) értékadások szerepelnek. Az őrzött együttfutó értékadások írják le az adatfüggőséget a viselkedési szintű elemi műveletek között,

míg az értékadások időzítését a folyamat törzsében elhelyezett utasítások vezérlik [S4].

A *3.3. listán* látható modell neve *értékkövető tömb modell* [107]. A modellben jelek szükségesek a folyamat és a tömb közötti adatcseréhez. A *pltype* típust felsorolt típusként határoztuk meg az *empty* és a *token* értékekkel. A *df0* jel értéke határozza meg a *dfb0* tömb őrző-kifejezésének értékét, így vezérli a tömbben szereplő értékadások végrehajtását. Az értékadások akkor hajtódnak végre, ha az őrző-kifejezés igaz értékűvé válik, vagy ha az őrző kifejezés igaz értéke alatt a kifejezés jobb oldalán álló jelek valamelyikén esemény történik. Az egyes értékadások sorrendje megegyezik a magas szintű viselkedési modell változóinak sorrendjével.

Az értékadások jobb oldalán álló függvények a viselkedési szintű elemi műveletek műveletvégző-egység könyvtárbeli valamelyik megfelelőjét takarják, azonban – a gyorsabb szimuláció érdekében – az elemi műveletek megfelelő viselkedési modellje is használható. Például az összeadás művelet viselkedési modellje a *3.4. lista* szerint akár olyan egyszerű is lehet, mint a VHDL „+” operátor, de nem alapművelet esetén legalább a viselkedési modellre szükség van.

```
function add (a,b:in integer) return integer is
begin
  return a+b;
end add;
```

3.4. lista. Az összeadás művelet legegyszerűbb viselkedési modellje

A magas szintű szintézis rendszerek viselkedési ábrázolási formaként általában vezérlési-adatfolyam gráfot alkalmaznak. Ez egy sajátságos ábrázolási forma, amely többnyire nem teszi lehetővé a szimulációt, vagy ha mégis lehetővé teszi, akkor azt egy járulékos, a magas szintű szintézis rendszeren kívüli környezetben, amely nem egy hardverleíró nyelvi környezet. Az *értékkövető tömb modell* VHDL nyelven adott, ami szimulálható és az adat- és vezérlési folyamot ugyanolyan kifejezőképességgel ábrázolja, mint a vezérlési-adatfolyam gráf.

B.4. Ütemezés és részleges erőforrás kiosztás

Az egyes viselkedési szintű elemi műveletek vezérlési lépésekbe történő ütemezése és a viselkedési megadás megvalósításához szükséges műveletvégző egységek részleges kiosztása történik ebben az átalakítási lépésben. Az átalakítás után előállt modell minősége nagyban függ az alkalmazott ütemező algoritmustól és az előírt tervezési megkötésektől. Az átalakítás egy lehetséges eredménye a *3.5. listán* látható.

```
architecture SED of example is
   signal a0, b0, c0, d0 : integer;
   signal va0, va1, vb0 : integer;
   signal cnt0 : bit;
   type pltype is (empty, token);
   signal df1, df2, df3 : pltype;
   constant t1 : time := 50 ns;
   constant dt:time:=1 ns;
```

```
begin process
variable a ,b ,c ,d : integer;
variable va : integer;
begin
   wait until start = '1';
   ready <= '0';
   a := pa; b := pb; c := pc; d := pd;
   wait until start = '0';
   a0  <= a; b0 <= b; c0 <= c; d0 <= d;
   df1 <= token; wait for t1; df1 <= empty;
   wait for dt;
   df2 <= token; wait for t1; df2 <= empty;
   wait for dt;
   df3 <= token; wait for t1; df3 <= empty;
   va  := va1; py <= va;
   READY <= '1';
end process;

dfb1 : block (df1 = token) begin
   va0  <= guarded add (a0, b0) after 20 ns;
   cnt0 <= guarded comp (c0, d0) after 20 ns;
end block;
dfb2 : block (df2 = token) begin
   vb0 <= guarded sub (c0, d0) after 20 ns when
 cnt0 = '1' else sub (d0, c0) after 20 ns;
end block;
dfb3 : block (df3 = token) begin
   va1 <= guarded mul (va0, vb0) after 40 ns;
end block;
end SED;
```

3.5. lista. Egy lehetséges ütemezés

Minden egyes tömb egy vezérlési lépést jelent az ütemezésben. A különböző vezérlési lépésbe ütemezett műveleteket, különböző tömbökben helyezzük el [S5]. Az egyes tömbök ütemezett indításáért a folyamat a felelős. A tömbökben az egyes értékadások után álló *after* nyelvi elem jelöli a kiválasztott műveletvégző egységen az adott művelet végrehajtásához szükséges időt, ha műveletvégző egységek viselkedési modelljét alkalmazzuk. Szerkezeti modell esetén maga a modell tartalmazza a késleltetést, így ekkor az értékadásnál ezt már nem tüntetjük fel.

Az átalakítási lépés önműködővé tételéhez a legtöbb jól ismert ütemező és erőforrás kiosztó algoritmus használható. A tervező feladata – a tervezési korlátok figyelembevételével – választani a rendelkezésre álló algoritmusok közül és előírni a kiválasztott algoritmus tervezési korlátoknak megfelelő paramétereit. A főbb paraméterek például a kívánt újraindítási idő, a használható hardver erőforrások (mint pl. összeadók és szorzók) megengedett legnagyobb száma, legnagyobb megengedett lappangási idő, stb., azonban egyéb megkötések is adhatók, amelyek például a tesztelhetőségre vagy a teljesítményfelvételre vonatkoznak [S10].

A fentieknek megfelelően számos viselkedési felépítmény állhat ezen átalakítási lépés mögött. Ha egy adott terv – valamely szempontból – nem kielégítő, akkor újraütemezéssel vagy viselkedési átalakítások [27] alkalmazásával a legtöbb esetben elérhető, hogy a kívánt követelményeknek megfelelő tervet kapjunk.

Különbséget kell tenni az egyes viselkedési szintű elemi műveletek különböző megvalósításai között. Ugyanazon viselkedési szintű elemi műveletnek számos

különböző megvalósítása létezhet egyidejűleg a műveletvégző-egység könyvtárban, amelyeknek eltérő jellemzőik lehetnek (pl. műveletvégzési idő, szilíciumterület-foglalás, teljesítményfelvétel, stb.). Például az összeadás művelete megvalósítható többek között a soros *ripple-carry* vagy a párhuzamos *átvitelgyorsító* összeadó típusokkal, amelyeket a műveletvégző-egység könyvtárbeli neveikkel különböztetünk meg. A viselkedési szintű elemi műveletek számos különböző megvalósítása közötti választás a szintézis során lehetséges és egyben szükséges is.

B.5. Regiszter kiosztás és a vezérlés viselkedési modellje

Az ütemezés és a szükséges műveletvégző egységek kiosztása után a vezérlő rész viselkedési modellje előállítható egy véges állapotgép megadással. A példában egy állapotregiszteres megoldást alkalmaztam, az állapotregisztert *fsm*-el jelölve. Ugyanebben az átalakítási lépésben a közbenső adatokat tároló regiszterek és a regisztereket a műveletvégző egységekkel összekötő multiplexerek kiosztása történik, párhuzamosan az elemi műveleteknek a kiosztott műveletvégző egységekhez való végső kötésével. Az átalakítás után előállt adatút modell a *3.6. listán* található.

```
DATAPATH :block
begin
   a0 <= pa when fsm = st_load else a0;
   b0 <= pb when fsm = st_load else b0;
   c0 <= pc when fsm = st_load else c0;
   d0 <= pd when fsm = st_load else d0;
   py <= va1;
   va0  <= add (a0, b0) after 20 ns when fsm = st1 else va0;
   cnt0 <= comp (c0, d0) after 20 ns when fsm = st1 else cnt0;
   vb0  <= sub (m1, m2) after 20 ns when fsm = st2 else vb0;
   va1  <= mul (va0, vb0) after 40 ns when fsm = st3 else va0;
   m1   <= c0 when cnt0 = '1' else d0 when cnt0 = '0' else m1;
   m2   <= d0 when cnt0 = '1' else c0 when cnt0 = '0' else m2;
end block;
```

3.6. lista. Regiszter kiosztás és a vezérlés viselkedési modellje

A kiosztott regiszterek – ha szükséges, akkor multiplexereken keresztül – a műveletvégző egységekhez kapcsolódnak, teljessé téve az adatutat. A modellben ponttól-pontig tartó összeköttetést alkalmaztam, azonban az olyan összeköttetések, amelyek egyidejűleg soha sem használtak, a későbbiekben buszokká vonhatók össze, lecsökkentve ezzel a szükséges multiplexer-bemenetek darabszámát. A további szintézishez a későbbiekben az egyes viselkedési szintű elemi műveletek viselkedési modelljét a kiválasztott műveletvégző-egység könyvtárbeli megvalósítására kell cserélni.

B.6. Regiszter-átviteli szintű modell

Az adatút teljessé tétele után a vezérlő rész viselkedési modellje lecserélhető a véges állapotgép logikai szintézist lehetővé tévő regiszter-átviteli szintű modelljére. Ez a modell az adatútnak megfelelően készül, ami a gyakorlatban az adatúttól függő rész beillesztését jelenti egy általános vázba.

```
CONTROLLER :block
begin
SHFSM <=st_load when (START ='1' and CLK ='1'
 and CLK'event)
else st1 when (FSM =st_load and CLK ='1' and CLK'event)
else st2 when (FSM =st1 and CLK ='1' and CLK'event)
else st3 when (FSM =st2 and CLK ='1' and CLK'event)
else stwait when (FSM =st3 and CLK ='1' and CLK'event)
else SHFSM;
shready <= '0' when FSM =st_load
           else '1' when FSM =stwait else shready;
count <=0 when FSM'event
          else count +1 when (CLK ='1' and CLK'event)
          else count;
FSM <=st_load after dt when SHFSM =st_load
  else st1 after dt when SHFSM =st1
  else st2 after dt when (SHFSM =st2 and count =2)
  else st3 after dt when (SHFSM =st3 and count =2)
  else stwait after dt when (SHFSM =stwait and count =2)
  else FSM;
READY <=shready after dt;
end block;
```

3.7. lista. A vezérlő rész regiszter-átviteli szintű modellje

Az adatút és a vezérlő rész ezen modellje adja a kezdeti viselkedési leírás regiszter-átviteli szintű modelljét. Ez a regiszter-átviteli szintű kód megfelelő formátumú ahhoz, hogy bemenete lehessen olyan logikai szintézis eszközöknek, amelyek már – részben – önműködően képesek a terv kapuszintű megvalósításának előállítására, illetve közvetlenül a gyártáshoz szükséges adatok szolgáltatására.

C. Megvalósítás

Az előző alpontban részletezett VHDL nyelvi átalakítás-sorozatot önműködővé tettem azzal, hogy kidolgoztam és következetesen alkalmaztam egy megfelelő programozási gyakorlatot, amelyet megvalósítottam a SYLANT programrendszerben. A SYLANT programrendszer a kezdeti viselkedési megadásból kiindulva lépésről-lépésre előállítja a kívánt regiszter-átviteli modellt.

Minden egyes átalakítási lépés jól meghatározott egyrészt az átalakítás kimeneti modellje által, másrészt az átalakító művelet, amely a bemeneti modellt a kimeneti modellé alakítja, valamint az átalakítás bemeneti modellje által. A SYLANT programrendszer értelmezi az adott lépés bemeneti modelljének megfelelő VHDL kódot, végrehajtja rajta az átalakítási lépésnek megfelelő átalakítást, és kimenetként az adott lépésnek megfelelő kimeneti modellt szolgáltatja.

Az *elvonatkoztatott működési modellről* a *magas szintű viselkedési modellre*, valamint a *magas szintű viselkedési modellről* az *értékkövető tömb modellre* történő átalakítási lépés elvileg nem igényel közbeavatkozást a tervező részéről. Ezek az átalakítási lépések könnyen önműködővé tehetők és alkalmazásuk nem változtatja meg a modell viselkedését. A rendszer leglényegesebb része az ütemező és erőforrás kiosztó algoritmus, ahol szükséges a tervező közbeavatkozása a tervezési korlátok kielégítése érdekében. A szintézis folyamatban körbejárás is lehetséges, esetleg

ismételt átalakítások szükségesek, amíg a kívánt eredmény előáll, ahogy ez a *3.21. ábrán* is látszik.

A nyelvi átalakítás eljárásban bármely ütemező és erőforrás kiosztó algoritmus alkalmazható. Például, az előző alfejezetben bemutatott szintézis példa kapcsán egy teljesítményfelvételre érzékeny ütemezési technikát [86] alkalmaztam. A regiszter és a multiplexer kiosztás, valamint az elemi műveleteknek a kiosztott műveletvégző egységekhez való végső kötése a gyakorlati alkalmazásoknak megfelelően önműködővé tehető. Természetesen a tervezőnek ezeket a lépéseket is célszerű felügyelni hiszen például a végső kötéseknek a teljesítményfelvételre is hatásuk van [S9].

3.4. Adiabatikus működésű áramkörök ütemezése

Az adiabatikus töltésvisszanyerő logika alkalmazása egy igen ígéretes lehetőség nagyon alacsony fogyasztású VLSI áramkörök készítésére. Az ilyen áramkörök az alacsony fogyasztást az ohmikus veszteség korlátozásával és a logikai kapuk terhelőkapacitásaiban tárolt energia visszaforgatásával érik el, amihez a szokásos egyenáramú táplálás helyett úgynevezett rezonáns tápegységet igényelnek. Az adiabatikus töltésvisszanyerő kapcsolások energiahatékonysága elsősorban alacsony frekvenciákon jelentős, és annak ellenére, hogy ez a hatékonyság a működési frekvencia növelésével romlik, energiafogyasztásuk még több 100 MHz-en is kedvezőbb, mint a statikus CMOS kapcsolásé [46][61]. Ezenkívül az adiabatikus kapcsolás energiahatékonysága egy adott működési frekvenciára optimalizált tranzisztorművel még tovább javítható [S18][S24].

Az utóbbi években számos új adiabatikus kapcsolás jelent meg a szakirodalomban, mint például a *tRERL* [53] és az *nRERL* [54], a *PAL* [45], a *2N-2P* [57], a *2N-2N2P* [58], a *CAL* [44], valamint a *TSEL* [46] és az *SCAL* [61] logikák. Ezekkel a logikai áramkörökkel bármilyen logikai függvény megvalósítható, gyakorlati alkalmazásukra jellemző példa az adiabatikus mikroprocesszor [48][64] és a FIR szűrő [49]. Nagy kapacitású memóriák kialakítása adiabatikus áramkörökkel azonban nem hatékony a túlságos szilíciumterület-többlet igénye miatt. Viszont az adiabatikus kapcsolások a felépítésükből adódó pipeline elvű működésnek köszönhetően előnyösen alkalmazhatók minden olyan alkalmazási területeken, ahol egymást követő adatokon ugyanazokat a műveleteket kell végrehajtani; ilyen alkalmazási terület például a digitális jelfeldolgozás.

A művelet-végrehajtási idő, illetve az átbocsátás fontos szempont a sokszor valósidejű digitális jelfeldolgozó rendszerekben. Az átbocsátás növelésének egyik nyilvánvaló módja az órajel periódusidejének csökkentése, azonban ekkor pont az energiahatékonyság – amiért az adiabatikus kapcsolást alkalmazzuk – romlik. Ezzel szemben a szerkezetbeli optimálás – mint például a párhuzamosság vagy a pipeline elv kihasználása – sokkal hatékonyabb eszköz az átbocsátás növelésére, mint pusztán az órajel-frekvencia növelése.

A gyakorlatban előforduló feladatok bonyolultsága – még az egyszerűbbek esetén is – megköveteli az önműködő tervezőeszközök használatát. Bár korábbi munkák foglalkoztak teljes adiabatikus rendszerek készítésével, adiabatikus adatutak viselkedési leírásból való előállítása mindezidáig megoldatlan maradt. Ismert, hogy a magas szintű logikai szintézis egyik legkritikusabb lépése az ütemezés, amely megadja, hogy mely műveletek végrehajtása melyik vezérlési lépésben kezdődik. Bár a szakirodalom bővelkedik ütemező algoritmusokban, valamint ezen belül a pipeline-technika kihasználására történő ütemezés is jól kidolgozott [7]-[8][77]-[81], adiabatikus kapcsolások ütemezésére azonban egyik technika sem alkalmas. Ezen eljárások adiabatikus áramkörökre vonatkozó alkalmazhatóságának hiányossága, hogy nem veszik figyelembe az adatútba az ütemezés után beépítendő multiplexerek műveletvégzési idejét.

3.4.1. Hagyományos ütemezés

A szakirodalomban található – hagyományos – magas szintű szintézis rendszerek ütemezés és erőforrás kiosztás után a regiszterek és a műveletvégző egységek közötti összeköttetéseket multiplexerek segítségével valósítják meg. A multiplexerek teszik lehetővé a műveletvégző egységek műveletek közötti megosztását. Ezekben a hagyományos rendszerekben a multiplexerek késleltetése nem gond, mert ezek a rendszerek egyrészt nem érzékenyek az időzítésre, másrészt nem is képesek pontos időzítést előállítani, mivel az csak a regiszter-átviteli szinten adott összetevők szilíciumon való elhelyezése és összeköttetése után lehetséges.

Tekintsük példaként a *3.8. listán* látható VHDL nyelvű viselkedési megadást, amelyben három összeadás műveletet kell elvégezni, és tegyük fel, hogy a tervezési megkötések legfeljebb két összeadó használatát engedik meg.

```
entity example is
   port (pa, pb, pc, pd : in integer;
         py : out integer;
         start : in bit;
         ready : out bit);
end example;
architecture HBM of example is
begin process
variable a, b, c, d: integer;
variable va, vb, vc : integer;
begin
   wait until start = '1';
   ready <= '0';
   a := pa; b := pb;
   c := pc; d := pd;
   wait until start = '0';
   va := a + b;
   vb := c + d;
   vc := va + vb;
   py <= vc;
   ready <= '1';
end process;
end HBM;
```

3.8. lista. Egy példa viselkedési megadás három összeadás művelettel

Egy lehetséges és megvalósítható ütemezést és erőforrás kiosztást a *3.4. táblázat* mutat, amely azzal a feltételezéssel készült, hogy az összeadás-művelet mindkét összeadón egy vezérlési lépés alatt végrehajtható.

vezérlési lépés	1. összeadó	2. összeadó
1	a+b	c+d
2		va+vb

3.4. táblázat. Egy lehetséges ütemezés és erőforrás kiosztás

A három összeadás két összeadón történő megvalósításához – illetve a példában az adatfüggőségek miatt is – két vezérlési lépésre van szükség, ahol az első összeadó a második vezérlési lépésben tétlen. A második összeadó mindkét vezérlési lépésben

dolgozik, különböző adatokon, amely adatokat multiplexereken keresztül kapja. A multiplexer és az összeadó – mint láncolt műveletvégző egységek – együttes késleltetési idejének, valamint a fizikai vezetékezés késleltetési idejének bele kell férnie egy vezérlési lépés idejébe.

3.4.2. Ütemezés alkalmazása adiabatikus logikákra

Adiabatikus áramkörökben a multiplexerek késleltetése a logikai kapuk fázisjel-vezérelt működéséből ered, ahol minden egyes fázisjelre csak egyetlen logikai szint kiértékelése megy végbe. Egy adiabatikus multiplexer logikai mélysége legalább egy, ami legalább egy fázisjelet igényel és így a multiplexer művelet végrehajtási ideje is igénybe vesz legalább egy vezérlési lépést.

Azáltal, hogy ütemezéskor valamennyi művelet valamennyi logikai szintjét valamely vezérlési lépéshez rögzítjük, multiplexerek nem szúrhatók be az adatútba az ütemezés után, mivel ki nem használt vezérlési lépések jellemzően nem állnak rendelkezésre a multiplexer művelet végrehajtására. Emiatt adiabatikus logikai áramkörök ütemezésére a hagyományos megközelítés nem alkalmazható [S27].

Ha az adiabatikus adatútba multiplexereket építünk, akkor a szükséges vezérlési lépések száma megnő, ami hatással van a többi művelet ütemezésére is. Ezért a gyakorlatban is megvalósítható, adiabatikus logikával felépített adatút készítéséhez a multiplexerek műveletvégzési idejét számításba kell venni az ütemezéskor.

Példaként tekintsük még egyszer a *3.8. listán* található viselkedési megadást, amelynek négyfázisú adiabatikus rendszerre egy lehetséges ütemezését és erőforrás kiosztását – ugyancsak legfeljebb két összeadót megengedve – a *3.5. táblázat* mutatja.

vezérlési lépés	fázis	1. összeadó						2. összeadó					
1	1	+1											
2	2		+1					+2					
3	3			+1					+2				
4	4				+1					+2			
5	1					+1					+2		
6	2						+1					+2	
7	3												+2
8	4												
9	1												
10	2							+3					
11	3								+3				
12	4									+3			
13	1										+3		
14	2											+3	
15	3												+3
16	4												

3.5. táblázat. Egy lehetséges ütemezés és erőforrás kiosztás adiabatikus műveletvégző egységek használatával

A *3.5. táblázatban* helyhiány miatt az „a+b” műveletet a „+1”, a „c+d” műveletet a „+2”, a „va+vb” műveletet pedig a „+3” jelöli, az adatút szélességét 16 bitre választva az összeadók 6 logikai szint mélységűek.

A táblázatból látható egyrészt, hogy az első összeadó, amelyik csak az „a+b” műveletet hajtja végre, az első vezérlési lépésben megkezdi működését. A második összeadó, amelyik a „c+d” és a „va+vb” műveleteket végzi, bemenő adatait multiplexereken keresztül kapja. Mivel a multiplexer művelet végrehajtása egy vezérlési lépést vesz igénybe, ezért ez az összeadó csak a második vezérlési lépésben kezdheti meg a működését. Másrészt az is látható a táblázatban, hogy bár a 8. vezérlési lépésben mind az „a+b”, mind a „c+d” művelet eredménye rendelkezésre áll, a „va+vb” művelet nem kezdhető meg 10. vezérlési lépés előtt, mivel a „va” és a „vb” adatoknak át kell haladniuk az első fázisjelre kapcsolt multiplexeren. Ez legkorábban a 9. vezérlési lépésben lehetséges, így az adatok csak a 10. vezérlési lépésben léphetnek be a második összeadó második fázisjelre kötött első logikai szintjére. Harmadrészt a *3.5. táblázatból* az is kiolvasható, hogy az így felépített adatút lappangási ideje 15, és az így elérhető legkisebb pipeline újraindítási idő pedig 12 vezérlési lépés, holott, a (3.7) összefüggés értelmében (lásd később) – a lappangási idő megnövekedése árán – az újraindítási időre 8 vezérlési lépés is elérhető lenne.

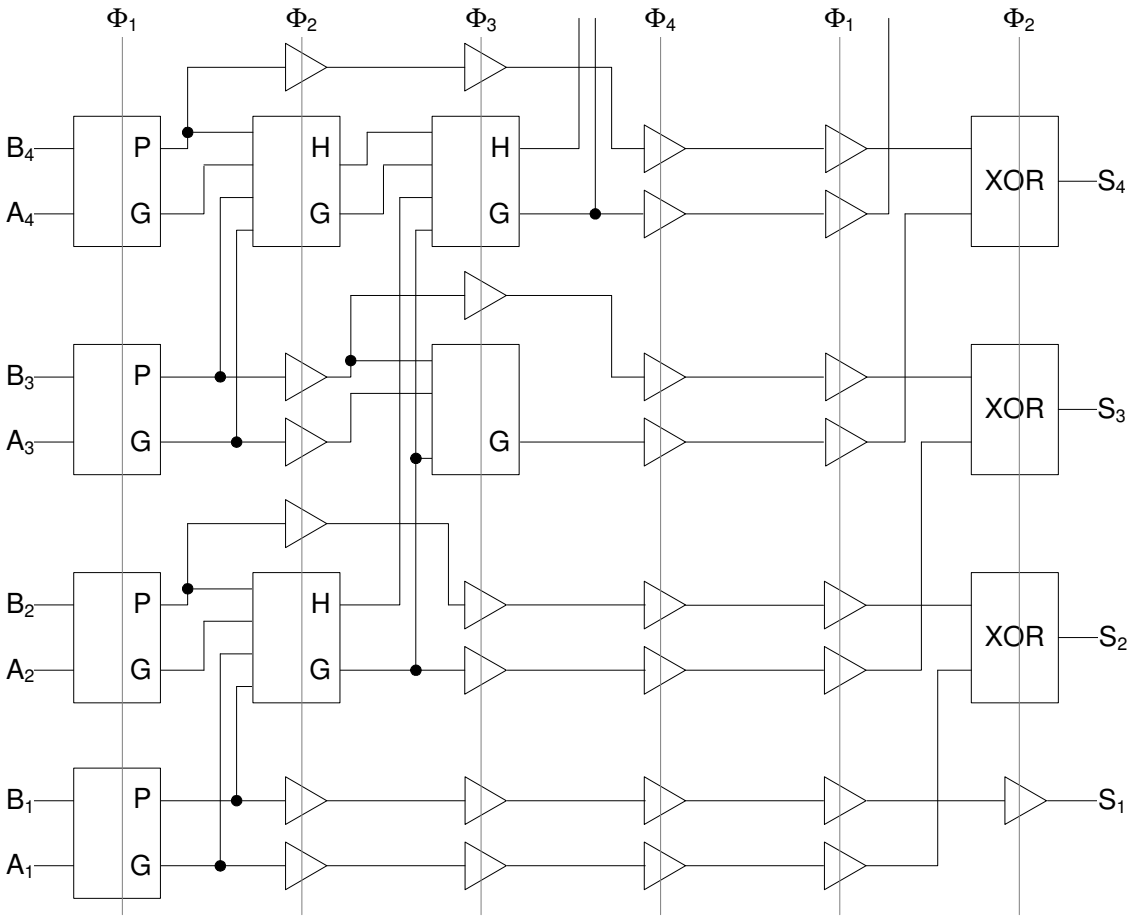
A következő alfejezetben röviden bemutatom az ütemezéshez használt adiabatikus logikának az ütemezés szempontjából fontos jellemzőit. A 3.4.4. alfejezetben ismertetem a kidolgozott ütemező eljárásomat, amely a megadott újraindítási idő mellett a lehető legkisebb számú műveletvégző egységek felhasználásával felépített adatút egy lehetséges ütemezését adja. Ez az eljárás egy módosított lista ütemezőt alkalmaz, így nem feltétlenül eredményezi a legkisebb számú műveletvégző egységek mellett a legkisebb lappangási időt. Ezért kidolgoztam egy bármilyen szempontból optimális megoldást nyújtó, egészértékű lineáris programozáson alapuló algoritmust is, ezt az algoritmust a 3.4.5. alfejezetben mutatom be. Bár az egészértékű lineáris programozás optimális eredményt ad, az egyenletek gyakorlati megoldása időigényes és nehéz, ezért ez az algoritmus csak kisebb feladatok megoldására alkalmas. Végül a 3.4.7. alfejezetben egy olyan modellezési technikát ismertetek, amely VHDL nyelven lehetővé teszi az adiabatikus adatút együtt szimulálását a digitális rendszer többi részével.

3.4.3. Az ütemezéshez használt adiabatikus logika

Az ütemezéshez használt adiabatikus logika a 3.1. alfejezetben bemutatott négyfázisú javított tulajdonságú adiabatikus kapcsolás. A négy fázisjel egymástól 90º-kal eltolt, mindegyik a feltöltés, a tartás, a kisütés és a várakozás periódusokat ismétli. A logikai kapu bemeneteinek stabilnak kell lenni a fázisjel feltöltő periódusában, és a logikai kapu kimenetén a logikai szint érvényes a tartás periódus alatt. Kisütés alatt a logikai kapu terhelőkapacitásában tárolt energiát a rezonáns tápegységbe visszavezetjük, míg a várakozó periódusban az előző logikai szinten lévő kapu (amely ekkor a feltöltő periódusban van) előállítja az új bemeneti értékeket. Így a logikai kapuk pipeline-elven működtethetők anélkül, hogy külön pipeline-regiszterekre szükség lenne.

A. Műveletvégző egységek

Valamennyi műveletvégző egységet a pipeline működésnek megelelően alakítunk ki, puffereket használva azokon a logikai szinteken, ahol egyébként logikai műveletre nincs szükség. Példaként a 16-bites átvitelgyorsító összeadó alsó négy bitjének pipeline-elvű felépítése valamint az egyes logikai szintjeinek a megfelelő fázisjelhez való csatlakozása látható a *3.22. ábrán*. Ezzel a felépítéssel maga a műveletvégző egység is egy pipeline, amely egy fázisjel minden felfutásakor új művelet végrehajtásába kezdhet, de a lappangási ideje rendszerint nagyobb, mint a fázisjel periódusideje. Ezáltal ütemezéskor egy adiabatikus műveletvégző egység második szintű pipeline-ként kezelendő. Példaként az átvitelgyorsító összeadó $2+log_2N$ számú logikai szinten megvalósítható, ahol N az összeadó bitszélessége. A 16-bites átvitelgyorsító összeadó lappangási ideje 1,5 fázisjel-ciklusidő, ami a négyfázisú rendszerben hat vezérlési lépésnek felel meg, azonban az összeadó azonos fázisjelenként, azaz minden negyedik vezérlési lépésben új bemeneti adatokat fogad.



3.22. ábra. 16-bites adiabatikus átvitelgyorsító összeadó alsó négy bitjének pipeline-elvű felépítése és a fázisjelek bekötése

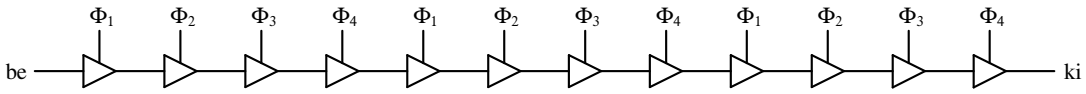
B. Multiplexerek

Egy, a *3.1. ábra* szerint felépített összetett logikai kapuban elvileg sokbemenetű multiplexer is megvalósítható, gyakorlati megfontolásokból azonban – a logikai kapu összetettségét is szem előtt tartva – nem érdemes négynél több bemenetet kialakítani. Ha a lappangási idő egyébként nem indokolja, akkor érdemesebb a négynél több bemenetű multiplexert multiplexerek láncba kapcsolásával, több logikai szinten megvalósítani.

C. Regiszterek

Regisztereket egybites tárolókból építhetünk fel. Az alkalmazott adiabatikus egybites tároló négy logikai kapu gyűrűbe kapcsolásából áll, ahogy azt a 3.1.2. alfejezetben bemutattam. A gyűrűben az egyik logikai kapu egy összetett kapu, amely tartalmazza a megfelelő logikát az új logikai állapot beírására, illetve „átlátszó" akkor, ha nem írunk be új értéket. Ez a kapu tulajdonképpen egy kétbemenetű multiplexer, míg a maradék három kapu puffer, amelyek továbbítják a megfelelő logikai értéket a gyűrűben a következő logikai kapuhoz.

Pipeline-elven működő tárolóelemet kialakíthatunk adiabatikus logikával úgy is, hogy annyi láncba kapcsolt puffert alkalmazunk, ahány vezérlési lépésen keresztül a logikai állapot megőrzésére szükség van. Erre mutat példát a *3.23. ábra*, ahol a pipeline-elven működő tárolóelem a bemenő adatot 12 vezérlési lépésen keresztül tartja. Ennek a megoldásnak előnye, hogy nem igényel vezérlőjelet (nem tartalmaz multiplexert), hátránya lehet viszont, hogy hosszú lánc esetén a szükséges logikai kapuk száma nagy. Ezért ez a megoldás nagy újraindítási idő mellett csak rövid láncok esetén alkalmazható. Kis újraindítási idő mellett az előbbi hátrány nem jelentkezik, mivel a pipeline adott esetben teljesen kihasznált lehet, így a szükséges logikai kapuk száma egyedi tárolóelemek alkalmazásával csak több lenne.

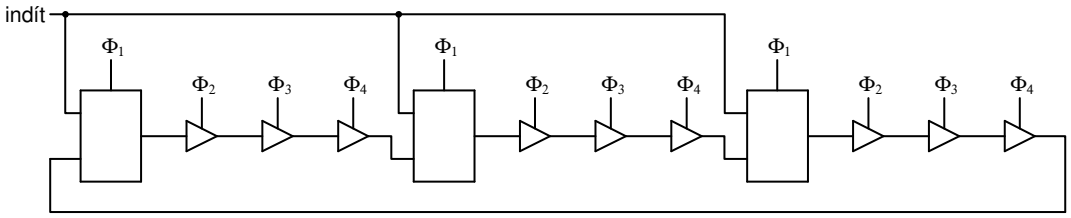


3.23. ábra. Pipeline-elven működő tárolóelem 12 vezérlési lépésre

D. Vezérlő egység

A vezérlő egységet, amely az adiabatikus multiplexerek vezérlőjeleit szolgáltatja, szintén adiabatikus logikával valósítjuk meg. A vezérlő egység a 3.1.2. alfejezetben ismertetett tárolóhoz hasonlóan szintén gyűrűbe kapcsolt logikai kapukból épül fel, azonban most a gyűrűben lévő logikai kapuk száma megegyezik a vezérlő lépésben mért újraindítási idő értékével. Az adatútban lévő multiplexerek vezérlőbemeneteinek számától és az igényelt vezérléstől függően több ilyen gyűrűre is szükség lehet. A logikai kapuk egynegyede összetett kapu, amelyekkel induláskor a megfelelő logikai érték beállítása történik, míg a többi kapu puffer. A *3.24. ábrán*

található vezérlő egység felhasználható a *3.5. táblázat* szerinti ütemezés 12 vezérlési lépésenként történő pipeline-elvű vezérlésére.



3.24. ábra. Adiabatikus vezérlő egység 12 vezérlési lépésenként történő újraindítási időre

3.4.4. Eljárás négyfázisú adiabatikus áramkörök ütemezéséhez

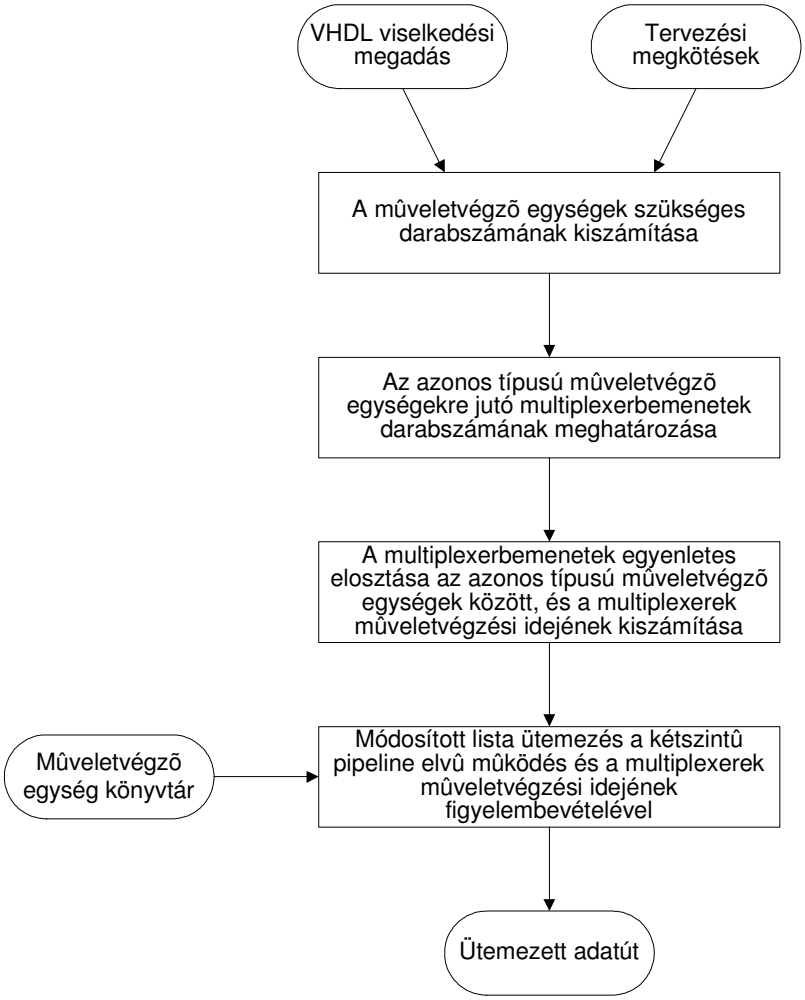
A legtöbb alkalmazásban a digitális jelfeldolgozó részrendszer számára a feldolgozás újraindítási ideje adott, és ehhez az újraindítási időhöz kell valamilyen szempontból optimális adatutat készíteni. Az előírt újraindítási idő sok esetben csak a pipeline elv alkalmazásával teljesíthető, ami különösen igaz az egyébként lassan működő adiabatikus áramkörökre. A lappangási idő ugyanakkor többnyire nem tartozik a fő tervezési megkötések közé.

A 2.5 illetve a 2.6. alfejezetekben láthattuk, hogy az ütemezés lehet végrehajtási idő korlátozott és/vagy erőforrás korlátozott; ugyanakkor kimutatták, hogy az esetek nagy többségében ütemezéskor az erőforrások számának megkötésével jobb eredményeket lehet elérni, mint a teljesítőképességre vonatkozó korlátokkal [9]. Indokolt volt ezért egy olyan eljárás kidolgozása, amely a megadott újraindítási idővel működő adatút ütemezését erőforrás korlátozott ütemezéssel biztosítja.

A négyfázisú adiabatikus áramkörök ütemezéséhez kidolgozott eljárás az azonos típusú műveletvégző egységek között, a hozzájuk csatlakozó multiplexerek bemeneteinek számának egyenletesen elosztásán alapul. Ahhoz, hogy az azonos típusú műveletvégző egységekhez tartozó multiplexer-bemenetek egyenletesen elosztott számát meghatározhassuk, a következő adatokra van szükségünk:

- a viselkedési leírás adott tervezési megkötésekkel történő megvalósításához szükséges műveletvégző egységek darabszámára minden egyes műveletvégző-egység típusból,
- az adatforrások forrástípusonkénti darabszámára, ahonnan az egyes típusú műveletvégző egységekhez bemenő adatok érkeznek.

Mindezen adatok a viselkedési leírásból és a tervezési megkötésekből nyerhetők a következőkben ismertetendő eljárás segítségével, amelynek fő lépéseit a *3.25. ábra* foglalja össze [S28].



3.25. ábra. A négyfázisú adiabatikus áramkörök ütemezéséhez kidolgozott eljárás fő lépései

A négyfázisú adiabatikus áramkörök ütemezéséhez kidolgozott eljárás először kiszámítja az előírt újraindítási idő eléréséhez szükséges műveletvégző egységek legkisebb darabszámát, anélkül, hogy a tulajdonképpeni ütemezést végrehajtaná.

A következő lépésben, felhasználva a műveletvégző egységek darabszámát és a viselkedési leírásban megadott műveletek közötti függőségeket, az azonos típusú műveletvégző egységekhez tartozó multiplexer-bemenetek darabszámának kiszámítása következik.

A harmadik lépésben az azonos típusú műveletvégző egységek, és a rájuk jutó multiplexer-bemenetek darabszámából az algoritmus műveletvégző-egység típusonként meghatározza az egy műveletvégző egységre jutó multiplexer-bemenetek számát, és így az egyes multiplexer-műveletek végrehajtásához szükséges vezérlési lépések darabszámát.

A negyedik lépés a tényleges ütemezést foglalja magában, ahol az algoritmus egy módosított lista ütemezéssel figyelembe veszi mind a multiplexer-műveletek végrehajtásához szükséges vezérlési lépések darabszámát, mind a kétszintű pipeline elv alkalmazásából adódó művelet átlapolásokat.

A négyfázisú adiabatikus áramkörök ütemezéséhez kidolgozott eljárás részletes ismertetése előtt bevezetek néhány jelölést, amelyeket a továbbiakban használok majd:

N_k	---	k típusú műveletek darabszáma a viselkedési leírásban
M_k	---	k típusú műveletet végrehajtó műveletvégző egységek darabszáma
DII	---	újraindítási idő vezérlési lépésben mérve
L	---	felső korlát az ütemezés hosszúságára vezérlési lépésben mérve
S_k	---	k típusú műveletvégző egységekhez csatlakozó multiplexerek bemeneteinek száma
I_k	---	egy k típusú műveletvégző egység egy bemenetére jutó multiplexer-bemenetek darabszáma
d_k	---	egy k típusú műveletvégző egység előtt elhelyezkedő multiplexerek műveletvégzési ideje vezérlési lépésben mérve
t_k	---	k típusú műveletvégző egység műveletvégzési ideje vezérlési lépésekben mérve
t'_k	---	k típusú műveletvégző egység ütemezéshez használt műveletvégzési ideje vezérlési lépésekben mérve

A. A műveletvégző egységek szükséges darabszámának kiszámítása

A fentebb bevezetett jelölésekkel élve a viselkedési megadás megvalósításához szükséges műveletvégző egységek legkisebb darabszáma műveletvégző-egység típusonként a (3.7) összefüggés szerint számítható ki, ahol a „⌈ ⌉” jel a következő egész számra való felfelé kerekítést jelenti.

$$\mathrm{M_k} = \left\lceil \frac{4 \cdot \mathrm{N_k}}{\mathrm{DII}} \right\rceil \tag{3.7}$$

B. Az azonos típusú műveletvégző egységekre jutó multiplexerbemenetek darabszámának meghatározása

Ebben a lépésben a különböző típusú műveletvégző egységek darabszámának függvényében műveletvégző-egység típusonként meghatározzuk a hozzájuk csatlakozó multiplexerek bemeneteinek darabszámát. A már bevezetett jelöléseket használva a keresett S_k értékek a viselkedési leírásban megadott műveletek közötti függőségek vizsgálatával kaphatók meg, a következő eljárást minden egyes k-ra elvégezve.

Kezdetben S_k egyenlő nullával. A kiinduló leírás alapján egy listát készítünk, amelyben a k típusú műveletekre összegyűjtjük azon különböző jellegű forrásokat, ahonnan a k típusú műveletekhez bemenő adat érkezik, valamint a különböző jellegű források mellett feltüntetjük a hozzá tartozó bemenetek előfordulási darabszámát is, amelyet E_a-val jelölünk, ahol a különbözteti meg az egyes forrásokat.

Az így elkészített listákban az egyes források jellege háromféle lehet, amelytől függően S_k a következőképpen változik: ha a forrás

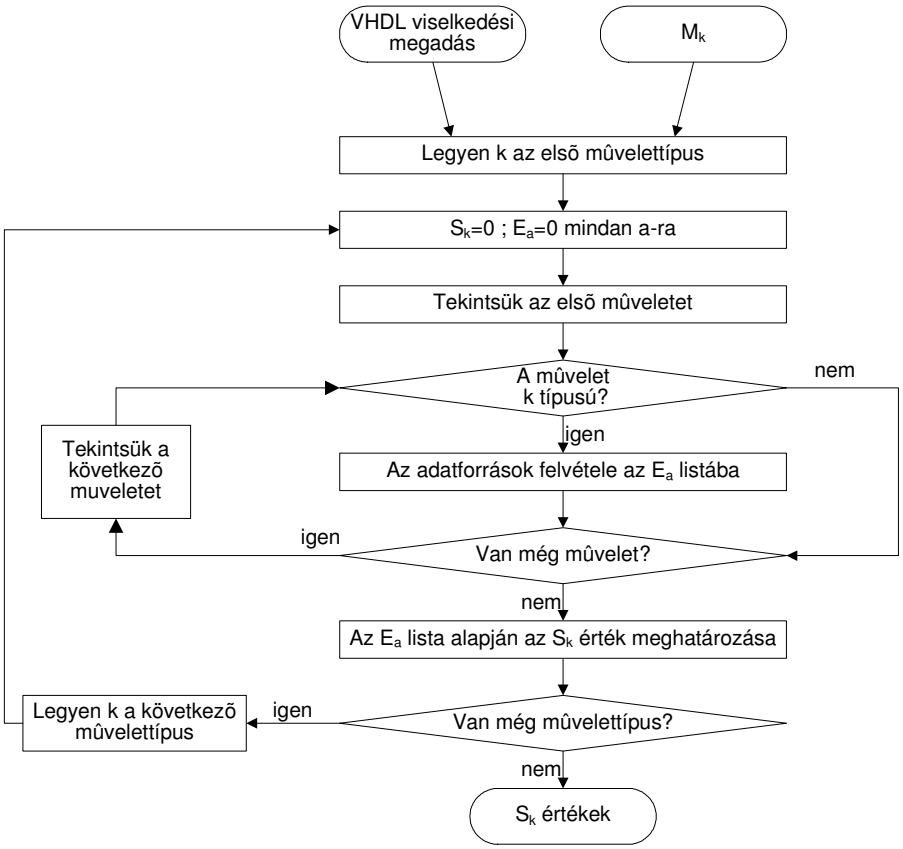
- bemeneti kapu, akkor $S_k=S_k+E_{bemeneti\ kapu}$,
- i típusú művelet, akkor $S_k=S_k+\min(E_{i\ típusú\ művelet}, M_i)$,
- állandó érték, akkor $S_k=S_k+\min(E_{állandó\ érték}, 2\cdot M_k)$ ha $S_k > M_k$, egyébként pedig S_k változatlan.

Ennek magyarázata a következő. A bemeneti kapukhoz szükséges multiplexer-bemenetek darabszáma a bemeneti kapuktól eredő bemenő csatlakozások számával egyenlő.

Ha egy k típusú művelethez bemenő adat egy i típusú művelet eredményeként érkezik, akkor az ehhez szükséges multiplexer-bemenetek száma attól függ, hogy az i típusú műveletektől a k típusú műveletekhez vezető csatlakozások száma és az i típusú műveletvégző egységek darabszáma hogyan viszonyul egymáshoz. Ha az előző a kisebb, a helyzet ugyanaz, mint a bemeneti kapuk esetén. Ha az utóbbi kisebb, akkor pedig csak az i típusú műveletvégző egységek darabszámával növekszik a szükséges multiplexer-bemenetek száma, mivel ekkor az i típusú műveletvégző egységeket megoszthatjuk az i típusú műveletek között.

Állandó érték kiválasztásához egy műveletvégző egységhez elég csak egyetlen multiplexer-bemenetet figyelembe venni az ütemezéskor, még akkor is, ha ezt az állandót több másik állandó közül választjuk ki, mivel az állandó értékek mindig rendelkezésre állnak, és így nem kell az adatfüggőségeket tekintetbe venni. Az éppen szükséges érték már előre kiválasztható úgy, hogy az adott vezérlési lépésben elérhető legyen. Ezt fejezi ki a $2\cdot M_k$ tag, ahol a kettes szorzó a későbbi kettővel való osztás miatt szükséges. Ha nem jut minden műveletvégző egységre állandó érték, akkor a szükséges multiplexer-bemenetek darabszáma csak az állandó értékek darabszámával növekszik. Abban a különleges esetben azonban, ha a bemeneti kapuktól és a többi típusú műveletvégző egységektől érkező bemenetek száma nem nagyobb mint a rendelkezésre álló műveletvégző egység száma, akkor az állandó értékek kiválasztásához az ütemezéskor a multiplexer-bemeneteket egyáltalán nem szükséges figyelembe venni. Ekkor ugyanis a nem állandó értékű bemenetek csak a műveletvégző egységek egyik bemenetéhez csatlakoznak, míg a másik bemenetekre tetszőleges számú állandó az adott vezérlési lépésre kiválasztható.

A műveletvégző egységekhez csatlakozó multiplexerek bemeneteinek darabszámát műveletvégző-egység típusonként meghatározó eljárást a *3.26. ábra* foglalja össze, amely az eljárással az S_k értékek megkaphatóak minden k-ra.



3.26. ábra. A műveletvégző egységekhez csatlakozó multiplexer-bemenetek darabszámának műveletvégző-egység típusonkénti kiszámítása

C. A multiplexerbemenetek egyenletes elosztása az azonos típusú műveletvégző egységek között és a multiplexerek műveletvégzési idejének kiszámítása

Mivel minden műveletvégző egység két adatbemenettel rendelkezik, és a *k* típusú műveletvégző egységekből M_k darab van, ezért az egyes műveletvégző egységek egyes bemeneteire jutó multiplexer-bemenetek egyenletesen elosztott darabszámát a (3.8) összefüggés adja.

$$I_k = \left\lceil \frac{S_k}{2 \cdot M_k} \right\rceil \tag{3.8}$$

A (3.8) összefüggés az azonos típusú műveletvégző egységekhez csatlakozó multiplexerek bemeneteinek száma között egyenletes eloszlást feltételez, amelynek teljesítése az erőforrás-kiosztás feladata lesz. Egy *k* típusú műveletvégző egységekhez csatlakozó multiplexerek műveletvégzési ideje vezérlési lépésben mérve kiszámítható a (3.9) összefüggés ismeretében, ahol egy logikai szinten legfeljebb négy bemenetű multiplexereket engedélyezünk.

$$d_k = \left\lceil \frac{\log_2(I_k)}{2} \right\rceil \tag{3.9}$$

Ütemezéskor egy műveletvégző egységet és a hozzá csatlakozó multiplexert együtt úgy tekinthetjük, mintha a műveletvégző egység műveletvégzési ideje a hozzá csatlakozó multiplexer műveletvégzési idejével megnőtt volna. Ezt fejezi ki a (3.10) összefüggés, ahol a t'_k jelenti a tényleges ütemezéshez használandó, vezérlési lépésben mért műveletvégzési időt.

$$t'_k = t_k + d_k \tag{3.10}$$

D. Módosított lista ütemezés a kétszintű pipeline-elvű működés és a multiplexerek műveletvégzési idejének figyelembevételével

A tényleges ütemezési lépéshez egy módosított lista ütemezést dolgoztam ki, amely képes a második szintű pipeline-elven működő műveletvégző egységeket ütemezni. Először meghatározzuk minden egyes művelet mozgékonyságát az *ASAP* (olyan hamar, amint lehetséges) és az *ALAP* (olyan későn, amint lehetséges) indítási értékek kiszámolásával, műveletvégzési időnek a t'_k értékeket használva. Az *ALAP* ütemezéshez egy felső korlát szükséges az ütemezés L hosszúságára, amely lehet paraméter, vagy az algoritmus választ egy megfelelően nagy értéket. Ha paraméterként megadjuk a megengedhető legnagyobb ütemezési hosszúságot, akkor előfordulhat, hogy az a rendelkezésre álló erőforrásokkal nem teljesíthető. Amennyiben nem írjuk elő a megengedhető legnagyobb ütemezési hosszúságot, akkor ebből a szempontból nem fontos, hogy L ténylegesen milyen értéket kap, így – negatív mozgékonyságot eredményezve – lehet akár kisebb is mint, az *ASAP* ütemezési hossz, mivel a mozgékonyság értékét csak a műveletek sorrendbe állítására használjuk.

A módosított lista ütemező a műveletvégző egységek minden típusára egy foglalási táblát tart fenn, amely táblának M_k darab sora van, oszlopainak száma pedig megegyezik a vezérlési lépésben megadott megengedhető legnagyobb ütemezési hosszúsággal. Ha a legnagyobb ütemezési hosszúság paraméterként nem adott, akkor itt a legnagyobb lehetséges érték alkalmazása szükséges, mivel a táblák dinamikus bővítése elbonyolítaná az algoritmust.

Egy művelet kezdete csak akkor ütemezhető a t vezérlési lépésbe, ha az adott művelethez tartozó foglalási tábla t sorszámú oszlopában még van üres hely. Ha egy művelet kezdetét a t vezérlési lépésbe ütemezzük, akkor az ütemező egy jelet tesz az adott művelethez tartozó foglalási tábla még üres sorának mindazon oszlopaiba, amelynek sorszámára igaz a (3.11) összefüggés, ahol C_k jelöli a k típusú művelethez tartozó foglalási tábla oszlopának sorszámát, n pedig egész szám és $0 < n < L/DII$.

$$C_k = t + n \cdot DII \tag{3.11}$$

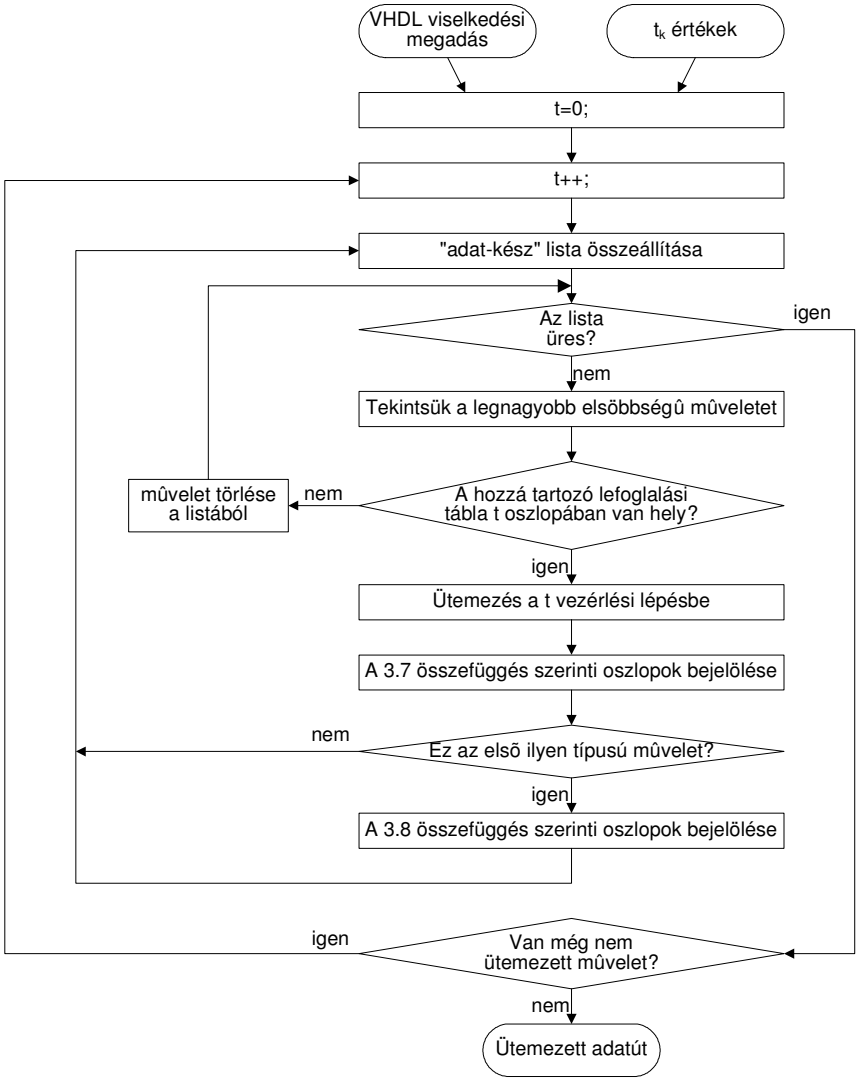
Ha egy k típusú műveletet első alkalommal ütemezünk, akkor az ütemező – az előzőeken túlmenően – a k típusú művelethez tartozó foglalási tábla minden olyan oszlopának minden sorába egy-egy jelet tesz, amelyre a (3.12) összefüggés igaz.

$$C_k \neq t + 4 \cdot n \tag{3.12}$$

A (3.11) összefüggés szerinti foglalás a pipeline-elvű működés biztosításához szükséges, hiszen a lista ütemezés eredetileg erre nem alkalmas, míg a

(3.12) összefüggés szerinti jelölés a foglalási táblában a műveletvégző egységeknek a műveletek közötti megosztását teszi lehetővé, és ezzel biztosítja, hogy az ütemezés a (3.7) összefüggéssel meghatározott darabszámú műveletvégző egységekkel megvalósítható legyen.

Egy hagyományos ütemező algoritmusban az adott típusú műveletvégző egységek az ugyanolyan típusú műveletek között megoszthatók, ha ezen műveletek időben nem lapolják át egymást. Adiabatikus műveletvégző egységek használatakor a második szintű pipeline lehetővé teszi az átlapolást, azonban az előbbi feltételen túl az erőforrás-megosztáshoz a négy fázisjeles vezérlés miatt még az is szükséges, hogy a megosztandó műveletvégző egységeken végrehajtandó műveletek azonos fázisjel alatt induljanak. A kidogozott ütemező algoritmus ezt úgy biztosítja, hogy az azonos típusú műveletvégző egységek ugyanazon logikai szintjeit ugyanazon fázisjelhez rendeli. Az azonos típusú műveletvégző egységek ugyanazon logikai szintjeinek ugyanazon fázisjelekhez való kötése ugyan átfogóan nem a legkedvezőbb ütemezést eredményezi, azonban ettől eltérni csak az erőforrás kiosztás ütemezésbe történő beépítésével lehetne. A tényleges ütemezési lépést a *3.27. ábra* foglalja össze.



3.27. ábra. Módosított lista ütemezés négyfázisú adiabatikus áramkörökhöz

3.4.5. Egészértékű lineáris programozáson alapuló algoritmus négyfázisú adiabatikus áramkörök ütemezéséhez

Az előző alpontban bemutatott ütemező eljárás ugyan optimális eredményt ad a szükséges műveletvégző egységek darabszámára nézve, azonban az ütemezés vezérlési lépésekben mért hosszúsága általánosságban nem a legkedvezőbb, így a szükséges járulékos elemek, mint például a pufferek száma több lehet a kelleténél. Ezért kidolgoztam egy egészértékű lineáris programozáson alapuló algoritmust is, amely tetszőleges szempontból optimális megoldást nyújt [S30].

Az egészértékű lineáris programozáson alapuló ütemezés lineáris egyenlőségek és egyenlőtlenségek felírását jelenti, amelyek leírják mind a viselkedési megadás műveleteinek egymástól való függését, mind a tervezési korlátokat. A tényleges ütemezés ezen egyenlőségek és egyenlőtlenségek megoldását jelenti úgy, hogy a megoldás mellett valamely célfüggvény értéke a legnagyobb vagy a legkisebb legyen.

A következő táblázatban néhány jelölést adok meg, amelyeket – az előző alpontban bevezetett jelöléseken felül – a továbbiakban használni fogok, valamint néhány korábban bevezetett jelölésnek most új értelmet adok:

o_i	---	az *i.* művelet a kiinduló leírásban
s_i	---	az o_i művelet legkorábbi (olyan hamar, amint lehetséges) indítási ideje
l_i	---	az o_i művelet legkésőbbi (olyan későn, amint lehetséges) indítási ideje
$x_{i,t}$	---	értéke egy ha az o_i művelet a *t.* vezérlési lépésben indul, egyébként nulla
t_i	---	az o_i művelet végrehajtásához szükséges vezérlési lépések száma
d_i	---	az o_i műveletet végrehajtó műveletvégző egység előtt elhelyezkedő multiplexer műveletvégzési ideje vezérlési lépésben mérve

A. Ütemezési korlátok

Valamennyi művelet végrehajtását a rá vonatkozó legkorábbi és a legkésőbbi indítási idők között kell megkezdeni. Mivel az egyes műveleteket végrehajtó műveletvégző egységek előtt elhelyezkedő multiplexerek műveletvégzési idejét nem tudjuk előre megmondani, ezért az egyes műveletek indítási ablakát csak a hagyományos esetre tudjuk meghatározni, vagyis amikor a multiplexerek műveletvégzési idejét nem vesszük számításba. Az egyes műveletek indítási ablakainak kezdetét és végét az olyan hamar, amint lehetséges (*ASAP*), illetve az olyan későn, amint lehetséges (*ALAP*) ütemezésekkel határozhatjuk meg. Ha a multiplexereket számításba tudnánk venni, akkor az egyes indítási ablakok szűkebbek lennének, azonban az *ASAP* és az *ALAP* ütemezésre kapott értékeket így is felhasználhatjuk arra, hogy az egyenletekből elhagyhassunk nyilvánvalóan nulla értékű változókat, csökkentve ezáltal a számítás bonyolultságát. Így a (3.13) szerinti egyenletek írhatók.

$$\sum_{t=s_i}^{l_i} x_{i,t} = 1 \qquad \text{minden egyes } i \text{ - re.} \tag{3.13}$$

B. Függőségi korlátok

Egy műveletvégző egység a bemenő adatait multiplexereken keresztül kapja, hogy aztán végrehajtsa rajtuk az általa ismert műveletet. Amint arról már korábban szó volt, egy multiplexer műveletvégzési ideje legalább egy vezérlési lépés (vagy a bemenetek számától függően több), és ezt a késleltetést egy megvalósítható adatút készítéséhez ütemezéskor figyelembe kell venni. Azért, hogy a műveletek közötti függőségeket a kiinduló leírásnak megfelelően megtartsuk, egy művelet végrehajtása csak akkor kezdődhet, ha az azt megelőző műveletek már befejeződtek. A fentieknek megfelelő egyenletek (3.14) szerintiek.

$$\sum_{t=s_i}^{l_i} t * x_{i,t} + t_i + d_i - \sum_{t=s_j}^{l_j} t * x_{i,j} \le 0$$

minden egyes *i*-re és *j*-re, ahol o_i közvetlenül megelőzi o_j-t,

$$\sum_{t=s_i}^{l_i} t * x_{i,t} + t_i + d_i - L \le 1 \qquad (3.14)$$

minden egyes *i*-re, ahol o_i-t nem követi további művelet.

C. Pipeline átlapolási korlátok

Egy műveletvégző egység, amely egy számítást vezérlési lépésben kifejezve a *t* időpontban kezd meg, pipeline elvű működésének köszönhetően a *t+4* időpontban újabb számításba kezdhet anélkül, hogy az előző műveletet befejezte volna. Egy művelet, amely vezérlési lépésben kifejezve a *T* időpontban indul, lefoglal egy műveletvégző egységet $T \le t < T+4$ ideig. Ez a (3.15) szerinti függvényekkel fejezhető ki, amelyet minden egyes műveletre (minden *i*-re) fel kell írni:

$$\begin{aligned} x'_{i,t} &= 1 \quad \text{ha } T + d_i \le t < T + d_i + 4 \\ &= 0 \quad \text{egyébként.} \end{aligned} \qquad (3.15)$$

A párhuzamosan használt *k* típusú műveletvégző egységek darabszáma a vezérlési lépésben kifejezett *t* időpontban egyenlő azon műveletek darabszámával, amelyek ugyanezen *t* időpontban *k* típusú műveletvégző egységet lefoglalnak. A *k* típusú műveletvégző egységek szükséges darabszáma egyenlő a bármely időpontban párhuzamosan használt *k* típusú műveletvégző egységek darabszámának legnagyobbikával. Ezt fejezik ki a (3.16) függvények.

$$M_k = \max \sum_{t=1}^{L/DII} x'_{i,j+t*DII} \qquad (3.16)$$

minden i,j és k-ra, ahol $0<j\le DII$ és o_i *k* típusú

D. Multiplexer korlátok

A multiplexer korlátok felírásához meg kell határozni az egyes típusú műveletvégző egységek darabszámának függvényében a különböző típusú műveletvégző egységekhez csatlakozó multiplexerek bemeneteinek számát. Ez a *3.26 ábrán* (lásd az előző alfejezetben) ismertetett algoritmus szerinti S_k értékek kiszámítását jelenti az egyes típusú műveletvégző egységek darabszámának függvényében. Egy műveletvégző egység két bemenettel rendelkezik, így a szükséges multiplexer bemenetek száma megoszlik a két bemenet között, ezen kívül a *k* típusú műveletvégző egységből M_k darab van. Így a 3.17 egyenlőtlenségeknek kell teljesülniük.

$$\frac{S_k}{2} \le \sum_{o_i} d_i \le \sum_{k=1}^{M_k} 4^{d_k} \qquad \text{ahol } o_i \textit{ k} \text{ típusú} \tag{3.17}$$

3.4.6. Ütemezési algoritmusok eredményei

A kidolgozott eljárással kapott eredményeket egy példa kíséretében szemléltetem, ahol tervezési példának a magas szintű logikai szintézisben jól ismert véges válaszú szűrőt választottam. A véges válaszú szűrő egy lehetséges (és általam használt) viselkedési megadása *3.9. listán* látható.

```
entity FIR_filter is
   port (px0, px1, px2, px3, px4, px5,px6,
         px7, px8 : in integer;
         py : out integer;
         start : in bit;
         ready : out bit);
end FIR_filter;
architecture HBM of FIR_filter is
begin process
variable x0, x1, x2, x3, x4 : integer;
variable x5, x6, x7, x8 : integer;
variable vo1, vo2, vo3, vo4, vo5 : integer;
variable vo6, vo7, vo8, vs1, vs2 : integer;
variable vs3, vs4, vs5, vs6, vs7 : integer;
variable vs8, vo10, vo11, vo12, vo13 : integer;
variable vo14, vo15 : integer;
constant w1, w2, w3, w4, w5 : integer;
constant w6, w7, w8 : integer;

begin
   wait until start = '1';
   ready <= '0';
   x0 := px0; x1 := px1; x2 := px2; x3 := px3;
   x4 := px4; x5 := px5; x6 := px6; x7 := px7;
   x8 := px8;
   wait until start = '0';
   vo1 := x0 + x1;
   vo2 := x1 + x2;
   vo3 := x2 + x3;
   vo4 := x3 + x4;
   vo5 := x4 + x5;
   vo6 := x5 + x6;
   vo7 := x6 + x7;
```

```
    vo8 := x7 + x8;
    vs1 := w0 * vo1;
    vs2 := w1 * vo2;
    vs3 := w2 * vo3;
    vs4 := w3 * vo4;
    vs5 := w4 * vo5;
    vs6 := w5 * vo6;
    vs7 := w6 * vo7;
    vs8 := w7 * vo8;
    vo9 := vs1 + vs2;
    vo10 := vs3 + vs4;
    vo11 := vs5 + vs6;
    vo12 := vs7 + vs8;
    vo13 := vo9 + vo10;
    vo14 := vo11 + vo12;
    vo15 := vo13 + vo14;
    py <= vo15;
    ready <= '1';
end process;
end HBM;
```

3.9. lista. A véges válaszú szűrő tervezési példa egy lehetséges viselkedési megadása

A. A műveletvégző egységek szükséges darabszámának kiszámítása

A véges válaszú szűrő viselkedési leírásában 15 darab összeadás és 8 darab szorzás művelet található, tehát $N_+=15$ és $N_*=8$. A (3.7) összefüggést felhasználva az újraindítási idő különböző értékeire az összeadó, illetve a szorzó műveletvégző egységek szükséges darabszámára a *3.12. táblázatban* (lásd később) megadott értékeket kapjuk.

B. Az azonos típusú műveletvégző egységekre jutó multiplexerbemenetek darabszámának meghatározása

A 3.4.4. alfejezeben láttuk, hogy az azonos típusú műveletvégző egységekre jutó multiplexerbemenetek darabszámának meghatározásához először minden egyes típusú műveletre össze kell gyűjtenünk azon különböző jellegű forrásokat, ahonnan az éppen vizsgált típusú műveletekhez bemenő adat érkezik, továbbá a különböző jellegű adatforrások mellett össze kell számolnunk a hozzá tartozó bemenetek előfordulási darabszámát is, amelyet E_a-val jelöltünk. A véges válaszú szűrő tervezési példa esetén az összeadás, illetve szorzás műveletekre a keresett E_a értékeket a *3.6.*, illetve a *3.7. táblázat* tartalmazza.

Forrás jellege	Darabszáma
$E_{bemeneti\ kapu}$	16
$E_{szorzás\ művelet}$	8
$E_{összeadás\ művelet}$	6

3.6. táblázat. A véges válaszú szűrő tervezési példában az összeadás műveletek különböző jellegű adatforrásai és ezek darabszámai

Forrás jellege	Darabszáma
$E_{összeadás\ művelet}$	8
$E_{állandó\ érték}$	8

3.7. táblázat. A véges válaszú szűrő tervezési példában a szorzás műveletek különböző jellegű adatforrásai és ezek darabszámai

Az egyes típusú műveletvégző egységekre jutó multiplexerbemenetek darabszámát a *3.26. ábrán* szereplő algoritmus lefutatásával kaphatjuk meg. A véges válaszú szűrő tervezési példára kapott S_k értékeket a *3.8.*, illetve a *3.9. táblázatban* találhatjuk (lásd alább).

C. A multiplexerbemenetek egyenletes elosztása az azonos típusú műveletvégző egységek között és a multiplexerek műveletvégzési idejének kiszámítása

A véges válaszú szűrő tervezési példa esetén a *k* típusú műveletvégző egységekhez csatlakozó multiplexerek bemeneteinek számát (S_k), továbbá a (3.8) összefüggés felhasználásával kapott, a *k* típusú műveletvégző egység egy bemenetére jutó multiplexer-bemenetek darabszámát (I_k), valamint a (3.9) összefüggés segítségével nyert, a *k* típusú műveletvégző egység előtt elhelyezkedő multiplexerek vezérlési lépésben megadott műveletvégzési idejét (d_k) különböző újraindítási időkre (*DII*) a *3.8.*, illetve a *3.9. táblázat* ismertet, összeadókra, illetve szorzókra nézve.

DII	$S_{összeadó}$	$I_{összeadó}$	$d_{összeadó}$
4	30	1	0
8	26	2	1
12	24	3	1

3.8. táblázat. A véges válaszú szűrő tervezési példában az összedók $S_{összeadó}$, $I_{összeadó}$ és $d_{összeadó}$ értékei különböző újraindítási idők mellett

DII	$S_{szorzó}$	$I_{szorzó}$	$d_{szorzó}$
4	8	1	0
8	16	2	1
12	11	2	1

3.9. táblázat. A véges válaszú szűrő tervezési példában a szorzók $S_{szorzó}$, $I_{szorzó}$ és $d_{szorzó}$ értékei különböző újraindítási idők mellett

D. A módosított lista ütemezés adatai a kétszintű pipeline-elvű működés és a multiplexerek műveletvégzési idejének figyelembevételével

A véges válaszú szűrő tervezési példát a kidolgozott módosított lista ütemezéssel ütemezve az összeadókra kapott foglalási táblát a *3.10. táblázatban* láthatjuk, ahol az ütemezést *DII=12* vezérlési lépésenként történő újraindítási idővel és 16 bit széles adatúttal végeztem. A 16 bit széles adatút mellett az adiabatikus összeadó műveletvégzési ideje 6, a szorzóé pedig 9 vezérlési lépésre adódik. A *3.10. táblázatból* helytakarékossági okból lehagytam a foglalási tábla azon oszlopait, amelyre a (3.12) összefüggés igazat ad. A *3.11. táblázatban* ugyanerre az ütemezésre a szorzókra vonatkozó foglalási táblát tüntettem fel.

vez. lépés	2	6	10	14	18	22	26	30	34	38	42	46	50	54	58	62	66	70
1. összeadó	1	6		1	6	9	1	6	9	1	6	9	1	6	9	1	6	9
2. összeadó	2	7		2	7		2	7	12	2	7	12	2	7	12	2	7	12
3. összeadó	3	8		3	8		3	8		3	8	13	3	8	13	3	8	13
4. összeadó	4			4			4	10		4	10	14	4	10	14	4	10	14
5. összeadó	5			5			5	11		5	11		5	11	15	5	11	15

3.10. táblázat. Az összeadókra vonatkozó foglalási tábla a véges válaszú szűrő tervezési példa *DII*=12-vel történő ütemezésekor

vez. lépés	1	5	9	13	17	21	25	29	33	37	41	45	49	53	57	61
1. szorzó			1	4	7	1	4	7	1	4	7	1	4	7	1	4
2. szorzó			2	5	8	2	5	8	2	5	8	2	5	8	2	5
3. szorzó			3	6		3	6		3	6		3	6		3	6

3.11. táblázat. A szorzókra vonatkozó foglalási tábla a véges válaszú szűrő tervezési példa *DII*=12-vel történő ütemezésekor

A *3.12. táblázat* a kidolgozott módosított lista ütemező algoritmussal kapott ütemezési eredményeket mutatja a véges válaszú szűrő tervezési példára. Az adatút szélességét minden esetben 16-bitre választottam, az újraindítási időket pedig 4 és 12 vezérlési lépés között változtattam. A táblázat oszlopai az újraindítási idő vezérlési lépésben mérve, az összeadók darabszáma, a szorzók darabszáma, a multiplexer bemenetek száma, a pufferek száma és az ütemezési hosszúság vezérlési lépésben kifejezve.

DII	M_+	M_*	*Mpx. bem.*	*Puffer*	*L*
4	15	8	0	20	38
8	8	4	36	70	51
12	5	3	42	142	63

3.12. táblázat. A kidolgozott, módosított lista ütemezést alkalmazó ütemező eljárás eredményei a véges válaszú szűrő tervezési példára, különböző újraindítási idők mellett

E. Az egészértékű lineáris programozáson alapuló ütemező eljárás eredményei

A véges válaszú szűrő tervezési példára felírva a 3.4.5. alfejezeben bemutatott ütemezési, függőségi, pipeline átlapolási és multiplexer korlátokat leíró egyenleteket és egyenlőtlenségeket, valamint ezeket az ütemezési hosszúság minimalizálása mellett megoldva a különböző újraindítási időkre a *3.13 táblázatban* látható eredményeket kapjuk.

DII	M_+	M_*	*Mpx. bem.*	*Puffer*	*L*
4	15	8	0	20	38
8	8	4	36	34	43
12	5	3	42	81	57

3.13 táblázat. Az egészértékű lineáris programozáson alapuló ütemező eljárás eredményei különböző újraindítási idők mellett.

A *3.12* és a *3.13 táblázatokból* kiolvasható, hogy mind a módosított lista ütemező algoritmus, mind az egészértékű lineáris programozáson alapuló ütemező eljárás azonos és egyben a legkisebb szükséges számú műveletvégző egységet alkalmazza, azonban a módosított lista ütemezés használatával ütemezési hosszúság (vagyis az ütemezett adatút lappangási ideje) nagyobb és a szükséges pufferek száma is több. Ennek az a magyarázata, hogy ez az ütemezés az azonos típusú műveletek kezdetét azonos fázisjelhez köti.

A felhasznált pufferek száma esetenként csökkenthető, ha az átmeneti adattárolást nem a pufferek (hosszú) láncba-kapcsolásával, hanem a *3.13. ábra* szerinti négy logikai-kapus gyűrűvel felépített tárolóelemmel kialakítva valósítjuk meg, ekkor azonban a tárolóelem vezérléséről gondoskodni kell.

3.4.7. VHDL modellezési technika a szimulációhoz

Az integrált áramkörök tervezése során felmerülő igen fontos feladat az, hogy ellenőrizni tudjuk, vajon a terv megfelel-e az eredeti megadásnak. Az áramkör megadása egy hardverleíró nyelven és ennek a kódnak szimulációval való ellenőrzése általánosan elfogadott ipari gyakorlat. A tervet rendszerint algoritmus szinten adják meg, majd finomítják a regiszter-átviteli szint felé. A különböző digitális áramkörök leírására szolgáló hardverleíró nyelvek közül a VHDL a legszélesebb körben használt szabványosított nyelv. A VHDL nyelvvel a terv számos elvonatkoztatási szinten leírható, így jól ábrázolható vele mind a viselkedési rendszermegadás, mind a regiszter-átviteli szintű terv.

A viselkedési szintű terv finomítása során, meghatározva az áramkör órajel-ciklusokhoz kötött viselkedését, és az ok-okozati időzítésről áttérve az órajelhez kötött időzítésre lehetőség nyílik a késleltetési időkkel figyelembe vett teljesítőképesség szimulációra. Átfogó teljesítőképesség adatokhoz rendszerint a teljes rendszer szimulációjára van szükség. Az adiabatikus adatútnak a digitális rendszer többi részével történő együtt-szimulálásához egy olyan modellezési technikára van szükség, amely hűen leírja az adiabatikus logika fázisjel-vezérelt viselkedését. Bár a szükséges modell órajelhez kötött időzítésen alapul, azonban elfogadható szimulációs időtartam eléréséhez még mindig viselkedési modellre van szükség.

A kidolgozott modellezési technika alapján az adatútban található minden egyes összetevőt egy VHDL egyed ír le [S27]. Az egyedmegadó rész fejléce leírja az egyed nevét, valamint a ki- és a bemeneti kapuit, míg a törzse a működését és az időzítési viszonyokat tartalmazza. A *3.10. listán* egy 16 bites adiabatikus összeadó VHDL modelljének egy töredéke látható, ahol a rövidség kedvéért csak az összeadó két legkisebb helyiértékű bitjét ábrázoltam.

```
entity adder16 is
port (pa, pb: in std_logic_vector(0 to 15);
      pf1, pf2, pf3, pf4 : in std_logic;
      py: out std_logic_vector(0 to 16));
end adder16;
architecture DFB of adder16 is
signal P1, G1, P2, G2, P3, G3, P4, G4, P5, G5,
           P6, G6 : std_logic_vector(0 to 15);
begin
 dfb1:block (pf1='1') begin
   P1(0)<=guarded pa(0) xor pb(0);
   G1(0)<=guarded pa(0) and pb(0);
   P1(1)<=guarded pa(1) xor pb(1);
   G1(1)<=guarded pa(1) and pb(1);
 end block;
 dfb2:block (pf2='1') begin
   P2(0)<=guarded P1(0);
   G2(0)<=guarded G1(0);
   P2(1)<=guarded P1(1);
   G2(1)<=guarded (G1(0) and P1(1)) xor G1(1);
 end block;
 dfb3:block (pf3='1') begin
   P3(0)<=guarded P2(0);
   G3(0)<=guarded G2(0);
```

```
  P3(1)<=guarded P2(1);
  G3(1)<=guarded G2(1);
end block;
dfb4:block (pf4='1') begin
  P4(0)<=guarded P3(0);
  G4(0)<=guarded G3(0);
  P4(1)<=guarded P3(1);
  G4(1)<=guarded G3(1);
end block;
dfb5:block (pf1='1') begin
  P5(0)<=guarded P4(0);
  G5(0)<=guarded G4(0);
  P5(1)<=guarded P4(1);
  G5(1)<=guarded G4(1);
end block;
dfb6:block (pf2='1') begin
  py(0)<=guarded P5(0);
  py(1)<=guarded G5(0) xor P5(1);
end block; end DFB;
```

3.10 lista. 16-bites adiabatikus összeadó VHDL modelljének egy töredéke.

A *pf1*, *pf2*, *pf3* és a *pf4* jelek a fázisjelek, amelyek az adatáramlást vezérlik az egymás után kapcsolt logikai szintek között. Az egyes logikai szinteken végrehajtott logikai függvényeket a VHDL tömbökben adom meg. Ezek a tömbök úgynevezett őrzött tömbök, amelyeken belül őrzött együttfutó értékadások szerepelnek. Ezen őrzött együttfutó értékadások írják le az adatfüggőséget az egyes logikai szintek között. Az értékadások akkor hajtódnak végre, ha az őrző kifejezés igaz értékűvé válik, vagy ha az őrző kifejezés igaz értéke alatt a kifejezés jobb oldalán álló jelek valamelyikén esemény történik. Az őrző kifejezést a megfelelő fázisjel magas értéke teszi igazzá. Mindez amellett, hogy hűen tükrözi az adiabatikus logika fázisjel-vezérelt viselkedését, jól leírja azt is, hogy bármely bemeneti jelváltozás a fázisjel magas szintje alatt hibás működést eredményez.

Az adatút a 3.10 listán látható összeadó-részlet felépítéséhez hasonló szerkezetű összetevők beültetéséből épül fel, amely összetevőket az úgynevezett összetevő-könyvtárban tárolunk. A fázisjeleket – a megfelelő sorrendben – minden egyes összetevőhöz bekötjük. Az összetevők be- és kimeneti kapuit jeleken keresztül kötjük össze, amely jelek felhasználhatók az áramkör belső pontjainak figyelésére is.

4. Összefoglalás

Az integrált áramkörtervezés manapság egyik legfontosabb kérdése az áramkör teljesítményfelvétele és ezzel együtt a melegedése. A statikus CMOS kapcsolások teljesítményfelvételének csökkentésére irányuló eljárások azonban nem hoztak átütő eredményt, továbbá ezek alkalmazhatóságát számos tényező korlátozza. A teljesítményfelvétel nagyságrendekkel való csökkentéséhez alapvetően új megoldásra van szükség, egy ilyen lehetőség az utóbbi években előtérbe került adiabatikus elven működő áramkörök alkalmazása.

Sajnos az adiabatikus elven működő áramkörök alkalmazhatóságát döntően behatárolja az, hogy ezen áramkörök az energiahatékonyságukat a működési sebesség rovására érik el. Ezért az adiabatikus áramkörök elsősorban olyan esetekben használhatók előnyösen, ahol a kis fogyasztás sokkal fontosabb követelmény, mint a nagy működési sebesség. Ilyen alkalmazásokra példák a hordozható, telepes berendezések, orvosi műszerek, emberi testbe beültetett gyógyászati szerkezetek, telemetriás rendszerek, intelligens telepes érzékelők és adatgyűjtők, stb.

A fenti probléma azonban mérsékelhető egyszerűbb felépítésű, nagyobb működési frekvenciákon is hatékony adiabatikus kapcsolásokkal, ezáltal az adiabatikus elven működő áramkörök alkalmazási területe növelhető. Mindezeken túl, az adiabatikus kapcsolások a statikus kapcsolásokhoz képest számos további előnnyel rendelkeznek, amelyek bizonyos esetekben döntőek is lehetnek az adiabatikus kapcsolások alkalmazására nézve.

Az egyik ilyen fontos tényező a keverten analóg és digitális jeleket is használó integrált áramkörökben teszi megfontolás tárgyává az adiabatikus áramkörök alkalmazását [S25]. Az ilyen integrált áramkörök tervezésekor a digitális részből az analóg részbe történő zajbecsatolás csökkentése komoly tervezési feladatot jelent, amely nagy odafigyelést és különleges technikák alkalmazását kívánja meg. Mivel az adiabatikus kapcsolások szinuszos jelekkel működnek, előnyösek a zajbecsatolás csökkentésében, mert nem keltenek olyan nagyfrekvenciás harmonikus zajokat, mint a négyszögjellel működő statikus társaik.

A másik nagy előnye az adiabatikus áramköröknek, hogy a felépítésükből adódó pipeline-elvű működésnek köszönhetően átbocsátásuk a logikai mélységtől független. Ez az előny különösen a digitális jelfeldolgozásban jelentkezik, ahol a kétszintű pipeline működés előnyösen kihasználható az átbocsátás növeléséhez anélkül, hogy a statikus kapcsolásokban szokásosan alkalmazott külön pipeline regiszterekre szükség lenne. Az átbocsátás növelése mellett a lappangási idő is alacsonyan tartható négyfázisú kapcsolás alkalmazásával, hiszen – szemben a kettő- vagy az egyfázisú kapcsolásokkal – itt egy órajel-ciklus alatt négy logikai fokozat kiértékelése történik meg.

További előnyei még az adiabatikus áramköröknek, hogy tervezésükkor nem jelentkeznek olyan nehézségek, amelyek kiküszöbölésére a statikus kapcsolásokban különös gondot kell fordítani. Ilyen például a hazárd, amely az adiabatikus logikákban ismeretlen, hiszen a logikai fokozatok vezérelt kiértékelése nem teszi lehetővé semelyik két jel között sem a versenyhelyzet kialakulását. Ugyancsak nem jelent gondot az egyidejű jelváltozás, annak ellenére, hogy tulajdonképpen mindig minden

jel egyidejűleg változik, mivel a rezonáns tápegység által felvett áram lényegesen kisebb az adiabatikus logikába befolyó, a terhelő kapacitásokat töltő áramnál. A rezonáns tápegység ugyanis csak a rendszer veszteségének megfelelő energia betáplálását igényli, és ez a veszteség – éppen az áramkör adiabatikus természetéből adódóan – igen kicsi.

Mindezeken kívül említést érdemel még, hogy az adiabatikus áramkörök elkészítése nem igényel a statikustól eltérő technológiát, ezáltal ugyanaz a gyártási folyamat használható mindkettőre, egy félvezetőlapkán a statikus és az adiabatikus áramkörök akár keverten is megvalósíthatók.

Az integrált áramkörtervezésnek a teljesítményfelvétel mellett a másik fontos kérdése a technológia adta alkatrészsűrűség és bonyolultság kihasználása. A regiszter-átviteli szinten történő tervezéssel a termelékenység már a meglévő tervezési összetevők újbóli felhasználásával sem növelhető tovább. A technológia kínálta lehetőségek és az áramkörtervezés eredményessége közötti szakadék csökkentéséhez a tervezésben használt elvonatkoztatási szint emelése elkerülhetetlen.

Mindemellett nagy jelentőséggel bír, és a jelen gyakorlatban a tervezés egyik legidőigényesebb részfeladata a valamilyen formában megadott terv érvényesítése. Egy magasabb elvonatkoztatási szinten megkezdett tervezés, ahogy a szilíciumon lévő rajzolat kialakítása felé halad, számos elvonatkoztatási szinten megy keresztül. A helyes működés ellenőrzése ezért az egyes elvonatkoztatási szinteken döntő fontosságú.

Manapság a rendszertervezés egyre inkább a hardver és a szoftver összetevők együttes tervezését jelenti, amelyhez jól meghatározott tervezési folyamatat szükséges. Ahogy nő ezen rendszerek összetettsége, egyre nagyobb az igény a hatékony rendszerszintű tervezési eljárások kidolgozására.

A korszerű hardverleíró nyelvek az áramköri rendszerek összetettségét magas szintű leírások útján modellezik, amellett, hogy az áramkör pontos szerkezetének leírására is képesek. A rendszertervezők így elvonatkoztatások útján tervezhetik meg rendszerük felépítését a különböző tervezési tartományokban, mely során tehát rendszerük adott tartománybeli lényeges vonásait emelik ki, elhanyagolva az érdektelen részleteket. A megfelelő elvonatkoztatási szint és modell így nagyban elősegíti a helyesen működő rendszerterv elkészítését, majd későbbi felosztását és a modell további finomítását.

A szimulációs adatok nyújtotta visszacsatolás alapján hozott tervezési döntések lehetővé teszik nem csupán a tervezési megkötéseket kielégítő megoldás megtalálását, hanem általuk az átfogóan legmegfelelőbb megoldás is elérhető, szemben a szimulációval nem támogatott, visszacsatolás nélküli, pusztán a tervezői tapasztalaton alapuló gyakorlattal szemben.

5. Köszönetnyilvánítás

Ezúton szeretnék köszönetet mondani mindenkinek, akik segítettek a doktori munkámban és az értekezés elkészültében.

Elsősorban köszönöm szüleimnek, hogy támogatásukkal, szeretetükkel biztosították számomra a tanulmányaim folytatásához szükséges családi és anyagi hátteret.

Köszönet illeti témavezetőmet, Dr. Hosszú Gábor egyetemi docenst, akinek biztatására kezdtem el a doktori tanulmányaimat, gondoskodott külföldi tanulmányutamról, munkám szakmai irányításáról, és bármikor fordulhattam hozzá segítségért.

Külön szeretném megköszönni Dr. Kovács Ferenc egyetemi tanárnak szakmai segítségét és az eredmények közlésében nyújtott nagyfokú támogatását.

Köszönöm Claus Schneidernek, hogy a Münchenben, a Siemens cég félvezető részlegénél töltött két féléves kutatásom alatt nagyban segítette munkámat.

Köszönet illeti a BME Elektronikus Eszközök Tanszékét, hogy lehetőséget biztosított doktori munkám végzésre, különösen a Média és Rendszerinformatikai Laboratóriumban kutatókat a számomra nyújtott segítségükért, köztük név szerint is Imre Alexandrát és Rácz Zoltánt a szimulációs eredményekért és a teszt-chip megtervezéséért.

6. Saját közlemények jegyzéke

[S1] G. Hosszú, F. Kovács, **L. Varga**, V. Gajodi: "VHDL Based Circuit Synthesis Using Language Transformations", *Int. Conf. on Parallel Architectures Compilation Techniques*, Paris, France, October 13-17, 1998, pp.130-135.

[S2] G. Hosszú, F. Kovács, **L. Varga**, V. Gajodi: "VHDL Based Circuit Synthesis Using Language Transformations", *Conf. on Application of Microprocessors in Automatic Control and Measurement*, Warsaw, Poland, October 13-14, 1998, pp. 42-48.

[S3] G. Hosszú, F. Kovács, **L. Varga**: "Design Procedure Based on VHDL Language Transformations", *IEEE International Symposium on Circuit and Systems*, Orlando, Florida, USA, May 30-June 2, 1999, Vol 1. pp. 407-410.

[S4] **L. Varga**, G. Hosszú, F. Kovács: "VHDL Based High-Level Synthesis for Datapath-Intensive Architectures", *Int. Workshop on Design, Test and Applications*, Dubrovnik, Croatia, June 14-16, 1999

[S5] **L. Varga**, G. Hosszú, F. Kovács: "Circuit Synthesis Based on VHDL Language Transformations", *IEEE Int. Conf. on Electronics, Circuit and Systems*, Paphos, Cyprus, Sep. 5-8, 1999, pp. 225-228.

[S6] G. Richly, **L. Varga**, J. Horváth, D. Tarján, G. Hosszú, F. Kovács: "Optimum Selection of Sound Stream Segments for Real-Time Identification", *DSP Germany*, Munich, Germany, Sep. 22-23, 1999, pp. 127-131.

[S7] A. Kun, R. Kozma, **L. Varga**, C. Schneider, F. Kovács, G. Hosszú: "VHDL-based Design and Analysis Methodology for Heterogeneous Digital Systems", *Electronic Devices and Systems Conference*, Brno, Czech Republic, Nov. 19-20, 1999, pp. 113-116.

[S8] **L. Varga**, G. Richly, J. Horváth Cz., G. Hosszú, F. Kovács: "Optimal Selection of Sound Stream Segments for Real-Time Identification", *Electronic Devices and Systems Conference*, Brno, Czech Republic, Nov. 19-20, 1999, pp. 240-243.

[S9] **L. Varga**, G. Hosszú, F. Kovács: "Resource-Sharing for Low-Power in High-Level Synthesis", *Electronic Devices and Systems Conference*, Brno, Czech Republic, Nov. 19-20, 1999, pp. 117-120.

[S10] **L. Varga**, G. Hosszú, F. Kovács: "A Power Reduction Technique in High-Level Synthesis of Datapaths", *Design and Diagnostics of Electronic Circuits and Systems,* Slovakia, April 5-7, 2000, pp. 142-145.

[S11] F. Kovács, G. Hosszú, G. Richly, **L. Varga**: "Monitoring Media Streams on the Internet", *Hungarian – Korean Joint Seminar*, Hungary, May 3-6, 2000

[S12] G. Richly, **L. Varga**, F. Kovács, G. Hosszú: "A Real-time method to characterize sound streams for occurrence monitoring of given sound-prints", *International Spring Seminar on Electronics Technology*, Hungary, May 6-10, 2000, pp. 103-105.

[S13] **L. Varga**, R. Kozma, A. Kun, G. Hosszú, F. Kovács, C. Schneider: "VHDL-Based System-Level Design Methodology for Multimedia Signal Processing Applications", *Mediterranean Electrotechnical Conf.*, Cyprus, May 29-31, 2000, pp. 814-817.

[S14] **L. Varga**, G. Hosszú, F. Kovács: "A Low-Power Design Technique for Digital Signal Processing Applications", *IEEE Mediterranean Electrotechnical Conf.*, Cyprus, May 29-31, 2000, pp. 827-830.

[S15] G. Richly, **L. Varga**, G. Hosszú, F. Kovács: "Short-Term Sound Stream Characterization for Reliable, Real-Time Occurrence Monitoring of Given Sound-Prints", *IEEE Mediterranean Electrotechnical Conf.*, Cyprus, May 29-31, 2000, pp. 526-528.

[S16] **Varga L.**, Richly G., Kozma R., Kovács F., Hosszú G.: "Mintaillesztési Algoritmus Fejlesztése és Megvalósítása", *Magyar Informatikusok Második Világtalálkozója*, Budapest, Hungary, 2000 jún. 5-8. pp. 1025-1035.

[S17] Richly G., **Varga L.**, Kozma R., Kovács F., Hosszú G.: "Internetes Média-Folyamon Alkalmazott Hanganyag Felismerő Rendszer", *Magyar Informatikusok Második Világtalálkozója*, Budapest, Hungary, 2000 jún. 5-8. pp. 827-837.

[S18] F. Kovács, **L. Varga**, G. Hosszú: "Circuit Optimization of Adiabatic Charge-Recovery CMOS PLA-s", *World Multiconference on Systemics, Cybernetics and Informatics*, Orlando, Florida, USA, July 23-26, 2000, Vol. IX. pp. 153-156.

[S19] F. Kovács, G. Hosszú, G. Richly, **L. Varga**: "Real-Time Identification of Predefined Records in Stream-Media Using Sound-prints Selected for Minimum Similarity", *World Multiconf. on Systemics, Cybernetics and Informatics*, Orlando, Florida, USA, July 23-26, 2000, Vol. IV. pp. 40-43.

[S20] **L. Varga**, F. Kovács, G. Hosszú, "A Novel Low Power CMOS Using Adiabatic Switching", *MicroCad 2001*, Miskolc, Hungary, March 1-2, 2001, pp. 57-61.

[S21] A. Imre, **L. Varga**, F. Kovács, G. Hosszú, "Comparative Study of Adiabatic Power Supply Generators", *MicroCad 2001*, Miskolc, Hungary, March 1-2, 2001, pp. 31-34.

[S22] R. Kozma, G. Richly, **L. Varga**, G. Hosszú, "Recognition System for Monitoring in Internet Media-stream for Sound-clips", *MicroCad 2001*, Miskolc, Hungary, March 1-2, 2001, pp. 45-49.

[S23] Kozma R., **Varga L.**, Hosszú G., Kovács F.: "Object Oriented Hardware Synthesis Basen on Language Transformation", *MicroCad 2001*, Miskolc, Hungary, March 1-2, 2001, pp. 39-43.

[S24] **L. Varga**, F. Kovács, G. Hosszú: "An Efficient Adiabatic Charge-Recovery Logic", *IEEE SoutheastCon 2001*, Clemson, South Carolina, USA, March 30 - April 1, 2001, pp. 17-20.

[S25] **L. Varga**, F. Kovács, G. Hosszú: "Mixed-Signal Method for Low-Power Pattern Preselection", *IEEE Design and Diagnostics of Electronic Circuits and Systems*, Győr, Hungary, April 18-20, 2001, pp. 103-105.

[S26] **L. Varga**, G. Hosszú, F. Kovács: "Adiabatic Charge-Recovery CMOS for Ultra-Low-Power", *IEEE Design and Diagnostics of Electronic Circuits and Systems*, Győr, Hungary, April 18-20, 2001, pp. 227-231.

[S27] **L. Varga**, G. Hosszú, F. Kovács: "A Scheduling Technique for Pipeline Datapaths Using Adiabatic Logic", *IEEE Design and Diagnostics of Electronic Circuits and Systems*, Győr, Hungary, April 18-20, 2001, pp. 307-310.

[S28] **L. Varga**, F. Kovács, G. Hosszú: "Approaches for Scheduling of Adiabatic Logic", *Int. Workshop on Logic & Synthesis*, Granlibakken, CA, USA, June 12-15, 2001, pp. 18-22.

[S29] Richly G., **Varga L.**, Kozma R, Hosszú G.: "Internetes Média-Folyamon Alkalmazott Hanganyag Felismerő Rendszer", *Informatika*, 4. évf. 2. szám, 2001 Május, pp. 18-25.

[S30] **L. Varga**, F. Kovács, G. Hosszú: "Datapath Synthesis Using Adiabatic Logic", WSES/IEEE CSCC 2001, Rethymnon, Greece, July 8-15, 2001.

[S31] **L. Varga**, F. Kovács, G. Hosszú: "An Improved Pass-Gate Adiabatic Logic", *IEEE ASIC/SOC 2001*, Washington, USA, September 12-15, 2001, pp. 208-211.

[S32] R. Kozma, **L Varga**, Cs. Horváth, F. Kovács, G. Hosszú: "Object-oriented Hardware Synthesis Using Language Transformations", *Int. Workshop on Control & Information Technology*, Ostrava, Czech Republic, Sep. 19-20, 2001, pp. 33-39.

[S33] **L. Varga**, G. Hosszú, F. Kovács: "Design Procedure Based on VHDL Language Transformations", *VLSI Design; International Journal of Custom-Chip Design, Simulation, and Testing*, Vol. 14, No. 4, 2002, pp. 349-354.

[S34] **L. Varga**, G. Hosszú, F. Kovács: "Two-level Pipeline Scheduling of Adiabatic Logic", *ISSE,* St. Marienthal, Germany, May. 2006, pp. 390-394.
[S35] **L. Varga**, G. Hosszú: "Kisfogyasztású integrált áramkörök tervezési kérdései", *Híradástechnika*, Vol. LXI, No. 12, 2006, pp. 45-51.
[S36] **L. Varga**, G. Hosszú: "High-level Synthesis of Four-phase Adiabatic Logic", 6th *Electronic Circuits and Systems Conference,* Bratislava, Slovakia, Sep. 6-7, 2007.
[S37] **L. Varga**, G. Hosszú: "Adiabatikus töltésvisszanyerő elven működő kisfogyasztású integrált áramkörök", *Elektronikai Technológia, Mikrotechnika*, megjelenés alatt.

7. Független idézettség

[C1] H. Lee, I. Na, C. Lee, Y. Moon: "A 16-bit adiabatic macro blocks with supply clock generator for micro-power RISC datapath", *International Technical Conference on Circuits / Systems, Computers and Communications*, Phuket, Thailand, July 16-19, 2002, pp. 1563-1566. [S24]

[C2] P. Cano, E. Batlle, T. Kalker, J. Haitsma: "A review of algorithms for audio fingerprinting", *IEEE International Workshop on Multimedia Signal Processing, St. Thomas, US Virgin Islands,* Dec. 2002, pp. 169-173. [S15]

[C3] Y. Shin, H. Lee, C. Lee, Y. Moon: "A Design of 16-bit Adiabatic Low-Power Microprocessor", Journal of the Institute of Electronics Engineers of Korea, Vol. 40, No 6, 2003. november, pp. 31-38. [S24]

[C4] Henry Y. K. Lau, K. L. Mak: "The design of flexible manufacturing systems using an extended unified framework", Journal of Manufacturing Technology Management, Vol. 15, No. 3, 2004, pp. 222-238. [S3]

[C5] M.M. Yang, J.A. Barby: "A novel fast low voltage dynamic threshold true single phase clocking adiabatic circuit", *IEEE International Symposium on Circuits and Systems,* Vancouver, Canada, 2004. május 23-26, pp. 289-292. ISBN 0-7803-8251-X, Publisher: IEEE, Piscataway, New Jersey, USA. [S24]

[C6] M. Arsalan, M. Shams: "An investigation into transistor-based adiabatic logic styles", *IEEE Northeast Workshop on Circuits and Systems*, Montreal, Canada, 2004. június 20-23, pp. 1-4. [S31]

[C7] Y. Shin, C. Lee, Y. Moon: "A Low Power 16-Bit RISC Microprocessor Using ECRL Circuits", *ETRI Journal,* Vol. 26, No. 6, December 2004, pp. 513-519. [S24]

[C8] P. Cano, E. Batlle, T. Kalker, J. Haitsma: "A Review of Audio Fingerprinting", *Journal of VLSI Signal Processing Systems*, Vol. 41, Issue 3, Nov. 2005, pp. 271-284. ISSN: 0922-5773, Publisher: Kluwer Academic Publishers, Hingham, MA, USA. [S15]

[C9] T. Akiba, M. Igarashi: "Hardware-operation description conversion method and program", *US Patent*, 2006. február 7, No: 6,996,788. [S5]

[C10] V. S. K. Bhaaskaran, S. Salivahanan, D. S. Emmanuel: "Semi-Custom Design of Adiabatic Adder Circuits", *19th International Conference on VLSI Design held jointly with 5th International Conference on Embedded Systems Design*, Hyderabad, India, 2006. január 3-7, pp. 745-748. [S31]

[C11] C. Schlachta: "Ein Verfahren zur Verbesserung der Energiee zienz durch Ladungsrückgewinnung in Digitalschaltungen ", Ph.D. értekezés, 2006. URL: http://deposit.ddb.de/cgi-bin/dokserv?idn=985741708&dok_var=d1&dok_ext=pdf&filename=985741708.pdf [S31]

[C12] P Cano: "Content-Based Audio Search: from Fingerprinting to Sematic Audio Retrieval", *Ph.D. értekezés*, 2006. URL: http://www.iua.upf.edu/mtg/publications/34ac8d-PhD-Cano-Pedro-2007.pdf. [S15]

[C13] Kirei Botond Sándor: "A VHDL kódtól az FPGA-ba való ágyazásig", *Műszaki Szemle*, 33. szám, 2006, Erdélyi Magyar Műszaki Tudományos Társaság, Kolozsvár, pp. 53-58. [S2]

[C14] V. Ponnusamy, K Gunavathi: "Energy Efficient Charge Recovery for Positive Feedback Adiabatic Logic", *IETE Technical Review*, (ISSN 0256-4602), Vol. 24, No. 2, March-April 2007, New Delhi, India, pp. 127-133. [S31]

8. Irodalomjegyzék

[1] D. D. Gajski, L. Ramachandran: "Introduction to High-Level Synthesis", *IEEE Design & Test of Computers*, Winter 1994, pp. 44-54.

[2] R. A. Walker, S. Chaudhuri: "Introduction to the Scheduling Problem", *IEEE Design & Test of Computers*, Summer 1995, pp. 60-69.

[3] B. M. Pangrle, D. D. Gajski: "Design Tools for Intelligent Silicon Compilation", *IEEE Trans. on Comp. Aided Design*, Vol. 6, No. 6, Nov. 1987, pp. 1098-1112.

[4] P. G. Paulin, J. P. Knight: "Algorithms for High-Level Synthesis", *IEEE Design & Test of Computers*, Vol. 6, No. 4, Dec. 1989, pp. 18-31.

[5] C. T. Hwang, J. H. Lee, J. C. Hsu: "A Formal Approach to the Scheduling Problem in High Level Synthesis", *IEEE Trans. on Comp. Aided Design*, Vol. 10, No. 4, Apr. 1991, pp. 464-475.

[6] C. Tseng, D. P. Siewiorek: "Automated Synthesis of Data Paths in Digital Systems", *IEEE Trans. on Comp. Aided Design*, Vol. 5, July 1986, pp. 379-395.

[7] C. Y. Roger Chen, M. Z. Moricz: "Data Path Scheduling for Two-Level Pipelining", *IEEE/ACM Design Automation Conf.*, 1991, pp. 603-606.

[8] H. T. Kunk, M. S. Lam: "Wafer-Scale Integration and Two-Level Pipelined Implementations of Systolic Arrays", *Journ. of Parallel and Distributed Computing*, 1984, pp. 32-63.

[9] R. Jain, A. Mujumdar, A. Sharma, H. Wang: "Empirical Evaluation of Some High-Level Synthesis Scheduling Heuristics", *Design Automation Conf.*, 1991, pp. 686-689.

[10] M. C. McFarland, A. C. Parker, R. Camposano: "The High-Level Synthesis of Digital Systems", *IEEE*, Feb. 1990, pp. 301-318.

[11] M. Liu, W. Wang, M. Orshansky: "Leakage power reduction by dual-V_{th} designs under probabilistic analysis of V_{th} variation", *Int. Symp. Low Power Electronics and Design*, Aug. 2004, pp. 2-7.

[12] R. Jejurikar, C. Pereira, R. Gupta: "Leakage Aware Dynamic Voltage Scaling for RealTime Embedded Systems", *IEEE/ACM Design Automation Conf.*, Jun. 2004.

[13] N. Dragone, C. Guardiani, R. Zafalon, P. Meier: "An Innovative Methodology for the Design Automation of low Power Libraries", *Int. Workshop on Power and Timing Modeling, Optimization and Simulation*, Oct. 1998, pp. 31-40.

[14] J. Nyathi, V. Beiu, S. Tatapudi, D. J. Betowski: "A Charge Recycling Differential Noise Immune Perceptron", *IJCNN*, Jul. 2004, pp. 1995-2000.

[15] R. S. Shelar, S. S. Sapatnekar: "An Efficient Algorithm for Low Power Pass Transistor Logic Synthesis", *Asia and South Pacific Design Automation Conf.*, Jan. 2002, pp. 87-92.

[16] D. Somasekhar, K. Roy: "Differential Current Switch Logic: a Low Power DCVS Logic Family", *IEEE Journ. Solid-State Circuits*, Vol. 31, No. 7, July. 1996, pp. 981-991.

[17] P. Vuillod, L. Benini, A. Bogliolo, G. D. Micheli: "Clock-Skew Optimization for Peak Current Reduction", *Kluwer Journ. of VLSI Signal Processing*, Vol. 16, No. 2-3, 1997.

[18] J. Ludwig, S. Nawab, A. Chandrakasan: "Low-Power Digital Filtering Using Approximate Processing", *IEEE Journ. Solid-State Circuits*, vol-31, No.3, Mar. 1996, pp. 395-400.

[19] R. Murgai, M. Fujita, A. Oliveira: "Using Complementation and Resequenceing to Minimize Transitions", *IEEE/ACM Design Automation Conf.*, June 1998, San Francisco, CA, pp. 694-697.

[20] IEEE Standard VHDL Language Reference Manual, *IEEE*, New York, 1988.

[21] L. A. Glasser, D. W. Dobberpuhl: "The Design and Analysis of VLSI Circuits", *Addison Wesley*, 1993.

[22] P. Paulin, J. Fréhel, M. Harrad, E. Berrebi, C. Liem, F. Nacabal, J. C. Herluison: "High-Level Synthesis and Codesign Methods: An Application to a Videophone Codec", *EURO-DAC with EURO-VHDL*, 1996.
[23] R. Brodersen, A. Chandrakasan, S. Sheng: "Technologies for Personal Communications", *IEEE Symp. on VLSI Circuits*, Tokyo, Japan, 1991, pp. 5-9.
[24] L. Benini, G. D. Micheli, E. Macii, M. Poncino, S. Quer: "Power Optimization of Core-Based Systems by Address Bus Encoding", *IEEE Trans. on VLSI Systems*, Vol. 6, No. 4, Dec. 1998, pp. 554-562.
[25] M. Potkonjak, M. Srivastava: "Behavioral Optimization Using the Manipulation of Timing Constraints", *IEEE Trans. on Comp. Aided Design of Int. Circuits and Systems*, Vol. 17, No. 10, Oct. 1998, pp. 936-947.
[26] A. Chandrakasan, S. Sheng, R. Brodersen: "Low-Power CMOS Digital Design", *IEEE Journ. Solid-State Circuits*, Vol. 27, No. 4, Apr. 1992, pp. 473-484.
[27] A. Chandrakasan, M. Potkonjak, R. Mehra, J. Rabay, R. Brodersen: "Optimizing Power Using Transformations", *IEEE Trans. on Comp. Aided Des. of Int. Circuits and Systems*, Vol. 14, No. 1, Jan. 1995, pp. 12-30.
[28] K. Roy, S. C. Prasad: "Circuit Activity Based Logic Synthesis for Low Power Reliable Operations", *IEEE Trans. on VLSI Systems*, Vol. 1, No. 4, Dec. 1993, pp. 503-513.
[29] S. Uppalapati, M. L. Bushnell, V. D. Agrawal: "Glitch-Free Design of Low Power Asics Using Customized Resistive Feedthrough Cells", *9th VLSI Design and Test Symp.*, Aug. 2005, pp. 41-48.
[30] C. Tsui, M. Pedram, A. Despain: "Technology Decomposition and Mapping Targeting Low-Power Dissipation", *IEEE/ACM Design Automation Conf.*, 1993, pp. 68-73.
[31] R. Iyer, D. Rossetti, M. Hsueh: "Measurement and Modeling of Computer Reliability as Affected by System Activity", *ACM Trans. on Computer Systems*, Vol. 4, No. 3, Aug. 1986, pp. 214-237.
[32] M. Pedram: "Power Minimization in IC Design", *ACM Trans. on Design Automat. Electron. Syst.*, Vol. 1, No. 1, Jan. 1996.
[33] N. H. Weste, K. Eshraghian: "Principles of CMOS VLSI Design: A System Perspective", Addison Wesley, 1993.
[34] C. H. Ziesler, S. Kim, M. C. Papaefthymniou: "A Resonant Clock Genereator for Single-Phase Adiabatic Systems", *Int. Symp. Low Power Electronics and Design*, Newport Beach, CA, Aug. 6-7, 2001.
[35] H. M. Meimand, A. A. Kusha: "Efficient power clock generation for adiabatic logic", *IEEE International Symposium on Circuits and Systems*, May 6-9, 2001, pp. 642-645.
[36] A. Stoffi, G. Iannaccone, S. Di Pascoli, E. Amirante, D. Schmitt-Landsiedel: "Four-phase power clock generator for adiabatic logic circuits", *Electronics Letters*, July 4, 2002, Vol 38, No. 14, pp. 689-690.
[37] A. G. Dickinson, J. S. Denker: "Adiabatic Dynamic Logic", *IEEE Journ. Solid-State Circuits*, Vol. 30, Mar. 1995, pp. 311-314.
[38] T. J. Gabara: "Pulsed Low Power CMOS", *Int. Journ. High Speed Electron Syst.*, Vol. 5, No. 2, 1994.
[39] D. Maksimovic, V. G. Oklobdzija: "Integrated Power Clock Generators for Low-Energy Logic", *IEEE Power Electron. Special. Conf.*, 1995, pp. 61-67.
[40] M. Steyaert, J. Craninckx: "1.1 GHz Oscillator Using Bondwire Inductance", *Electronics Letters*, 1994.
[41] N. Tzartzanis, W. C. Athas: "Design and Analysis of a Low-Power Energy-Recovery Adder", *IEEE 5th Great Lakes Symp. on VLSI*, Mar. 16-18, 1995, pp. 66-69.
[42] R. Hinman, M. Schlecht: "Recovered Energy Logic – A Highly Efficient Alternative to Today's Logic Circuits", *IEEE Power Electron. Spec. Conf.*, 1993.

[43] S. G. Younis, T. Knight: "Practical Implementation of Charge Recovering Asymptotically Zero Power CMOS", *Symp. on Integrated Systems*, 1993, pp. 234-250.
[44] D. Maksimovic, V. G. Oklobdzija, B. Nikolic, K. W. Current: "Clocked CMOS Adiabatic Logic with Integrated Single-Phase Power-Clock Supply", *IEEE Trans. VLSI Systems Vol. 8, No. 4,* Aug. 2000 pp. 460-463.
[45] V. G. Oklobdzija, D. Maksimovic, F. Lin: "Pass-Transistor Adiabatic Logic Using Single Power-Clock Supply", *IEEE Trans. Circuits and Systems Vol. 44, No. 10,* Oct. 1997 pp. 842-846.
[46] S. Kim, M. C. Papaefthymniou: "True Single-Phase Energy-Recovering Logic for Low-Power, High-Speed VLSI", *Int. Symp. Low Power Electronics and Design*, Monterey, CA, Aug. 10-12, 1998, pp. 167-172.
[47] S. M. Yoo, S. M. Kang: "A Bootstrapped NMOS Charge Recovery Logic", *IEEE 8th Great Lakes Symp. on VLSI*, Feb. 1998, pp.30-33.
[48] W. C. Athas, N. Tzartzanis, L. J. Svensson, L. Peterson: "A Low-Power Microprocessor Based on Resonant Energy", *IEEE Journ. Solid-State Circuits*, Nov. 1997 pp. 1693-1701.
[49] W. C. Athas, W-C Liu, L. J. Svensson: "Energy-Recovery CMOS for Highly Pipelined DSP Design", *Int. Symp. Low Power Electronics and Design*, Monterey, CA, Aug. 12-14, 1996.
[50] J. G. Koller, W. C. Athas: "Adiabatic Switching, Low Energy Computing, and the Physics of Storing and Erasing Information", *Workshop on Physics and Computation*, Oct. 1992.
[51] W. C. Athas, J. G. Koller, L. J. Svensson: "An Energy-Efficient CMOS Line Driver Using Adiabatic Switching", *IEEE 4th Great Lakes Symp. on VLSI Design*, Mar. 1994, pp.159-164.
[52] W. C. Athas, L. J. Svenson, J. G. Koller, N. Tzartzanis, E. Chou: "Low-Power Digital Systems Based on Adiabatic-Switching Principles", *IEEE Trans. on VLSI Systems*, Vol. 2, No. 4, Dec. 1994, pp. 398-407.
[53] J. Lim, D. K. Won, S. I. Chae: "Reversible Energy Recovery Logic Circuit without Nonadiabatic Energy Loss", *Electron Lett.*, Vol. 34, No. 4, Feb. 1998, pp. 344-346.
[54] J. Lim, D. G. Kim, S. I. Chae: "nMOS Reversible Energy Recovery Logic for Ultra-Low-Energy Applications", *IEEE Journ. Solid-State Circuits*, Vol. 35, No. 6, June 2000, pp. 865-875.
[55] S. G. Younis, T. Knight: "Asymptotically Zero Energy Split-Level Charge Recovery Logic", *Int. Workshop on Low Power Design*, Napa Valley, CA, 1994, pp. 177-182.
[56] A. Kramer, J. S. Denker, S. C. Avery, A. G. Diskinson, T. R. Wik: "Adiabatic Computing with the 2N-2N2D Logic Family", *IEEE Symp. on VLSI Circuits*, 1994, pp. 25-26.
[57] Y. Moon, D. K. Jeong: "An Efficient Charge Recovery Circuit", *IEEE Journ. Solid-State Circuits*, Vol. 31, No. 4, Apr. 1996 pp. 514-522.
[58] A. Kramer, J. S. Denker, B. Flower, J. Moroney: "2nd Order Adiabatic Computation with 2N-2P and 2N-2N2P Logic Circuits", *Int. Symp. on Low Power Design*, Apr. 1995, pp. 191-196.
[59] B. Frigyik, Á. Csurgay, F. Kovács, W. Porod: "Low Power Conservative Circuits in CMOS", *Int. Workshop on Power and Timing Modeling, Optimization and Simulation*, Lingby, 1998, pp. 345-353.
[60] C. Kim, S. M. Yoo, S. M. Kang: "NMOS Energy Recovery Logic", *IEEE 9th Great Lakes Symp. on VLSI*, 1999, pp. 310-313.
[61] S. Kim, M. C. Papaefthymniou: "Single-Phase Source-Coupled Adiabatic Logic", *Int. Symp. Low Power Electronics and Design*, San Diego, Aug. 1999, pp. 97-99.

[62] Y. Takahashi, T. Sekine, M. Yokoyama: " A 4-bit multiplier using a two phase drive adiabatic dynamic CMOS logic", *Proc. ITC-CSCC*, July 8-11, 2007, pp. 205-206.
[63] N. Tzartzanis, W. C. Athas: "Energy-Recovery for the Design of High-Speed, Low-Power Static RAMs", *Int. Symp. Low Power Electronics and Design*, Monterey, CA, Aug. 12-14, 1996. pp. 55-60.
[64] Y. Shin, C. Lee, Y. Moon: "A Low Power 16-Bit RISC Microprocessor Using ECRL Circuits", *ETRI Journal,* Vol. 26, No. 6, Dec. 2004. pp. 513-519.
[65] J. Lim, D. G. Kim, S. I. Chae: "A 16-bit Carry-Lookahead Adder Using Reversible Energy Recovery Logic for Ultra-Low-Energy Systems", *IEEE Journ. Solid-State Circuits*, Vol. 24, No. 6, June 1999, pp. 898-903.
[66] C. L. Seitz, A. H. Frey, S. Matisson, S. D. Rabin, D. A. Speck, J. A. Snepsheut: "Hot-Clock NMOS", *Chapel Hill Conf. on VLSI*, 1985.
[67] P. Teichmann, J. Fischer, S. Henzler, E. Amirante, D. Schmitt-Landsiedel: "Power-Clock Gating in Adiabatic Logic Circuits", *PATMOS*, Leuven, Sep. 21-23, 2005, pp. 638–646.
[68] S. Kim, C. H. Ziesler, M. C. Papaefthymniou: "Design, Verification, and Test of a True Single-Phase 8-bit Adiabatic Multiplier", *Conf. on Advanced Research in VLSI*, Mar. 2001.
[69] S. Kim, C. H. Ziesler, M. C. Papaefthymniou: "A True Single-Phase 8-bit Adiabatic Multiplier", *IEEE/ACM Design Automation Conf.*, Las Vegas, Jun. 18-22, 2001.
[70] M. C. Knapp, P. J. Kindlmann, M. C. Papaefthymniou: "Implementing and Evaluating Adiabatic Arithmetic Units", *IEEE Custom Integrated Circuit Conf.*, 1996.
[71] A. Schlaffer, J. A. Nossek: "Register Design for Adiabatic Circuits", *Int. Workshop on Power and Timing Modeling, Optimization and Simulation*, 1999.
[72] G. Lehmann, B. Wunder, K. D. Müller-Glaser: "A VHDL Reuse Workbench", *EURO-DAC with EURO-VHDL*, 1996.
[73] Csurgay Árpád, Simonyi Károly: "Az információtechnika fizikai alapjai", 1997.
[74] L. J. Svensson, J. G. Koller: "Adiabatic Charging without Inductors", *Int. Workshop on Low-Power Design*, Apr. 1994, pp. 159-164.
[75] L. J. Svensson, J.G. Koller: "Driving a Capacitive Load without Dissipating fCV^2", *IEEE Symp. on Low-Power Electronics*, Oct 1994.
[76] F. Gao, J. P. Hayes: "Total Power Reduction in CMOS Circuits via Gate Sizing and Multiple Threshold Voltages", *IEEE/ACM Design Automation Conf.*, Jun. 2005, pp. 31-36.
[77] H. S. Jun, S. Y. Hwang: "Design of a Pipelined DataPath Synthesis System for Digital Signal Processing", *IEEE Trans. VLSI Systems Vol. 2, No. 3,* Sep. 1994 pp. 292-303.
[78] N. Park, A. C. Parker: "Sehwa: A Software Package for Synthesis of Pipelines from Behavioral Specification", *IEEE Trans. on Comp. Aided Design*, Vol. 7, No. 3, Mar. 1988, pp. 356-370.
[79] K. Hwang, A. E. Cavasant, C. Chang, M. A. d'Abreu: "Scheduling and Hardware Sharing in Pipelined Data Paths", *IEEE Int. Conf. on Computer Aided Design*, Nov. 1989, pp. 24-27.
[80] W. F. J. Verhaegh, P. E. R. Lippens, E. H. L. Aarts, J. H. M. Korst, J. L. van Meerbergen, A. van der Werf: "Improved Force-Directed Scheduling in High-Throughput Digital Signal Processing", *IEEE Trans. on Comp. Aided Design*, Vol. 14 1995, pp. 945-960.
[81] N. Park, F. J. Kurdahi: "Module Assignment and Interconnect Sharing in Register-Transfer Synthesis of Pipelined Data Paths", *IEEE Int. Conf. on Computer Aided Design*, Nov. 1989, pp. 16-19.

[82] H. Trickey: "Flamel: A High-Level Hardware Compiler", *IEEE Trans. on Comp. Aided Design*, Vol. 6, Mar. 1987, pp. 259-269.
[83] M. R. Stan, W. P. Burleson: "Bus-Invert Coding for Low-Power I/O", *IEEE Trans. on VLSI Systems*, Vol. 3, No. 1, Mar. 1995, pp. 49-58.
[84] J. Y. Chen, W. B. Jone, J. S. Wang, H. I. Lu, T. F. Chen: "Segmented Bus Design for Low-Power System", *IEEE Trans. on VLSI Systems*, Vol. 7, No. 1, Mar. 1999, pp. 25-29.
[85] R. Mehra, J. Rabaey: "Exploiting Regularity for Low-Power Design", *IEEE Int. Conf. on Computer Aided Design*, 1996.
[86] J. Monteiro, P. Ashar, A. Mauskar: "Scheduling Techniques to Enable Power Management", *IEEE/ACM Design Automation Conf.*, Las Vegas, Jun. 1996.
[87] G. Lakshminarayana, A. Raghunathan, N. K. Jha, S. Dey: "Power Management in High-Level Synthesis", *IEEE Trans. on VLSI Systems*, Vol. 7, No. 1, Mar. 1999, pp. 7-14.
[88] J. M. Chang, M. Pedram: "Module Assignment for Low-Power", *EURO-DAC with EURO-VHDL*, 1996.
[89] J. M. Chang, M. Pedram: "Register Allocation and Binding for Low Power", *IEEE/ACM Design Automation Conf.*, Jun. 1995, pp. 29-35.
[90] S. Raje, M. Sarrafzadeh: "Variable Voltage Scheduling", *Int. Symp. Low Power Design*, Apr. 1995, pp. 9-14.
[91] N. Azizi, M. M. Khellah, V. De, F. N. Najm: "Variations-Aware Low-Power Design with Voltage Scaling", *IEEE/ACM Design Automation Conf.*, Jun. 2005, pp. 529-534.
[92] N. Weng, J. S. Yuan, R. F. DeMara, D. Ferguson, M. Hagedorn: "Glitch Power Reduction for Low Power IC Design", *Ninth Annual NASA Symposium on VLSI Design*, Nov. 2000, pp. 7.5.1–7.5.7.
[93] R. Mehra, J. Rabaey: "Behavioral Level Power Estimation and Exploration", *Int. Workshop on Low Power Design*, Napa Valley, California, Apr. 1994, pp. 197-202.
[94] J. E. Crenshaw, M. Sarrafzadeh: "Accurate High Level Datapath Power Estimation", *EURO-DAC*, 1997.
[95] P. E. Landman, J. M. Rabaey: "Architectural Power Analysis: The Dual Bit Type Method", *IEEE Trans. on VLSI Systems*, Vol. 3, No. 2, June 1995, pp. 173-187.
[96] M. Nemani, F. N. Najm: "High-Level Area and Power Estimation for VLSI Circuits", *IEEE Trans. on Comp. Aided Design of Int. Circuits and Systems*, Vol. 18, No. 6, June 1999, pp. 697-713.
[97] R. A. Walker, D. E. Thomas: "Design Representation and Transformation in the System Architect's Workbench", *IEEE Int. Conf. on Computer Aided Design*, Nov. 1987, pp. 166-169.
[98] R. Camposano, R. M. Tabet: "Design Representation for the Synthesis of Behavioral VHDL Models", *Int. Symp. Comput. Hardware Description Languages and Their Applications*, Washington DC, June 1989, pp. 49-58.
[99] R. Camposano, R. A. Bergamaschi, C. E. Haynes, M. Payer, S. M. Wu: "High Level VLSI Synthesis", *Kluwer Academic Publishers*, 1991.
[100] J. Biesenack, M. Koster, A. Langmaier, S. Ledeux, S. März, M. Payer, M. Pilsl, S. Rumler, H. Soukup, N. Wehn, P. Duzy: "The Siemens High-Level Synthesis System CALLAS", *IEEE Trans. on VLSI Systems*, Vol. 1, No. 3, Sep. 1993, pp. 244-252.
[101] R. A. Bergamaschi, R. O'Connor, L. Stok, M. Moricz, S. Prakash, A. Kühlmann, D. S. Rao: "High-Level Synthesis in an Industrial Environment", *IBM Journ. Res. Development*, Vol. 39, Jan./Mar. 1995, pp. 131-148.
[102] D. Knapp, T. Ly, D. MacMillen, R. Miller: "Behavioral Synthesis Methodology for HDL-Based Specification and Validation", *IEEE/ACM Design Automation Conf.*, 1995.

[103] Z. Peng: "Digital System Simulation with VHDL in a High-Level Synthesis System", *Microprocessing and Microprogramming, the Euromicro Journal*, Vol. 35, 1992.
[104] A. A. Jerraya, H. Ding, P. Kission, M. Rahmouni: "Behavioral synthesis and component reuse with VHDL", *Kluwer Academic Publishers*, Boston, 1997.
[105] D. Gajski, F. Vahid, S. Narayan, J. Gong, "Specification and Design of Embedded Systems" *Prentice Hall*, 1994.
[106] L. Pirmez, M. Rahmouni, P. Kission, A. Pedroza, A. Mesquita, A. A. Jerraya: "Analysis of Different Protocol Description Styles in VHDL for High-Level Synthesis", *EURO-DAC with EURO-VHDL*, 1996.
[107] P. Keresztes: "Value Trace VHDL Blocks and Their Application in High Level Synthesis ", *2nd Workshop on Libraries, Component Modeling and Quality Assurance*, Toledo, Spain, April 1997, pp. 225-232.
[108] J. Gong, C. T. Chen, K. Kücükcakar: "Architectural Rule Checking for High-Level Synthesis", *Design Automation and Test in Europe*, 1998, pp. 949-950.
[109] T. Karnik, Y. Ye, J. Tschanz, L. Wei, S. Burns, V. Govindarajulu, V. De, S. Borkar: "Total Power Optimization By Simultaneous Dual-Vt Allocation and Device Sizing in High Performance Microprocessors", *IEEE/ACM Design Automation Conf.*, Jun. 2002.
[110] A. A. Duncan, D. C. Hendry: "Area Efficient DSP Datapath Synthesis", *EURO-VHDL Conf.*, 1995, S-6.2.
[111] C. H. Gebotys, M. I. Elmasry: "Global Optimization Approach for Architectural Synthesis", *IEEE Trans. on Comp. Aided Design of Int. Circuits and Systems*, Vol. 12, No. 9, Mar. 1988, pp. 1266-1278.
[112] R. Cloutier, D. Thomas: "The Combination of Scheduling, Allocation and Mapping in a Single Algorithm", *IEEE/ACM Design Automation Conf.*, Jun. 1990.
[113] G. Sery, S. Borkar, V. De: "Life Is CMOS: Why Chase the Life After?" *IEEE/ACM Design Automation Conf.*, Jun. 2002, pp. 78-83.
[114] J. Fischer, P. Teichmann, A. Gargagli-Stoffi, E. Amirante, D. Schmitt-Landseidel: "Scaling Trends in Adiabatic Circuits", *1st Int'l Workshop on Reversible Computing*, Ischia, Italy, May 4-6, 2005.
[115] S. Bhunia, N. Banerjee, Q. Chen, H. Mahmoodi, K. Roy: "A Novel Synthesis Approach for Active Leakage Power Reduction Using Dynamic Supply Gating" *IEEE/ACM Design Automation Conf.*, Jun. 2005, pp. 479-484.