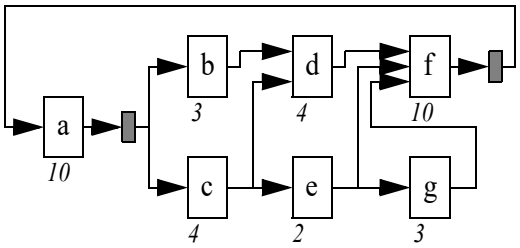
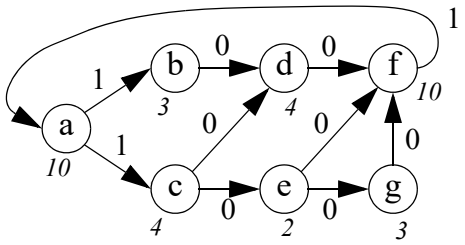
Retiming, automaatide testitavuse alused – ülesanded ja näidislahendused

1. Retiming

Antud on joonisel kujutatud skeem. Skeemi plokkideks võivad olla nii loogikalülid kui ka keerulisemad funktsioonid. Plokkidevahelised ühendused võivad vastata nii ühele bitile (loogikalülide korral) kui ka siinidele/sõnadele (funktsioonid). Retiming töötab sõltumata kasutatavast abstraktsioonitasemest. Oluline on see, et skeem oleks teisendatav sünkroonseks loogikavõrkgraafiks - sõlm vastab plokile, kaar ühendusele ja kaare kaal sünkroonsele viitele (registri olemasolu). Skeemile vastav võrkgraaf on toodud skeemi all. Igal ploki viide on toodud ka sõlme juures (sõlme kaal).





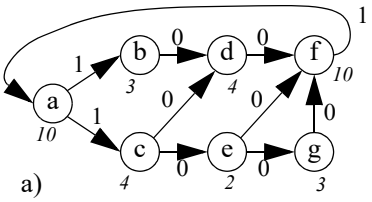
Formaalselt võiks skeemi kirjeldada ka andmesõltuvuste alusel: $a=f_a(r_f)$, $r_a=z^{t-1}(a)$, $b=f_b(r_a)$, $c=f_c(r_a)$, $d=f_d(b,c)$, $e=f_e(c)$, $f=f_f(d,e,g)$, $g=f_g(e)$ ja $r_f=z^{t-1}(f)$; kus $z^{t-1}(x)$ tähistab mäluelementi (registrit). Plokkide (kombinatoorsed) viited on (ajaühikutes): $d_a=10$, $d_b=3$, $d_c=4$, $d_c=4$, $d_e=2$, $d_f=10$ ja $d_g=3$. Registritel kombinatoorne viide puudub. Taktsageduse periood $t_\phi=15$ ajaühikut.

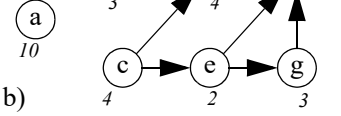
Retiming'u algoritm:

- (1) kombinatoorse skeemi saamiseks eemaldada registrid (servad >0);
- (2) leida sõlmede andmete valmisoleku ajad (t_x) eeldadusel, et kombinatoorsete sisendite andmete valmisoleku ajad $t_i=0$, st. et signaaliteede viited tuleks leida (kombinatoorsetest) sisenditest iga sõlme väljundini;
- (3) tähistada kõik sõlmed, mille korral $t_x>t_\phi$, ehk need kombinatoorsed teed, mille puhul ajaline piirang on rahuldamata; algoritmi töö lõpeb, kui kõikide sõlmede jaoks $t_x\leq t_\phi$;
- (4) taastada registrid ja teostada märgitud sõlmede retiming 1 võrra (nihutades registrid vastavate sõlmede väljundi(te)st sisendi(te)), tagasi punkti (1);

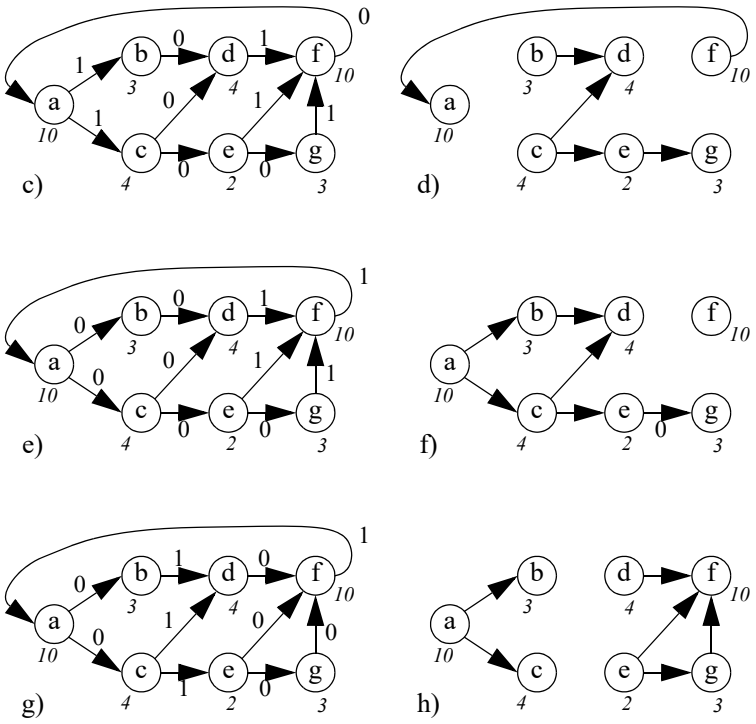
Sõlme retiming: Pole oluline, kui märgitud sõlme väljundis registrit pole - alati tekib mingil hetkel register, kui esialgne kirjeldus oli korrektne, st. tsüklis on vähemalt üks register. Või teisiti sõnastades - kõik teed lõpevad registriga ja retiming nihutab registri alati liiga aeglase tee lõpust (sisendite suunas) kohani, kus ajaline piirang on rahuldatud.

1. iteratsioon, sammud (1) & (2):
$t_a=10$, $t_b=3$, $t_c=4$, $t_d=8$, $t_e=6$, $t_g=9$, $t_f=\underline{19}$ (a&b joonisel).
samm (3): $t_f>15$ - sõlm *f* tuleb retime'da.
samm (4): register viiakse sõlme *f* väljundist (lahutades 1) tema sisenditesse (liites 1), vt. c).

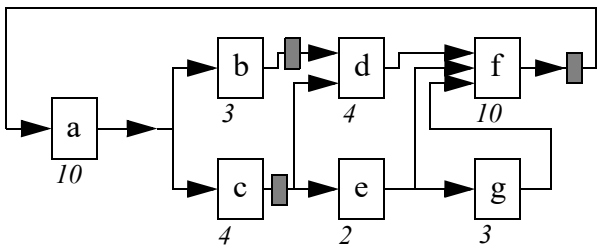




2. iteratsioon, sammuds (1) & (2):
t_a=<u>20</u>, t_b=3, t_c=4, t_d=8, t_e=6, t_g=9, t_f=10 (c&d joonisel).
samm (3): t_a>15 - sõlm *a* tuleb retime'da.
samm (4): register viiakse sõlme *a* väljunditest tema sisendisse (e).
3. iteratsioon, sammud (1) & (2):
t_a=10, t_b=13, t_c=14, t_d=<u>18</u>, t_e=<u>16</u>, t_g=<u>19</u>, t_f=10 (e&f joon.).
samm (3): t_d>15, t_e>15 and t_g>15 - sõlmed *d*, *e* ja *g* tuleb retime'da.
samm (4): register viiakse *d*, *e* ja *g* väljunditest nende sisenditeni (g).
4. iteratsioon, sammud (1) & (2):
t_a=10, t_b=13, t_c=14, t_d=4, t_e=2, t_g=5, t_f=15 (g&h joonisel).
samm (3): kõik t_x≤15 - ajalised piirangud on rahuldatud.



Lõplikus skeemis ei pea olema nelja registrit (4 serva kaaluga 1), sest sõlme c väljundis on on tegelikult ainult üks register.



2. Automaatide testitavuse alused

Automaatide testitavusel, nagu ka kombinatsioonskeemide testitavuse puhul, vaadeldakse üldjuhul kahte aspekti. Esiteks, seada automaat soovitud olekusse - seadejada. Teiseks, määrata kindlaks, millises olekus automaat on (oli) - kontrolljada. Mõlemal jada puhul on üldiseks juhuks olukord, kus jooksev olek pole teada. Seega peaks seadejada olema suuteline viima automaadi suvalisest olekust mingisse kindlasse soovitud olekusse ette määratud taktsammude jooksul ja kontrolljada peaks suutma mingi ettemääratud arvu sammudega suutma tekitada sellise väljundsignaalide jada, et automaadi oleks oleks üheselt identifitseeritav. Testide genereerimise lihtsustamiseks üritatakse tihti hakkama saada ainult seadejadadega - automaat viiakse soovitud olekusse ja siis kontrollitakse tema reaktsiooni. Seadejada võib samuti vaadelda kahes osas - suvalisest olekust esimesse olekusse (või mingisse muusse olekusse) ja esiemesest olekust kõikidesse teistesse olekutesse. Olulisem on just leida see seadejada esimene pool, sest muud olekud on esimesest olekust enamasti lihtsalt kättesaadavad.

Olgu antud joonisel kujutatud automaat. Väljundid pole olulised, sest vaja oleks leida just selline jada (või mitu jada), mis suvalisest olekust viib sama taktsammude jooksul esimesse

olekusse. Kokku on automaadil viis olekut ja esiteks oleks vaja leida kõikvõimalikud teed suvalisest olekust esimesse olekusse.

S1->S1: x1 peab olema 0 ja x2 pole oluline (edaspidi tähistus "0-"), seda jada saab kasutada hiljem, sest "0-" võimaldab hoida automaati soovitud aja olekus S1.

S2->S1: eksisteerib kolm erinevat teed:

- 3 sammu: S2-S4-S5-S1: "-1","--","--";
- 4 sammu: S2-S3-S4-S5-S1: "-0","-0","--","--";
- 3 sammu: S2-S3-S5-S1: "-0","-1","--".

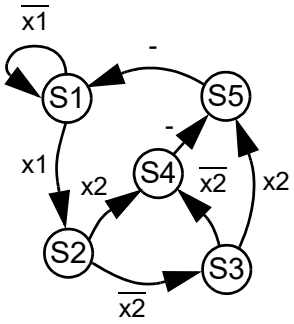
S3->S1: eksisteerib kaks erinevat teed:

- 3 sammu: S3-S4-S5-S1: "-0","--","--";
- 2 sammu: S3-S5-S1: "-1", "--".

S4->S1: üks tee: 2 sammu: S4-S5-S1: "--","--".

S5->S1: üks tee: 1 samm: S5-S1: "--".

i^t	s^t	s^{t+1}
$\overline{x1}$	S1	S1
x1		S2
$\overline{x2}$	S2	S3
x2		S4
$\overline{x2}$	S3	S4
x2		S5
1	S4	S5
1	S5	S1



Olekust S2 saab olekusse S1 vähemalt kolm sammuga, seega peab seadejada olema vähemalt kolm sammu (sisendkombinatsiooni) pikk. Ühtlasi tuleks leida ka teiste olekute jaoks jadad, mis on täpselt kolm sammu pikad (vastasel korral peab olema teda, millises olekus automaat on). Selliste jadade leidmiseks saab kasutada ootetsükleid (olek S1 antud näites) ja/või kombineerides erinevaid harusid. Üldiselt peaks arvestama sellega, et algselt on kasulik vaadelda kõikvõimalikke kombinatsioone ja neist valida välja sobivaim.

Jadad pikkusega kolm:

S1->S1: jada A - "0-", "0-", "0-" (S1-S1-S1-S1 ehk kolm sammu oodatakse olekus S1);

S2->S1: jada B - "-1", "--", "--" (S2-S4-S5-S1);

S2->S1: jada C - "-0", "-1", "--" (S2-S3-S5-S1);

S3->S1: jada D - "-0", "--", "--" (S3-S4-S5-S1);

S3->S1: jada E - "-1", "--", "0-" (S3-S5-S1-S1 ehk ootab ühe täiendava sammu olekus S1);

S4->S1: jada F - "--", "--", "0-" (S4-S5-S1-S1);

S5->S1: jada G - "--", "0-", "0-" (S5-S1-S1-S1).

Need jadad tuleks kokku kombineerida selliselt, et ei tekiks konflikte. Konflikt tähendab vastuolulisi sisendtingimusi. Näiteks on konfliktis jadad B ja D - esimesel juhul peab x2 olema 1, teisel juhul 0. Kontrollimine toimub lihtsa JA tehte abil - näiteks "-1" & "-0" == "-x" ehk $(x2)(\overline{x2})=0$.

Jadade kombinatsioonid:

- jada A annab "0-", "0-", "0-";
- jada B annab eelnevaga (läbi JA tehte) "01", "0-", "0-" ("0-","0-","0-"&"-1","--","--"); (jada C ei vaadelda, sest tegu on olekust S2-st väljuva alternatiivse teega, mis peab olema konfliktis jadaga B - tegu on teineteist välistavate teedega);
- jada D on eelnevaga konfliktis - "01", "0-", "0-"&"-0","--","--"="0x","0-","0-", st. selline jada pole võimalik, sest mingi sisend saab olla kas 0 või 1 (ja mitte mõlemad);
- jada E annab eelnevaga (A&B) "01", "0-", "0-" ("01","0-","0-"&"-1","--","0-");
- jada F annab eelnevaga (A&B&E) "01", "0-", "0-" ("01","0-","0-"&"--","--","0-"); ja
- jada G annab eelnevaga (A&B&E&F) "01", "0-", "0-" ("01","0-","0-"&"--","0-","0-").

Teine legaalne jada pikkusega kolm:
- jada A annab “0-”, “0-”, “0-”;
- jada C annab eelnevaga (A) “00”, “01”, “0-” (“0-”,“0-”,“0-”&“-0”,“-1”,“--”);
- jada D annab eelnevaga (A&C) “00”, “01”, “0-” (“00”,“01”,“0-”&“-0”,“--”,“--”);
  (jada E ei vaadelda, sest tegu on olekust S3-st väljuva alternatiivse teega);
- jada F annab eelnevaga (A&C&D) “00”, “01”, “0-” (“00”,“01”,“0-”&“--”,“--”,“0-”); ja
- jada G annab eelnevaga (A&C&D&F) “00”, “01”, “0-” (“00”,“01”,“0-”&“--”,“0-”,“0-”).

Kokku on kaks legaalset seadejada pikkusega kolm, mis viivad automaadi suvalisest olekust olekusse S1: [“01”, “0-”, “0-”] ja [“00”,“01”,“0-”].

Võib juhtuda, et minimaalse pikkusega legaalset jada pole võimalik leida, sest tekivad konfliktid. Või on automaat osa mingist suuremast süsteemist, mille muud komponendid määravad pikemad seadejadad. Olgu näiteks vajalikuks pikkuseks neli (kõik pikemad jadad on juba neljastest lihtsalt tuletatavad).

Jadad pikkusega neli:
S1->S1: jada A - “0-”, “0-”, “0-”, “0-” (S1-S1-S1-S1-S1);
S2->S1: jada B - “-1”, “--”, “--”, “0-” (S2-S4-S5-S1-S1);
S2->S1: jada C - “-0”, “-0”, “--”, “--” (S2-S3-S4-S5-S1);
S2->S1: jada D - “-0”, “-1”, “--”, “0-” (S2-S3-S5-S1-S1);
S3->S1: jada E - “-0”, “--”, “--”, “0-” (S3-S4-S5-S1-S1);
S3->S1: jada F - “-1”, “--”, “0-”, “0-” (S3-S5-S1-S1-S1);
S4->S1: jada G - “--”, “--”, “0-”, “0-” (S4-S5-S1-S1-S1);
S5->S1: jada H - “--”, “0-”, “0-”, “0-” (S5-S1-S1-S1-S1).

Kuuest (1x3x2x1x1) võimalikust kombinatsioonist on kolm konfliktivabad:
- A&B&F&G&H: [“01”, “0-”, “0-”, “0-”];
- A&C&E&G&H: [“00”, “00”, “0-”, “0-”]; ja
- A&D&E&G&H: [“00”, “01”, “0-”, “0-”].

Neist esimene ja kolmas langevad esimese kolme sammu osas kokku kolmeste seadejadadega. Erinevus seisneb selles, et on lisandunud täiendav ootesamm olekus S1.