Digitaalsüsteemid / Digital Systems

IAS0150 – 6 EAP 4 2-2-0 E S

https://moodle.taltech.ee/course/view.php?id=34891 – IAS0150 Digitaalsüsteemid (2025)

Peeter Ellervee

ICT-526 620 2258 511 3631 LRV@ati.ttu.ee

https://ati.ttu.ee/~lrv/

- **16 loengut, 16 praktikum-harjutust**
- **kaks kodutööd, trükkplaadi projekteerimine, neli testi**
- **kirjalik eksam, eeldusteks esitatud kodutööd ja osalemine praktikumides (testid!)**
- **koondhinne (max 100) – kodutööd 25+15, trükkplaadilabor 10, testid 20, eksam 30**
  - **hinded: 1: 50-59, 2: 60-69, 3: 70-79, 4: 80-89, 5: 90-...**

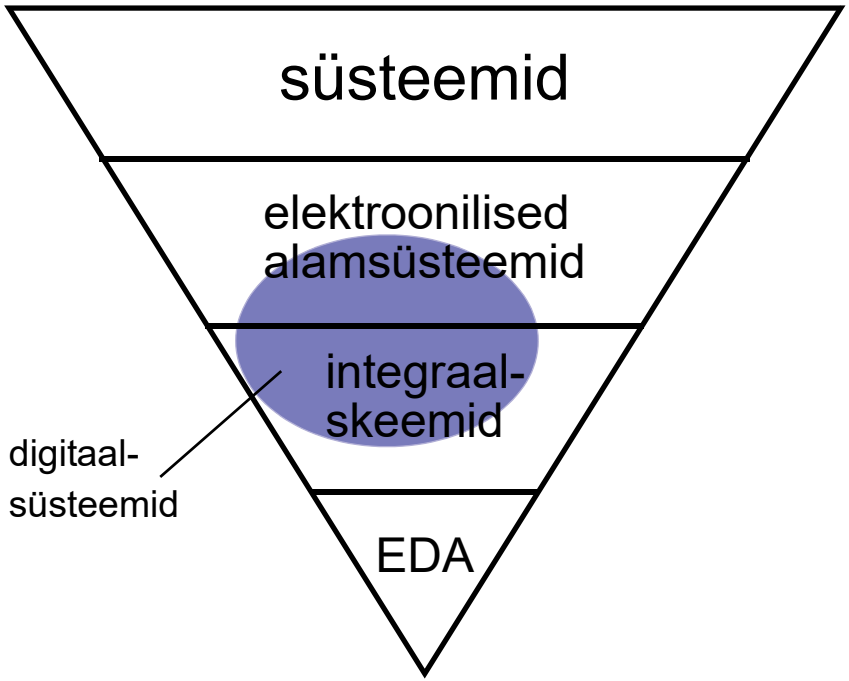
Digitaalsüsteemid

- **Süsteemid**
  - ***NB! Piirid pole täpselt paigas...***
  - **Mehhaanikasüsteem – *liikumine***
  - **Elektrisüsteem – *elektrienergia***
  - **Elektroonikasüsteem – *infotöötlus***
    - **Analoogsüsteem – signaalide esitamine ja töötlus pidevate suurustena signaalide väärtused: 0...5 V, -10...+10 mA, jne.**
    - **Digitaalsüsteem – signaalide esitamine ja töötlus diskreetsete suurustena signaalide väärtused: 0/1, tõene/väär, true/false, high/low, jne.**
  - **Sardsüsteem (embedded system)**
    - **Kaasajal peamiselt (hajutatud) digitaalsüsteem, mis sisaldab nii analoog-alamsüsteeme aga ka mehhaanilisi ja elektrilisi komponente**
    - **Suvaline digitaalsüsteem sisaldab alati analoog, elektrilisi ja mehhaanilisi komponente – nt. nivoomuundurid, toide, lülitid, ...**
    - **Küberfüüsikalised süsteemid – süsteem + keskkond, kasutajad, ...**

Digitaalsüsteemid

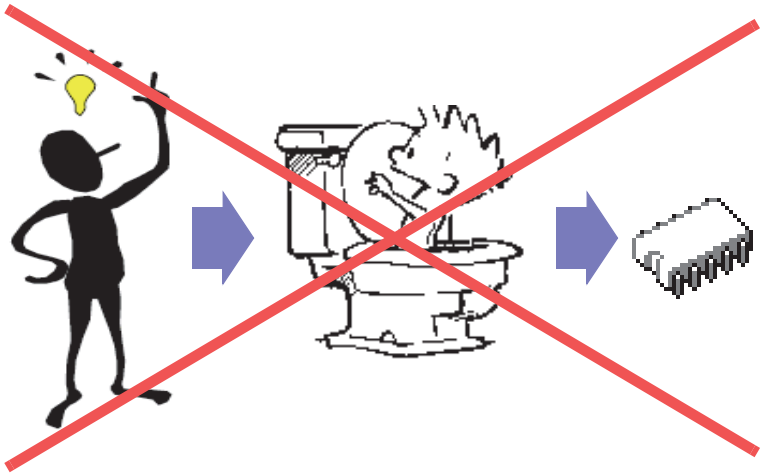
- **Digitaalsüsteem**
  - **digitaalne andmetöötlus**
  - **andmeosa, juhtosa, sisend/väljund**
- **Primaarturud**
  - **infosüsteemid**
  - **telekommunikatsioon**
  - **laiatarbe-elektroonika**
- **Sekundaarturud**
  - **süsteemid (nt. transport)**
  - **tootmine (nt. robotid)**
- **VLSI tehnoloogiate rakendused**

  VLSI – Very Large Scale Integration



Digitaalsüsteemide projekteerimine e. disain

- **Müüt – kõrgtaseme projekteerimine on ainult üks samm**



- **Vajalikud on iteratsioonid**
  - **funktsionaalsus**
  - **disaini eesmärgid**

PDSA - Plan, Do, Study, Act
PDCA - Plan, Do, Check, Adjust

Disaini põhietapid

- **Süsteemi disain – System design**
  a.k.a. **Arhitektuuri süntees – Architectural-level synthesis**
  - **kirjeldus / spetsifikatsioon → plokk-skeem**
  - **makroskoopilise struktuuri määramine *ehk***
    kuidas on peamised ühendusplokid omavahel ühendatud

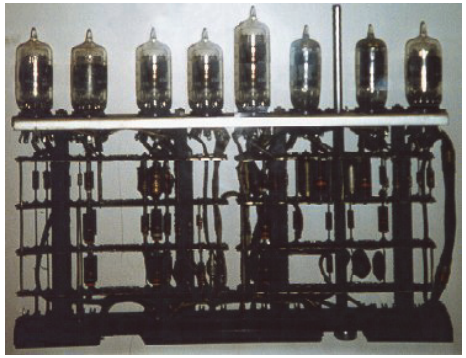
- **Loogikadisain – Logic Design**
  - **plokk-skeem → loogikalülid**
  - **mikroskoopilise struktuuri määramine *ehk***
    kuidas on loogikalülid omavahel ühendatud

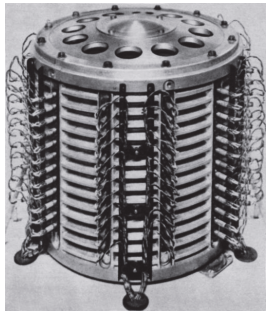
- **Füüsiline disain – Physical design**
  a.k.a. **Geomeetria süntees – Geometrical-level synthesis**
  - **loogikalülid → transistorid, ühendusjuhtmed, mikroskeem**

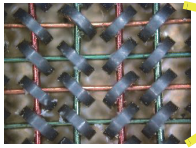
Abstraktsioonitasemed

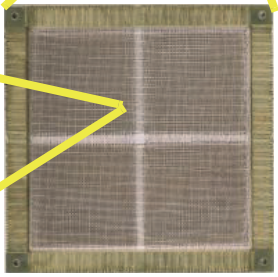
Tase	Abstraktsioon	Töövahendid
süsteem	käitumine ruumis ja ajas kui suuniste, ajastuse ja s/v spetsifikatsioonid	plokk-skeemid, diagrammid, kõrgtaseme (programmeerimis) keeled
arhitektuur	funktsionaalsete olemite üldine süsteem (organisatsioon)	riistvara kirjelduskeeled, moodulite paigalduse planeerimine takt-sageduse ja pinna ennustamiseks
registersiirded	andmevoo funktsionaalsete moodu-lite ja mikrokäskude sidumine	süntees, simuleerimine, verifitseerimine, testi analüüs, ressursside vajaduse hinnang
funktsionaalsed moodulid	primitiivsed operatsioonid ja juhtimisviisid	teegid, mooduli generaatorid, skeemisisestus, testimine
loogika	loogikalülide Boole' funktsioonid	skeemisisestus, süntees ja simuleerimine, verifitseerimine, PLA vahendid
lülitused	transistorahelate elektrilised omadused	RC leidmine, ajastuse verifitseerimine, elektriline analüüs
kristalli pind	geomeetrilised piirangud	pinna redaktor, ahelate ekstraheerimine, DRC, paigaldus ja trasseerimine (ruutimine)

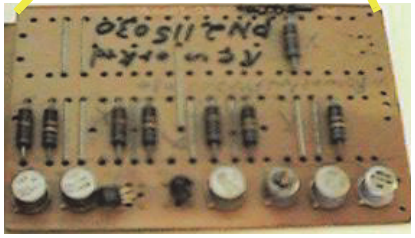
Digitaalsüsteemide realiseerimine – ajalugu ja tänapäev

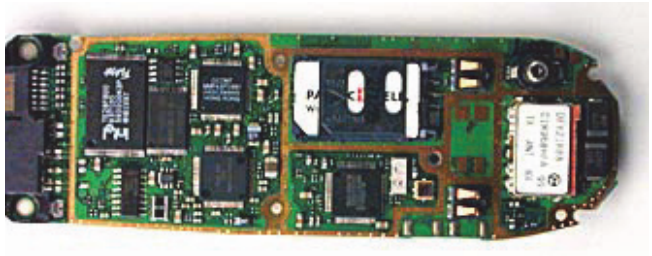






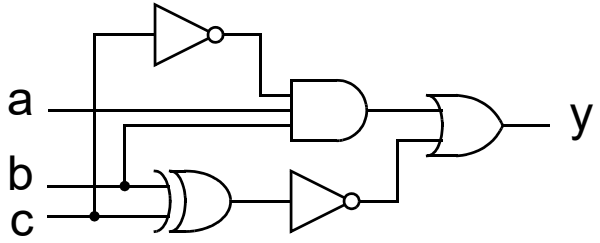


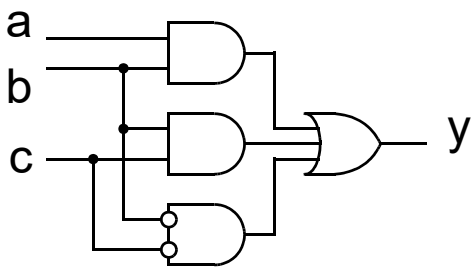




Algoritmist skeemini

- **Peatuspunkt / sünkroniseerimine**
  - **olek (automaadi olek)**
  - **mäluelemendid**
  - **takteerimine**
- **Funktsioon**
  - $y = \overline{(\,b \oplus c\,)} + a\,b\,\overline{c}$
- **Normaalkuju**
  - $y = a\,b + b\,c + \overline{b}\,\overline{c}$
- **Tõeväärtustabel**
  - **mitu sisendit & mitu väljundit**





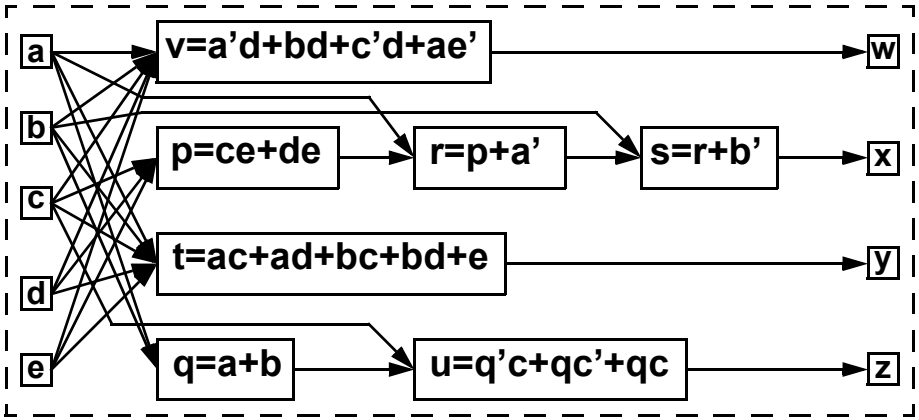
abc	y
000	1
001	0
010	0
011	1
100	1
101	0
110	1
111	1

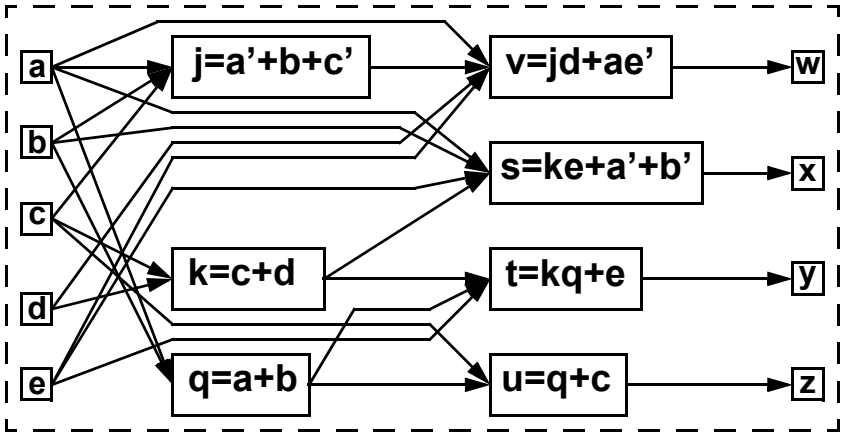
Algoritmist skeemini (2)

- **Lõplik funktsioon**

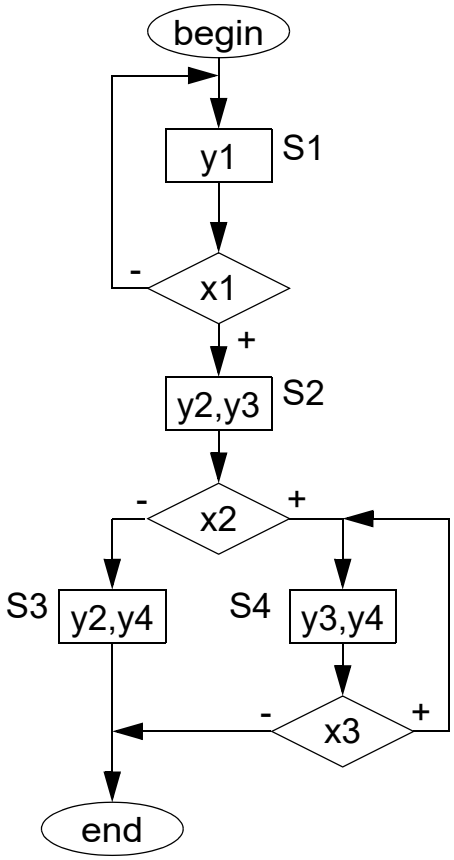
```
j=a'+b+c'
k=c+d
q=a+b
s=ke+a'+b'
t=q+c
u=q+c
v=jd+ae'
```

- **Funktsioone/loogikalülisid – 7 ja 7**
- **Literaale – 33 ja 20**
- **Viide – 3 ja 2 (sõlmi)**
- **Viide – 9 ja 7 (sõlmi+literaale)**





Algoritmist skeemini (3)



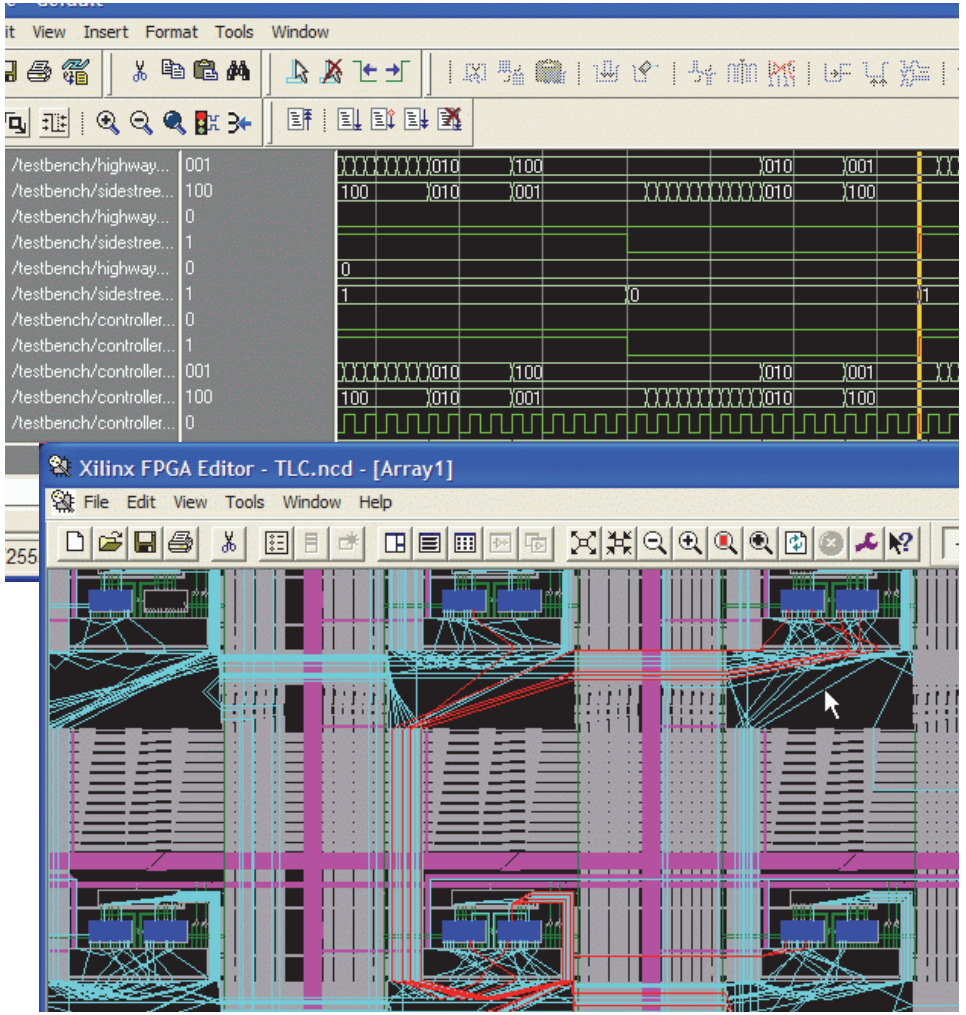
i^t	s^t	q1 q0	s^{t+1}	q1 q0	o^t
x1’	S1	0 0	S1	0 0	y1
x1			S2	0 1	
x2’	S2	0 1	S3	1 1	y2,y3
x2			S4	1 0	
-	S3	1 1	S1	0 0	y2,y4
x3’	S4	1 0	S1	0 0	y3,y4
x3			S4	1 0	

- Skeem
- $n0=\overline{q1}q0$ $n1=q1\overline{q0}$
- $d1=n0+x3n1$ $d0=x1y1+\overline{x2}n0$
- $y1=\overline{q1}\ \overline{q0}$ $y2=q0$
- $y3=n0+n1$ $y4=q1$

Projekteerimine tänapäeval

- **Riistvara kirjelduskeel**

```
--
-- Highway is green, sidestreet is red.
--
if  sidestreet_car = NoCar  then
  wait until  sidestreet_car = Car;
end if;
-- Waiting for no more than 25 seconds ...
if  highway_car = Car  then
  wait until  highway_car = NoCar  for 25 sec;
end if;
-- ... and changing lights
highway_light <= GreenBlink;
wait for 3 sec;
highway_light <= Yellow;
sidestreet_light <= Yellow;
wait for 2 sec;
highway_light <= Red;
sidestreet_light <= Green;
```



Turg e. $$$

- **Projekteerimise maksumus**
  - ***projekteerimisaeg*** **& kristallide** ***tootmise*** **hind**
  - **suured** ***kapitalimahutused***
  - **pea-aegu võimatu** ***parandada***
- **Muudatuste kõrge hind**
  - **suured** ***tootmismahud*** **rentaablimad**
  - ***null-defekti*** **on äärmiselt oluline**
  - ***turusuundumuste*** **järgimine oluline**
- **Hind pöördvõrdeline tootmismahuga**
  - **üldotstarbelised protsessorid - odav kuid pole alati kasutatav**
  - **ASIC (Aplication-Specific Integrated Circuits) – häälestamine vastavalt vajadusele (nt. telekommunikatsioon)**
  - **prototüübid – väljatöötluses on paindlikkus äärmiselt oluline**
  - **spetsrakendused (nt. sateliidid)**
- **Rekonfigureeritavus**
  - **paindlikud tooted, võimalus modifitseerida töötavat skeemi**

Automatiseeritud projekteerimine

- Gordon Moore seadus (1965)
- Edusammud tehnoloogias
  - väiksemad skeemid
  - suurem jõudlus
  - rohkem transistore kristallil
- Suurem integratsiooniaste
  - kompleksemad süsteemid
  - arvutusvõimsuse odavnemine
  - suurem töökindlus
- Automatiseerimine võimaldab:
  - uusimate tehnoloogiate kasutamist
  - vähendada projekteerimiskulutusi
  - kiirendada projekteerimist

Automatiseerimise ajalugu (natuke idealistlik vaade)

- **Alberto Sangiovanni-Vincentelli, 2003**
  - ***Age of Gods* – 1964-1978**
  - ***Age of Heroes* – 1979-1993**
  - ***Age of Men* – 1993-...(2002)**

- **Projekteerija jõudlus**
  - **1990 – 4 K elementi / aastas / disainer**
  - **1995 – insener teeb kõik (RTL→GDSII) – 9.1K**
  - **2000 – suurte plokkide korduvkasutus, süntees (RTL→GDSII) – 91K**
  - **2005 – käitumuslik ja arhitektuurne tase, riist- ja tarkvara (koos)disain – 200K**
  - **2010 – väga suurte plokkide korduvkasutus (multi-tuumad) – 1200K**
  - ***Tänapäev ja tulevik – riist- ja tarkvara koosverifitseerimine, täidetav spetsifikatsioon jne.***

Miks kahendloogika?

- Digitaali eelised
  - Tulemuste korratavus – samad sisendväärtused annavad alati sama tulemuse
    - analoog – temperatuur, toitepinge, vananemine, ...
  - Projekteerimise lihtsus – loogikafunktsioonid, optimeerimisalgoritmid
  - Paindlikkus ja funktsionaalsus – erinevad algoritmid, sama funktsionaalsus (võimsustarbe, kiiruse, suuruse jne. erinevused)
  - Programmeeritavus – programmeerimiskeeled / riistvara kirjelduskeeled
  - Töökiirus
  - Turu ja tehnoloogia areng – ränikiipide/-tehnoloogia skaleeritavus/korratavus
- Analoogi eelised
  - Differentsiaalvõrrandite realiseerimine
  - Energeetiline efektiivsus, kõrge töösagedus
- Mitmevalentne loogika – rohkem kui kaks diskreetset väärtust
  - Suurem infotihedus – nt. 4-, 8- ja 16-valentsed mälud
  - Boole'i algebra edasiarendus – funktsioonide süsteemi minimeerimine

Dgitaalsüsteem – disaininäide #1

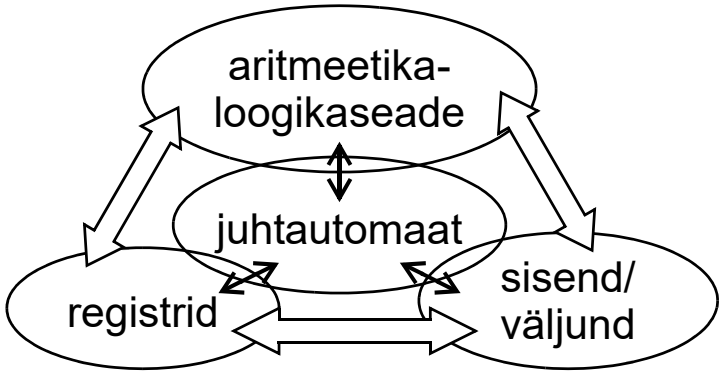
- Neli kahend-sisendit ja -väljundit – nt. 4 lülitit (S1-S4) ja 4 valgusdioodi (L1-L4)
- Sisendite muutumine muudab väljundeid
  - kui S1=1 & S2=0, siis L1←1, muidu L1←0
  - kui S1=0 & S3↑, siis V++ (V[1]=L2, V[0]=L3)
  - kui S1=1 & S2=1 & S4↓, siis L4← ¬L4

- Võimalik programm

```
int s3p=0, s4p=0, v=0; l4=0;
while (1) {
  if (s1&!s2) l1=1; else l1=0;
  if (!s1&((s3^s3p)&s3)) v++;
  if (v>3) v=0;
  l2=v/2; l3=v%2;
  if (s1&s2&((s4^s4p)&!s4)) l4~=l4;
  s3p=s3; s4p=s4; wait_100ms();
}
```

  - s1…s4 – lülitid == DIP1…DIP4 programmis
  - l1…l4 – LED-d (tuled) == leds – kõik 4 ühes sõnas!
  - üksikute bittidega manipuleerimine?

- Protsessor / kontroller
  - sisendid/väljundid
  - vahetulemused
  - töötlus- e. arvutus-sõlm
  - juht-osa



Disaininäide #1 – mikrokontroller

- **Esialgne programm – leds: and -> '0'; or -> '1'; dubleeritud lugemine**

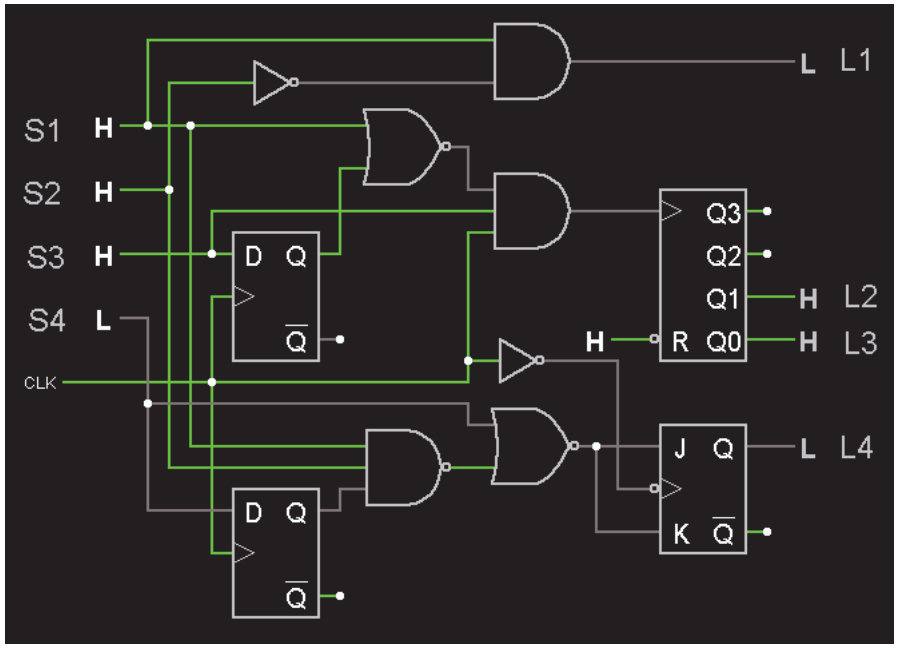
```
char leds=0,dip3p=0,dip4p=0,v=0;     // ROM – 548 / RAM – 81
while (1) {
if (DIP1&&!DIP2) leds|=0b1000; else leds&=0b0111;
  if (!DIP1&&((DIP3^dip3p)&&DIP3)) v++;
  if (v>3) v=0;
  leds=(leds&0b1001)|(v<<1);
  if (DIP1&&DIP2&&((DIP4^dip4p)&&!DIP4)) leds^=0b0001;
  dip3p=DIP3; dip4p=DIP4; led_out(leds); delay_100ms;
}
```

- **Optimeeritud programm [~~Shannoni arendus]**

```
char leds=0,dip3p=0,dip4p=0,v=0; // ROM – 518 / RAM – 81
while (1) {
  if (DIP1) {
    if (DIP2) { if ((DIP4^dip4p)&&!DIP4) leds^=0b0001; leds&=0b0111; }
    else leds|=0b1000;
  }
  else {
    if ((DIP3^dip3p)&&DIP3) v++;       if (v>3) v=0;
    leds=(leds&0b0001)|(v<<1);
  }
  dip3p=DIP3; dip4p=DIP4; led_out(leds); delay_100ms;
}
```

Disaininäide – skeem

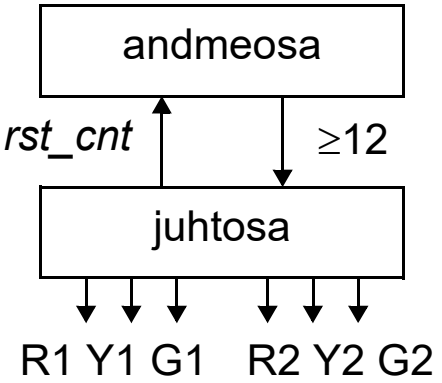
- Tegevused
  - kui S1=1 & S2=0, siis L1←1, muidu L1←0
  - kui S1=0 & S3↑, siis V++ (V[1]=L2, V[0]=L3)
  - kui S1=1 & S2=1 & S4↓, siis L4← ¬L4
- Kolm parallelset osa
- Kombinatsioon-skeem
  - L1 ← S1 & !S2
- Loendur + loogika
  - D-tiger – S3p ← S3
  - v++ ← !S1 & !S3p & S3
  - L2 ← V[1]; L3 ← V[0]
- T-triger + loogika
  - D-tiger – S4p ← S4
  - !L4 ← S1 & S2 & S4p & !S4
- Asünkroonsed pisihädad
  - takti ja signaali muutuse sünkroniseerimine
  - vt. S3 muutmist…



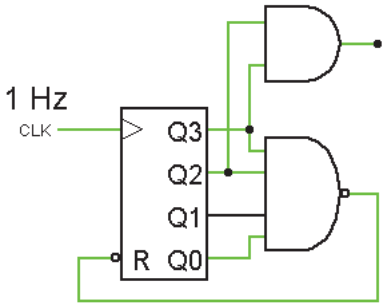
`https://ati.ttu.ee/~lrv/IAS0150/switch4led4.txt`

Näide #2 – valgusfoor kui digitaalsüsteem

- **Digitaalsüsteem – andmeosa + juhtosa**
- **Kogutsükkel 30 sek., andurid puuduvad**
  - roheline 12 sek. punane
  - kollane 3 sek. kollane+punane
  - punane 12 sek. roheline
  - kollane+punane 3 sek. kollane
- **Juhtosa – automaat**
  - I={<12,≥12}, O={R1,Y1,G1,R2,Y2,G2,rst_cnt(?)}
- **Andmeosa – loendur (0...14)**
  - 0…14 → 4 bitti (0…15!)
  - asünkroonne nullimine kui loendur == 15 (e. 4-NAND)
  - 12 sek. == 0…11 / 3 sek. == 12…14 e. =12
  - ≥12 == 1100+1101+1110 (+1111 määramatusena)
  - ≥12 == 11-- (e. 2-AND)
  - rst_cnt vajalikkus? – sõltub juhtosa “tarkusest”

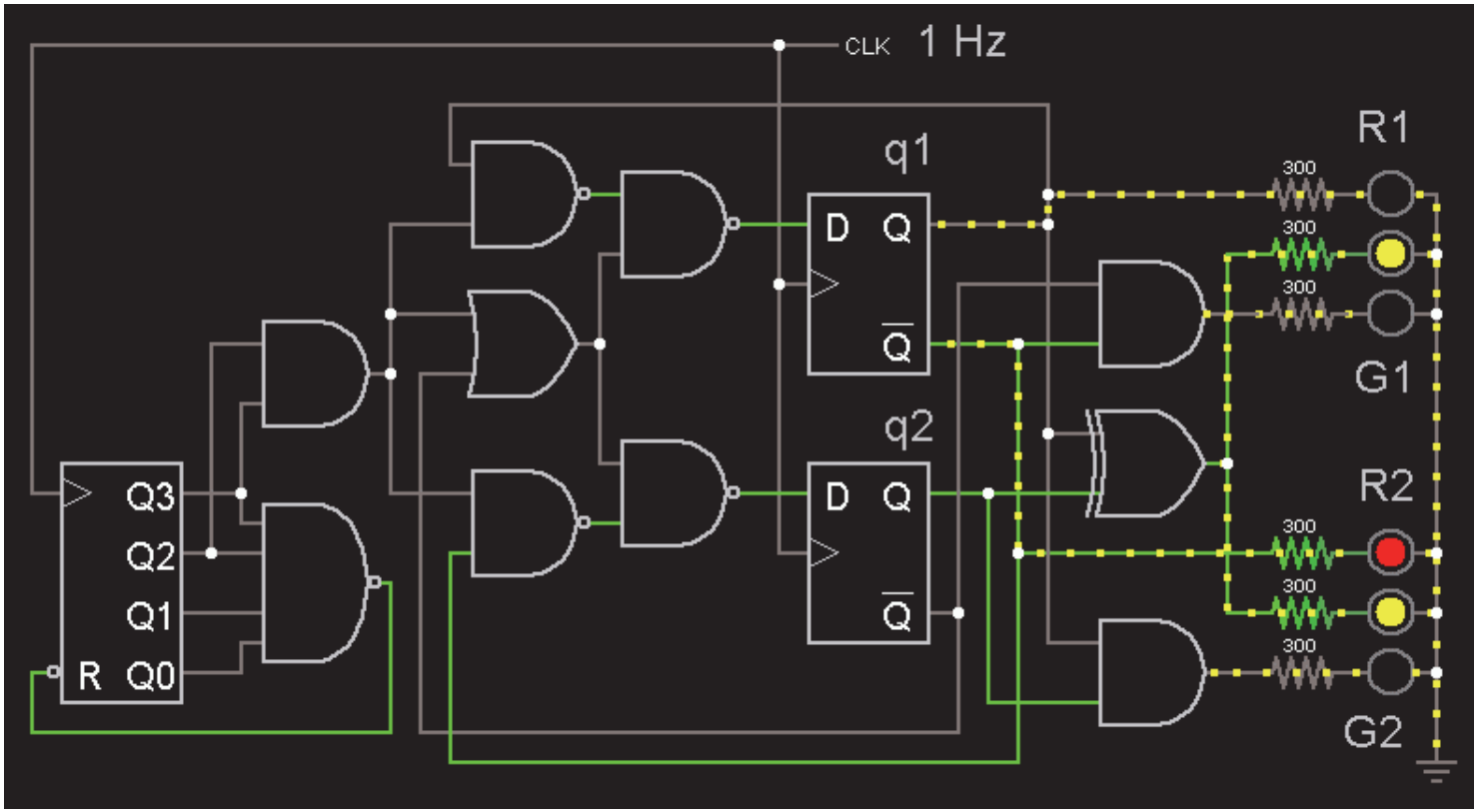


Andmeosa



`https://ati.ttu.ee/~lrv/IAS0150/tlc-datapath.txt`

Valgusfoor – tulemus



https://ati.ttu.ee/~lrv/IAS0150/tlc-applet.txt

Näide #3 – diferentsiaalvõrrand

$$\frac{d^2y}{dx^2} + 5\frac{dy}{dx}x + 3y = 0$$

- Kuidas jõuda algoritmist realisatioonini?

- Tarkvaraline realisatsioon
  - protsessor ette antud
  - leida (assembler)käskude jada

- Riistvaraline realisatsioon
  - protsessor pole ette antud
  - leida (mikro)käskude jada

```
process
  variable a,dx,x,u,y,x1,y1: integer;
begin
  cycles(sysclock,1); a:=inport;
  cycles(sysclock,1); dx:=inport;
  cycles(sysclock,1); y:=inport;
  cycles(sysclock,1); x:=inport;
  cycles(sysclock,1); u:=inport;
  loop
    cycles(sysclock,7);
    x1 := x + dx; y1 := y + (u * dx);
    u := u-5 * x * (u * dx) - 3 * y * dx;
    x := x1; y := y1;
    exit when not (x1 < a);
  end loop;
end process;
```

Tarkvaraline realisatsioon == kompileerimine

- **Diferentsiaalvõrrand**

$$\frac{d^2y}{dx^2} + 5\frac{dy}{dx}x + 3y = 0$$

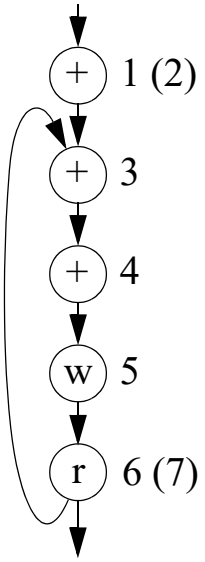
```
{
  sc_fixed<6,10> a,dx,y,x,u,x1,x2,y1;
  while ( true ) {
    wait(); a=inport.read();
    wait(); dx=inport.read();
    wait(); y=inport.read();
    wait(); x=inport.read();
    wait(); u=inport.read();
    while ( true ) {
      for (int i=0;i<7;i++) wait();
      x1 = x + dx;  y1 = y + (u*dx);
      u = u - 5*x*(u*dx) - 3*y*dx;
      x = x1; y = y1;
      if (!(x1<a)) break;
    }
    outport.write(y);
  };
}
```

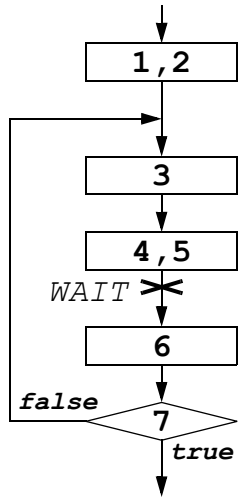
```
# R1:a, R2:dx, R3:y, R4:x, R5:u,
# R6:x1, R7:x2, R8:y1, R9:tmp
    ...
_loop_$32:
    ADD.fx   R6, R4, R2     # x1=x+dx
    MUL.fx   R9, R5, R2     # tmp=u*dx
    ADD.fx   R8, R3, R9     # y1=y+tmp
    MUL.fx   R9, R4, R9     # tmp=x*tmp
    MUL.fx   R9, R9, $5     # tmp=5*tmp
    SUB.fx   R5, R5, R9     # u=u-tmp
    MUL.fx   R9, R3, R2     # tmp=y*dx
    MUL.fx   R9, R9, $3     # tmp=3*tmp
    SUB.fx   R5, R5, R9     # u=u-tmp
    ADD.fx   R4, R6, $0     # x=x1
    ADD.fx   R3, R8, $0     # y=y1
    SUB.fx   R9, R6, R1     # tmp=x1-a
    JMP.neg  _loop_$32      # ...break
    ...
```

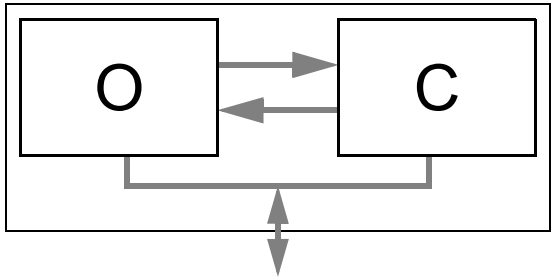
Näide #4 – otsimine mälust

```
...
base := ... + ... ;
l1: for i in 1 to max_address loop
  x := memory(base+i);
  exit l1 when x = x_required;
end loop l1;
...
```

```
...
base := ... + ... ;                -- 1
i := 0;                            -- 2
loop
  i := i + 1;                      -- 3
  -- x := memory(base+i);
  tmp := base + i;                 -- 4
  address <= tmp;                  -- 5
  wait on clk until clk='1';       --(5)
  x := data;                       -- 6
  exit when x = x_required;        -- 7
end loop;
...
```

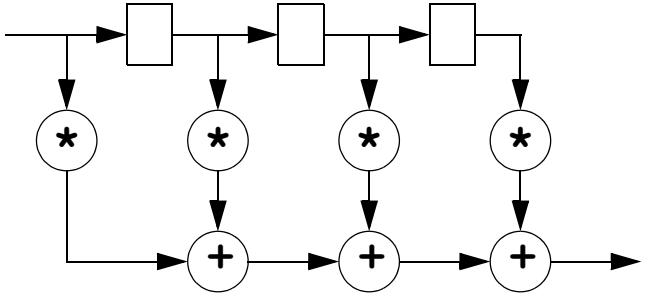


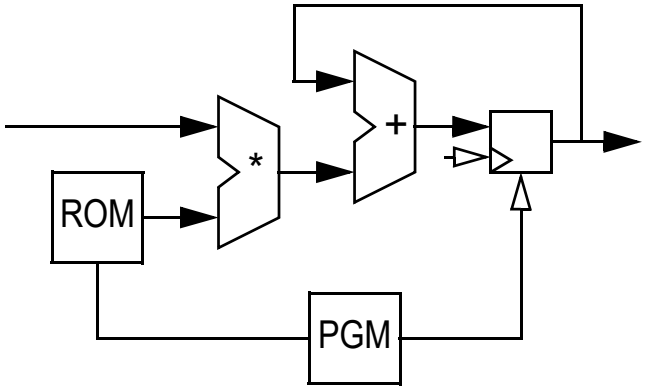


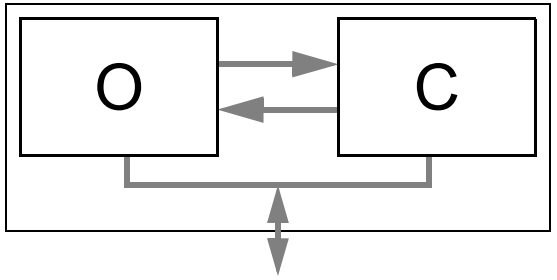


Näide #5 – FIR filter

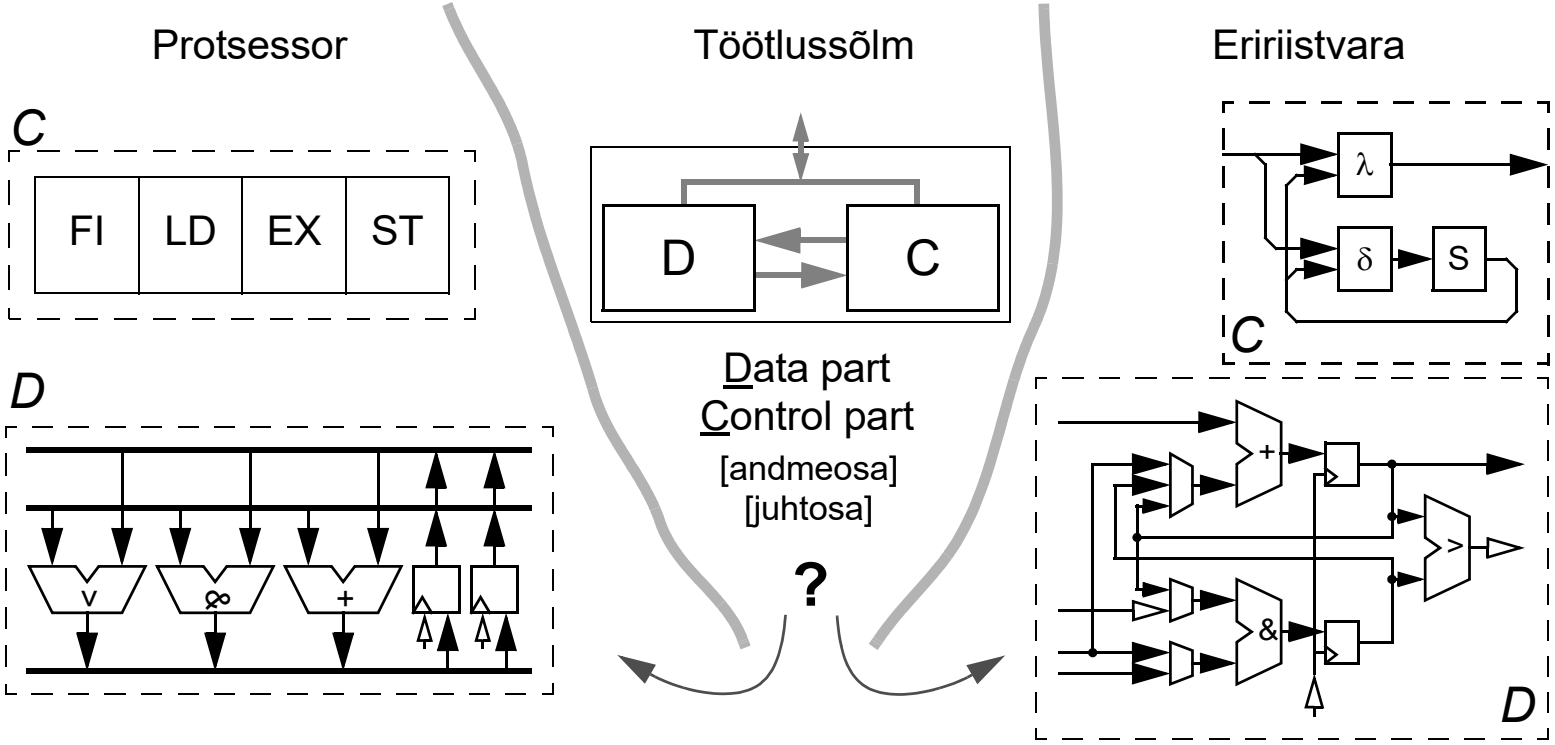
```
...
x=c0*i(0)+c1*i(1)+c2*i(2)+c3*i(3);
...
```







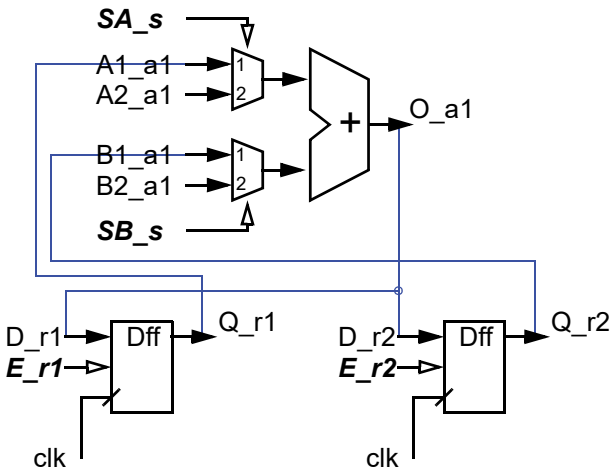
Tarkvara või Riistvara?



- **Tarkvara**
  - **universaalne andmetee – aeglane, suured mõõtmed, energianäljane**
  - **universaalne algoritm – äärmiselt paindlik, kuid nõuab palju ruumi (bitte)**
- **Riistvara**
  - **fikseeritud andmetee – kiire, kompaktne, väike energiatarve**
  - **paindlikkus määratud juhtautomaadiga, algoritm sageli fikseeritud**

Andme- ja juhtosa

- **Andmeosa (operatsioonautomaat)**
  - **andmete töötlus (operatsioonid e. arvutamine)**
    - **kombinatsioonskeemid (loogikafunktsioonid)**
  - **andmete salvestamine (mälu)**
    - **registrid (mäluelemendid)**
- **Juhtosa (juhtautomaat)**
  - **operatsioonide (tingimuslik) järjestamine**
    - **(eelmiste) operatsioonide tulemused**
    - **välised tingimused (sisendsignaalid)**
- **Algoritm**
  - **operatsioonide järjestus ~~ mikroprogramm**
- **Register-siirete tase**
  - **Register-Transfer Level (RTL)**
  - **register → kombinatsioonskeem → register → ...**



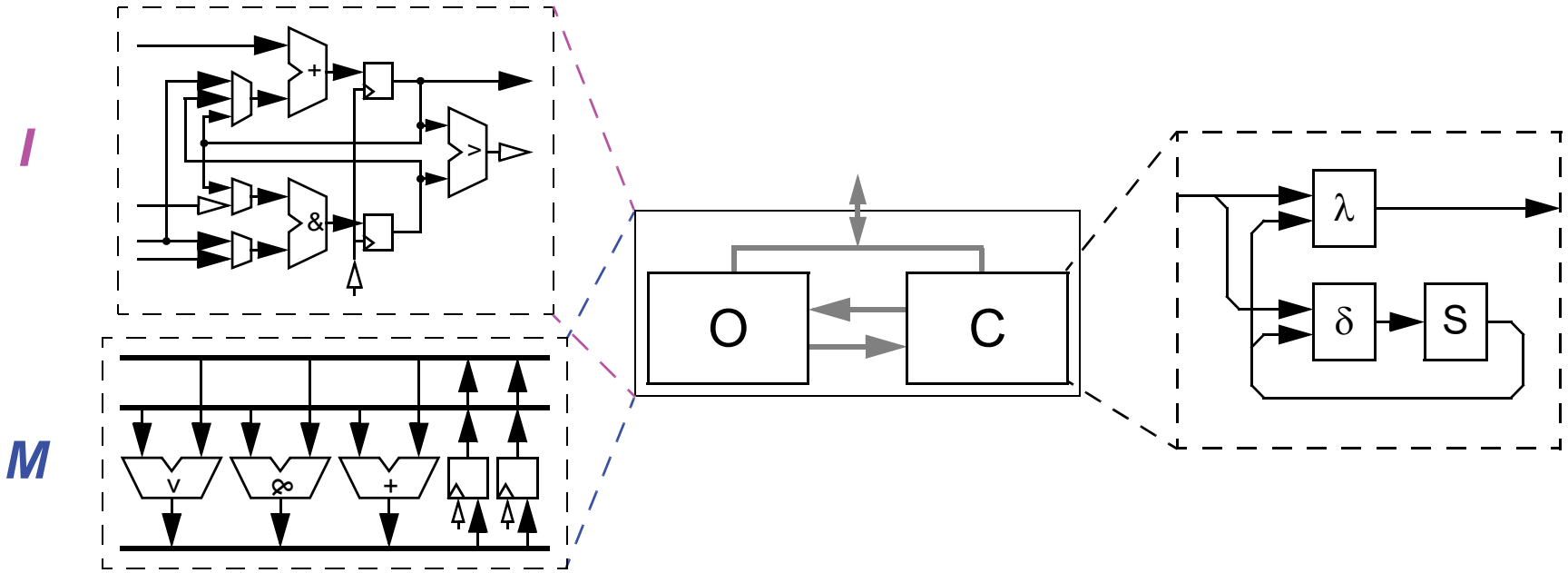
O_a1≡D_r1≡D_r2 Q_r1≡A1_a1
Q_r2≡B1_a1 n≡A2_a1 m≡B2_a1

x = 2 * n + 2 * m	SA_s	SB_s	E_r1	E_r2
1: r2 <- n	2	0	0	1
2: r2 <- n + r2	2	1	0	1
3: r1 <- m	0	2	1	0
4: r1<- r1 + m	1	2	1	0
5: r1 <- r1 + r2	1	1	1	0

SA_s & SB_s – väärtus 0 annab väljundisse “00...00”

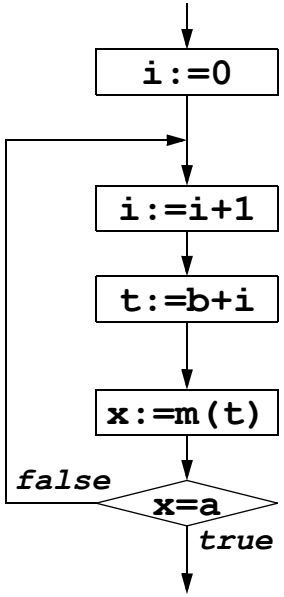
Andmeosa struktuur

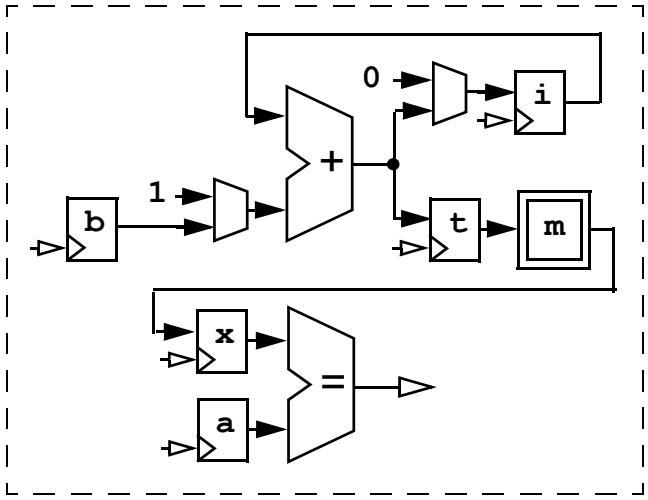
- I-automaat (~~eririistvara)
  - võimaldab üheaegselt sooritada kõiki funktsionaalselt ühitatavaid operatsioone (mikrokäske)
  - iga registri sisendis kombinatsioonskeem
- M-automaat (~~protsessor)
  - iga unikaalse operatsiooni jaoks on oma täitursõlm, üheaegselt võimalik täita tüübilt erinevaid operatsioone
  - andmesiinid



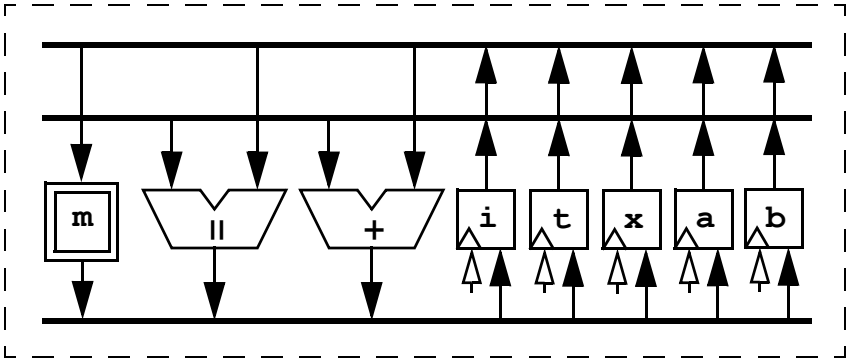
Andmeosa – eridisain või üldotstarbeline?

[Otsimine mälust]





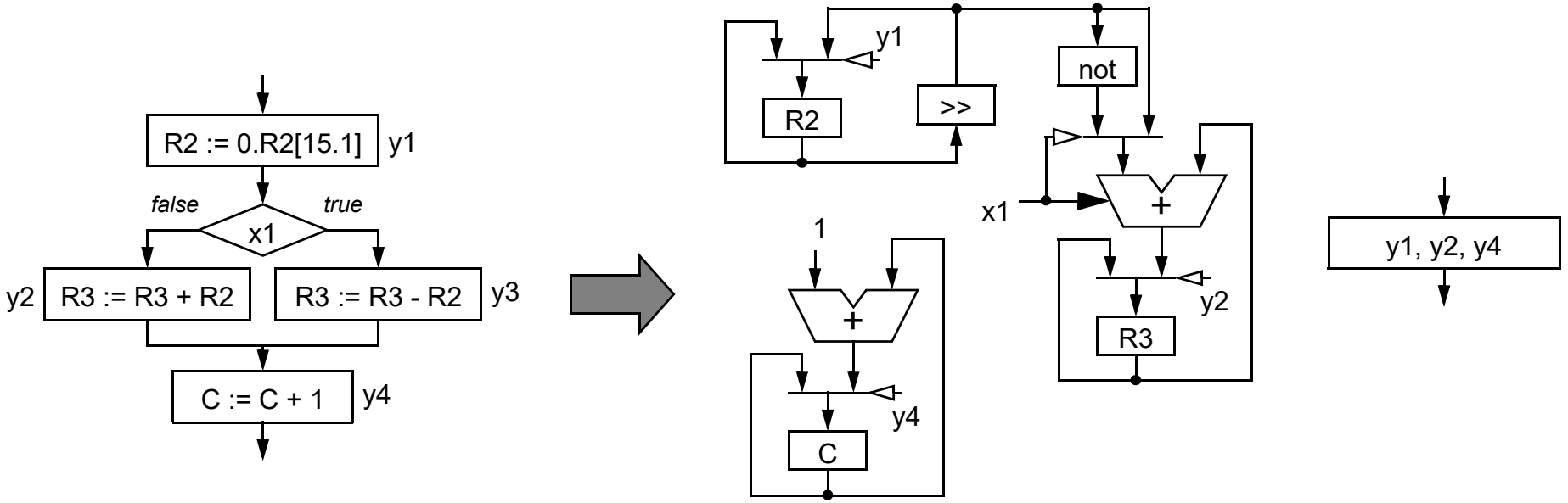
eridisain (e. *I*)
+ kiire
+ väike
-- paindumatu



üldotstarbeline (e. *M*)
++ paindlik
- aeglane
-- suur

Realisatsiooni optimaalsus?

- **operatsiooni hind riistvaras**
  - **nihutamine == traatide teisiti viimine**
  - **liitmist ja lahutamist teostab sama seade – summaator (+ülekanne)**
- **sõltumatud operatsioonid võib täita korraga**
  - **tegelike andmesõltuvuste analüüs – nt. R2 nihutamise tulemus**



Näide #6 – algoritmi realiseerimise optimaalsus

- Algoritm kui programm
  - operatsioonid – andmeosa / järjestus ja tingimused – juhtosa

- GCD (Greatest Common Divisor) – suurim ühistegur

```
while ( x != y ) {
  if ( x < y ) y = y - x;
  else         x = x - y;
}
```

  - operatsioonid – võrdlused ja lahutamine
  - konkreetse operatsiooni realiseerimine?
    - A < B == A - B < 0 / A != B == A - B != 0 – kõike saab teha lahutajaga?!
    - A - B == A + $\overline{B}$ + 1 – lahutaja == liitja + invertorid (ja ülekanne 1)
  - sama riistvara kõikidele operatsioonidele või korraga arvutamine?
    - kiirus või suurus? [ja kas see ongi probleem?]

Andmete esitamine

- Kümnendarvude esitamine kahendkoodis (kahendarvudena)
  - Arvutamine – liitmine, lahutamine, korrutamine, jagamine
  - Teisendamine – sisend/väljund 10-nd kujul
  - Erinevad arvuformaadid – täisarvud, murdarvud jne.
- Täiendkood
  - Kõige levinum, positsiooniline – $N = \sum_{i=-m}^{n-1} b_i \cdot 2^i$
  - Negatiivne arv – -A == $\overline{A}$+1

- Täisarvud: $116_{10} == 01110100_2$ $-116_{10} == 10001100_2$ [n=8, m=0]
- Püsikomaarvud: $4{,}125_{10} == 0100.0010_2$ [n=4, m=4]
- Ujukomaarvud: $M*2^E$ $4{,}125_{10} == 0{,}515625*2^3 == 0.0011.1000010$
  - S.E.M == sign.exponent.mantissa (märk.aste.mantiss) – E=-8..+7, M=0,5..0,(9)

Digitaalsüsteem

- **Digitaalsüsteem = andmeosa + juhtosa**