Digitaalsüsteem

- **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, ...**

Disainināide

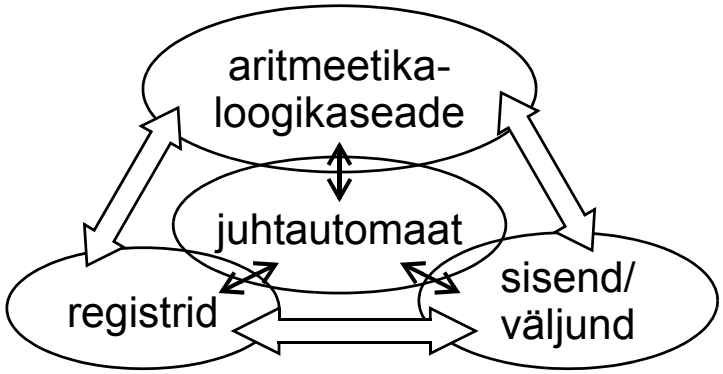
- Neli kahend-sisendit ja -väljundit – nt. 4 lülitit (S1-S4) ja 4 valgusdioodi (L1-L4)
- Sisendite muutumine muudab väljundeid
  - kui S1=1 & S2=0, siis L1←1, muidu L1←0
  - kui S1=0 & S3↑, siis V++ (V[1]=L2, V[0]=L3)
  - kui S1=1 & S2=1 & S4↓, siis L4← ¬L4

- 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 – mikrokontroller

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

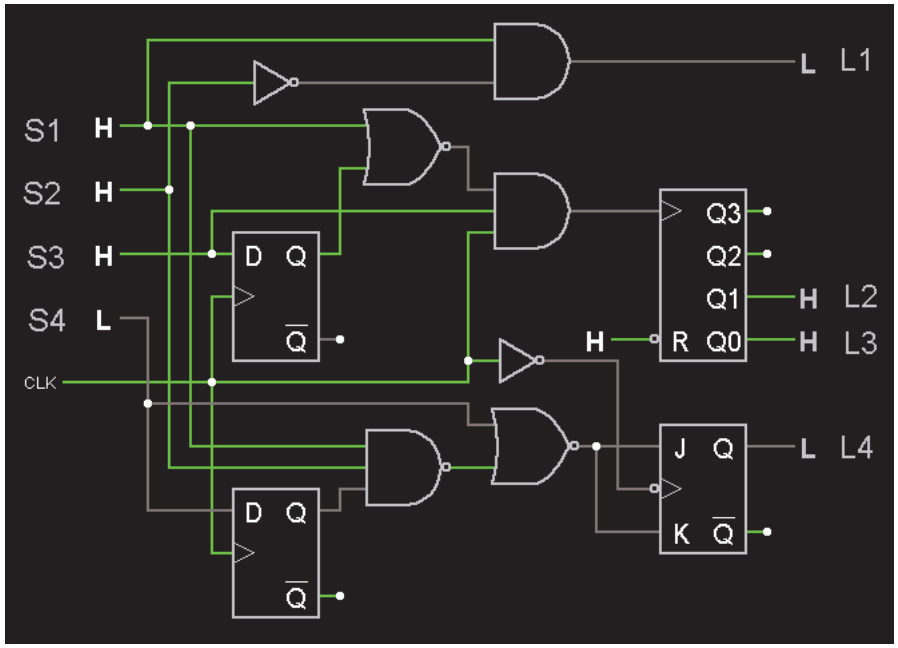
```
char leds=0,dip3p=0,dip4p=0,v=0;     // ROM – 548 / RAM – 81
while (1) {
if (DIP1&&!DIP2) leds|=0b1000; else leds&=0b0111;
  if (!DIP1&&((DIP3^dip3p)&&DIP3)) v++;
  if (v>3) v=0;
  leds=(leds&0b1001)|(v<<1);
  if (DIP1&&DIP2&&((DIP4^dip4p)&&!DIP4)) leds^=0b0001;
  dip3p=DIP3; dip4p=DIP4; led_out(leds);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

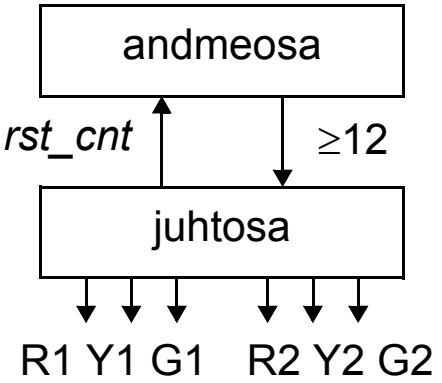
- Tegevused
  - kui S1=1 & S2=0, siis L1←1, muidu L1←0
  - kui S1=0 & S3↑, siis V++ (V[1]=L2, V[0]=L3)
  - kui S1=1 & S2=1 & S4↓, siis L4← ¬L4
- Kolm parallelset osa
- Kombinatsioon-skeem
  - L1 ← S1 & !S2
- Loendur + loogika
  - D-tiger – S3p ← S3
  - v++ ← !S1 & !S3p & S3
  - L2 ← V[1]; L3 ← V[0]
-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



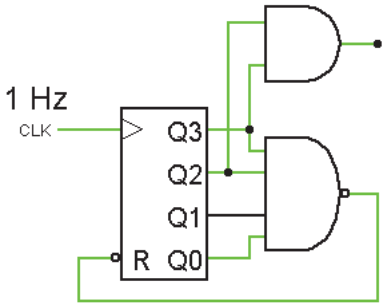
`http://mini.pld.ttu.ee/~lrv/IAS0150/switch4led4.txt`

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”

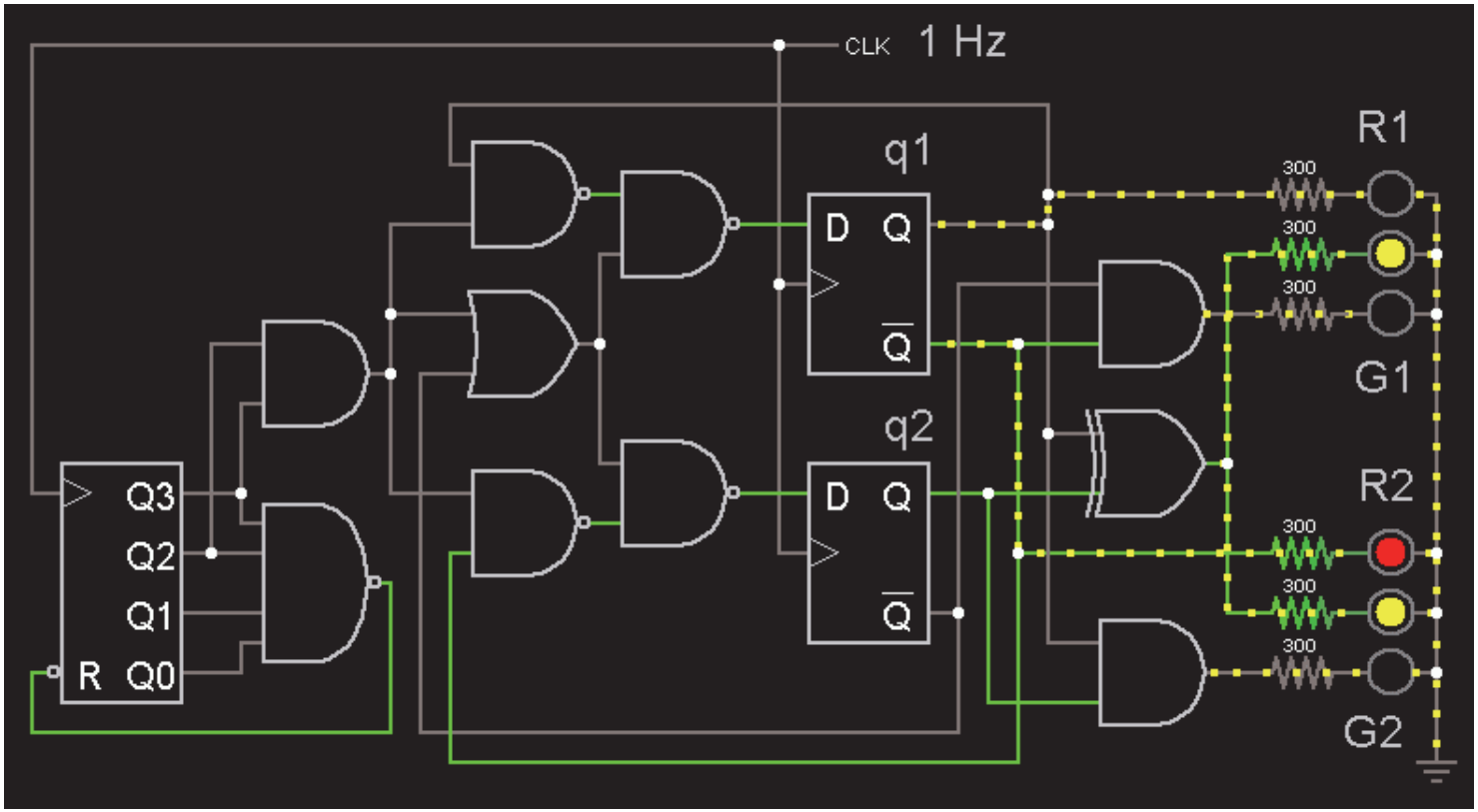


Andmeosa



`http://mini.pld.ttu.ee/~lrv/IAS0150/tlc-datapath.txt`

Valgusfoor – tulemus



http://mini.pld.ttu.ee/~lrv/IAS0150/tlc-applet.txt

Näide #3 – diferentsiaalvõrrand

- Kuidas jõuda algoritmist realisatioonini?

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

- 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**

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

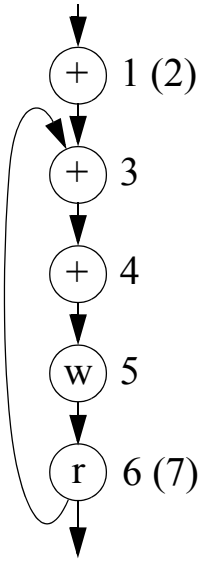
```
{
  sc_fixed<6,10> a,dx,y,x,u,x1,x2,y1;
  while ( true ) {
    wait(); a=inport.read();
    wait(); dx=inport.read();
    wait(); y=inport.read();
    wait(); x=inport.read();
    wait(); u=inport.read();
    while ( true ) {
      for (int i=0;i<7;i++) wait();
      x1 = x + dx;  y1 = y + (u*dx);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

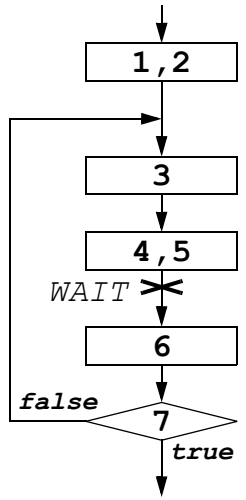
```
# R1:a, R2:dx, R3:y, R4:x, R5:u,
# R6:x1, R7:x2, R8:y1, R9:tmp
    ...
_loop_$32:
    ADD.fx   R6, R4, R2     # x1=x+dx
    MUL.fx   R9, R5, R2     # tmp=u*dx
    ADD.fx   R8, R3, R9     # y1=y+tmp
    MUL.fx   R9, R4, R9     # tmp=x*tmp
    MUL.fx   R9, R9, $5     # tmp=5*tmp
    SUB.fx   R5, R5,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

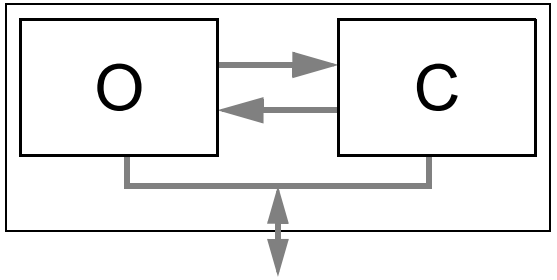
Näide #4 – otsimine mälust

```
...
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;
...
```

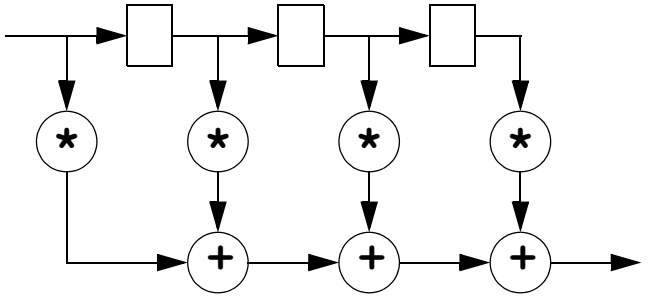


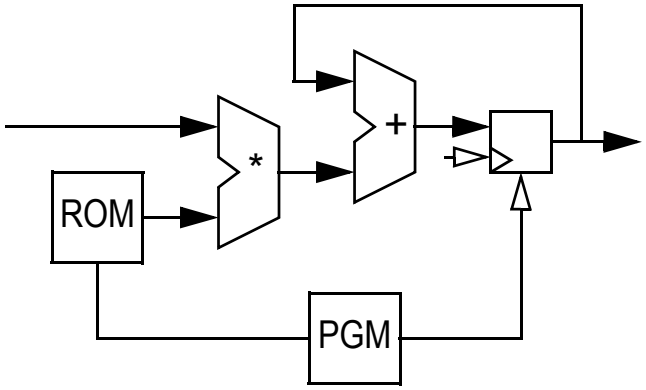


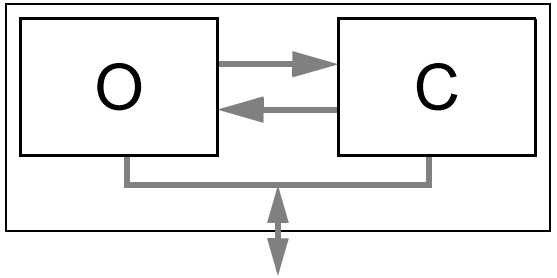


Näide #5 – FIR filter

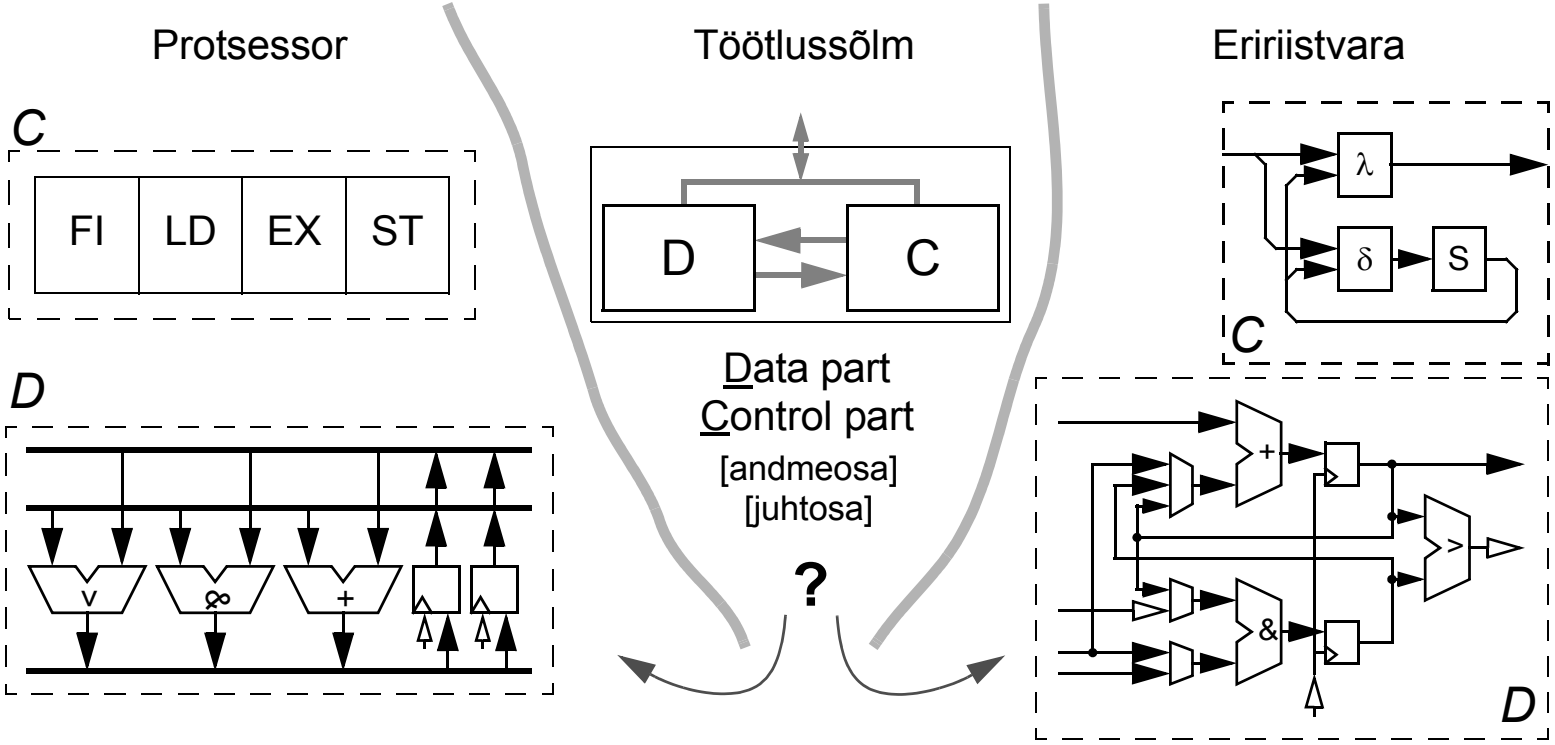
```
...
x=c0*i(0)+c1*i(1)+c2*i(2)+c3*i(3);
...
```







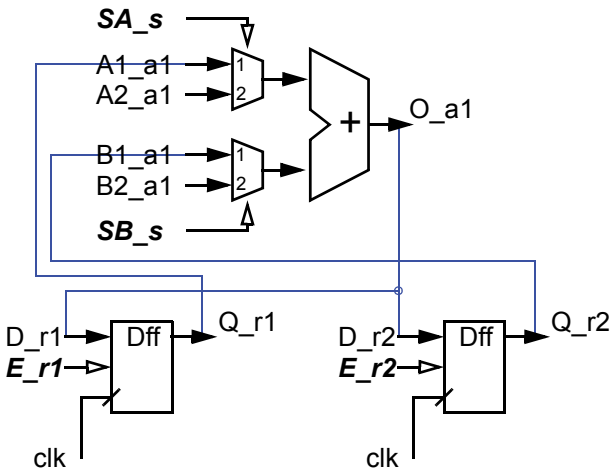
Tarkvara või Riistvara?



- **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 → ...



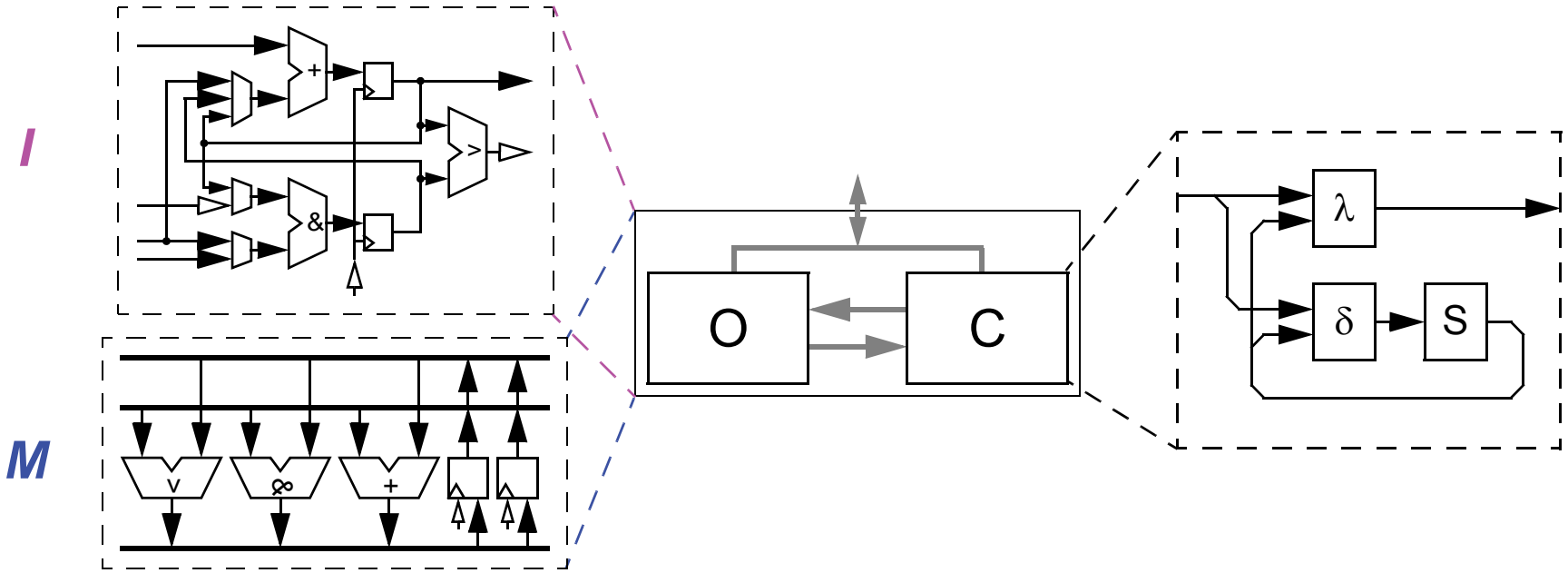
O_a1≡D_r1≡D_r2 Q_r1≡A1_a1
Q_r2≡B1_a1 n≡A2_a1 m≡B2_a1

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”

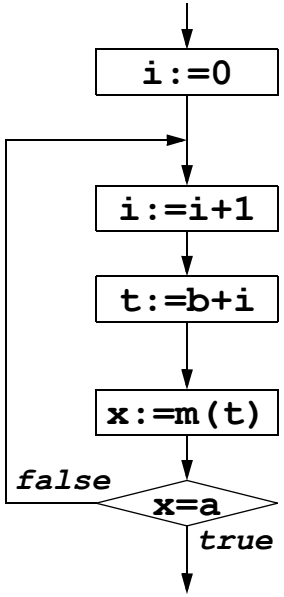
Andmeosa struktuur

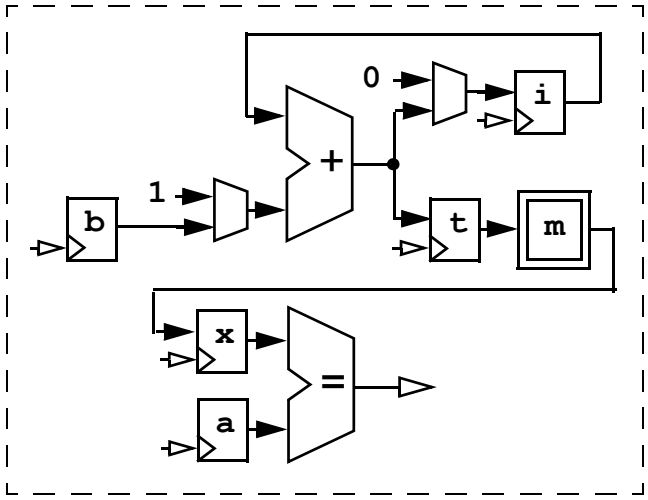
- I-automaat (~~eririistvara)
  - võimaldab üheaegselt sooritada kõiki funktsionaalselt ühitatavaid operatsioone (mikrokäske)
  - iga registri sisendis kombinatsioonskeem
- M-automaat (~~protsessor)
  - iga unikaalse operatsiooni jaoks on oma täitursõlm, üheaegselt võimalik täita tüübilt erinevaid operatsioone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



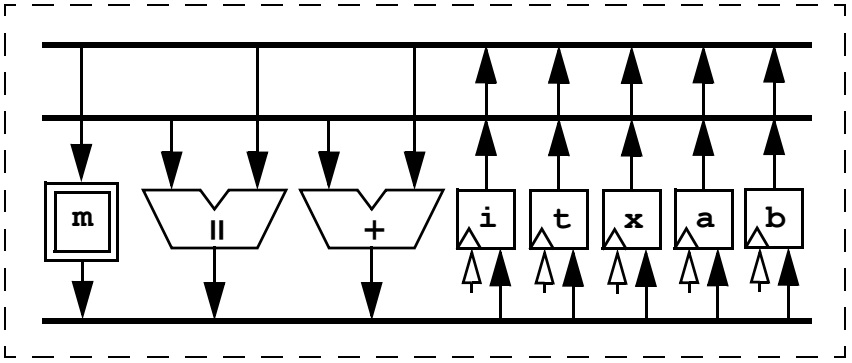
Andmeosa – eridisain või üldotstarbeline?

[Otsimine mälust]





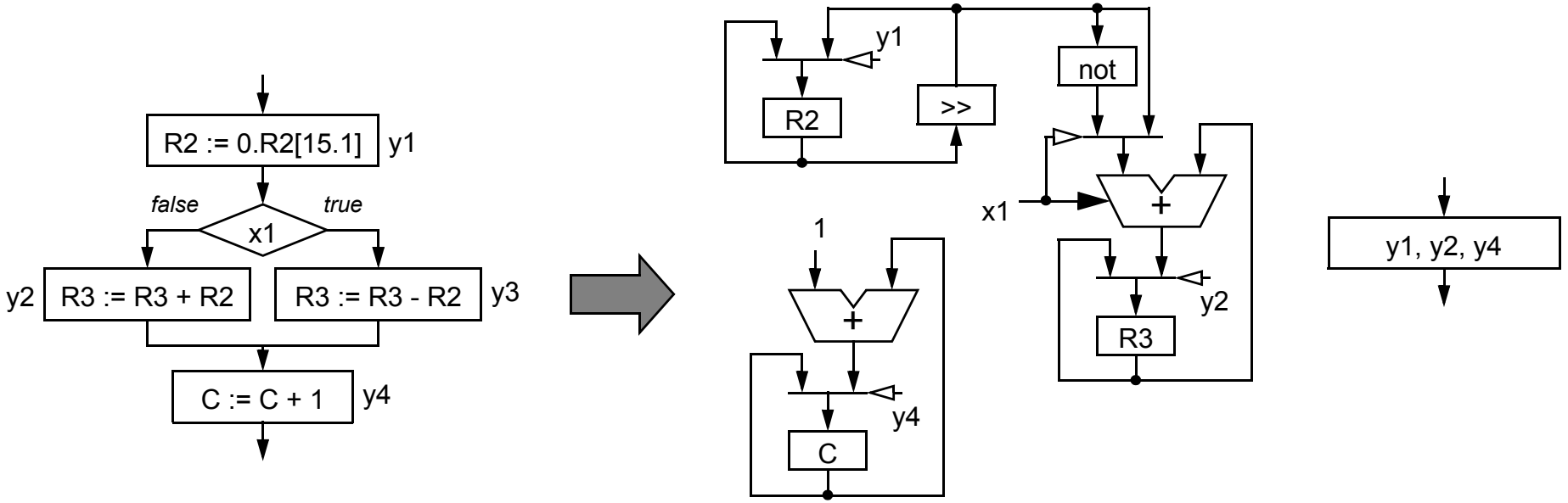
eridisain (e. *I*)
+ kiire
+ väike
-- paindumatu



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

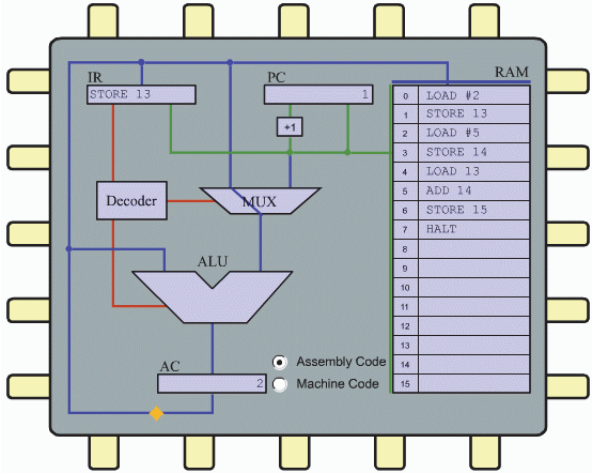
Realisatsiooni optimaalsus?

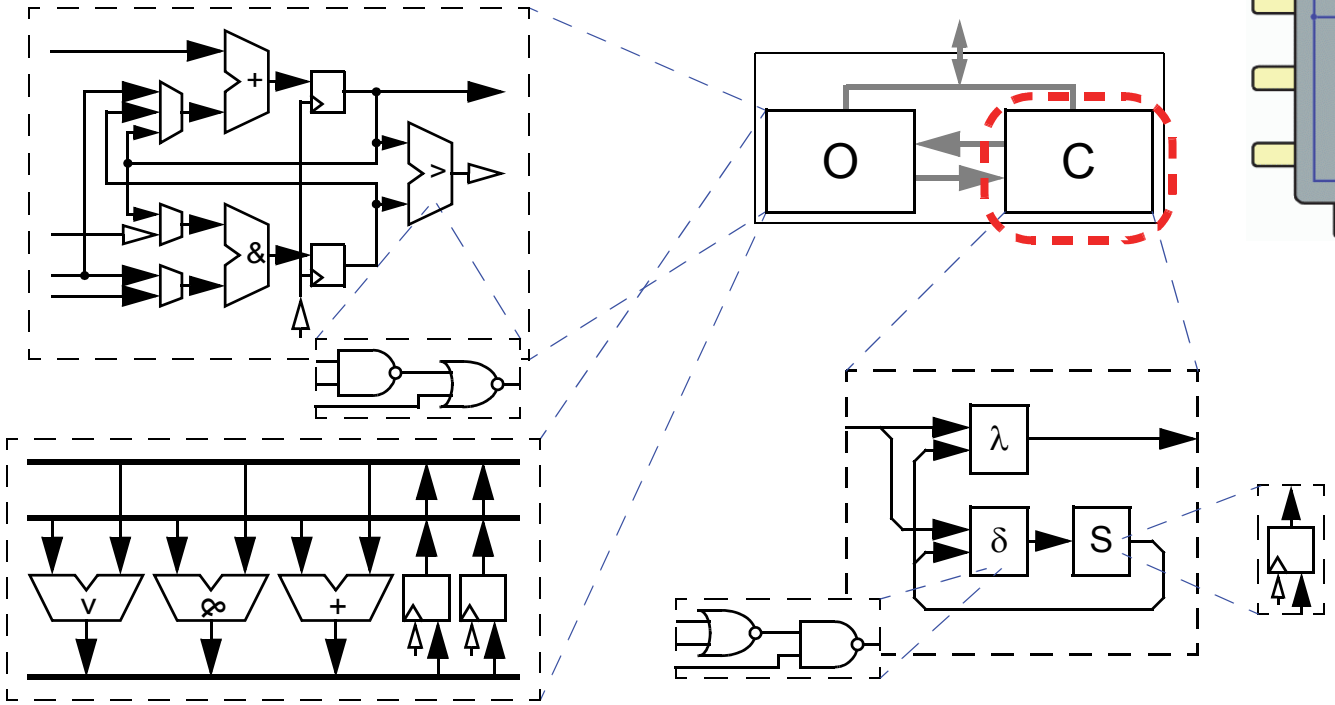
- **operatsiooni hind riistvaras**
  - **nihutamine == traatide teisiti viimine**
  - **liitmist ja lahutamist teostab sama seade – summaator (+ülekanne)**
- **sõltumatud operatsioonid võib täita korraga**
  - **tegelike andmesõltuvuste analüüs – nt. R2 nihutamise tulemus**



Juhtautomaat

- **Digitaalsüsteem = andmeosa + juhtosa**

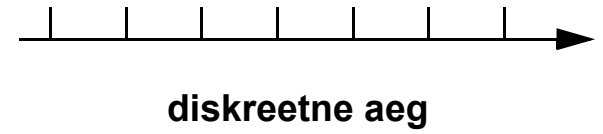


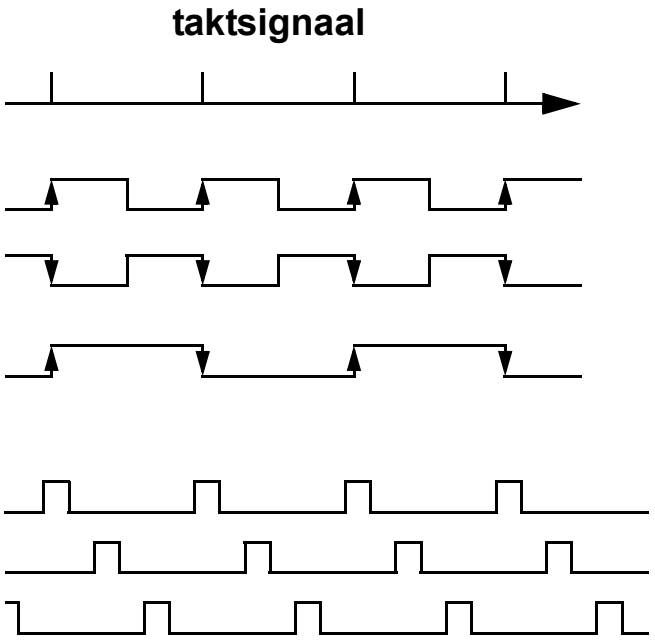


`http://courses.cs.vt.edu/~csonline/MachineArchitecture/Lessons/index.html`

Diskreetne aeg

- Rangelt järjestatud sündmuste jada (ajamomentide jada)
  - üksiksündmuse kestus on 0
  - sündmuste vaheline ajavahemik pole oluline
- Taktsignaal (clock)
  - reaalne diskreetse aja esitusviis
  - üksiksündmus == taktsignaali front
    - tõusev või langev front – ühefaasiline taktsignaal (single phased clock)
    - tõusev ja langev front – kahefaasiline taktsignaal (double phased clock)
  - varasem mitmefaasiline taktsignaal – eri faasid tüürisid eri mäluelemente





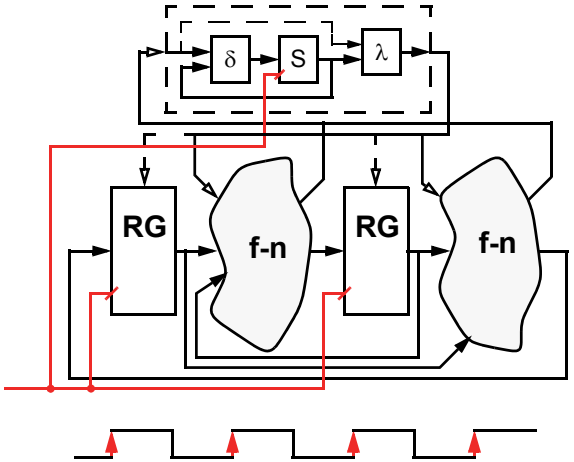
Takteerimine

- register → kombinatsioonskeem → register → ...
- registri tüüp määrab taktsignaali faaside arvu ja lubatud tagasisided

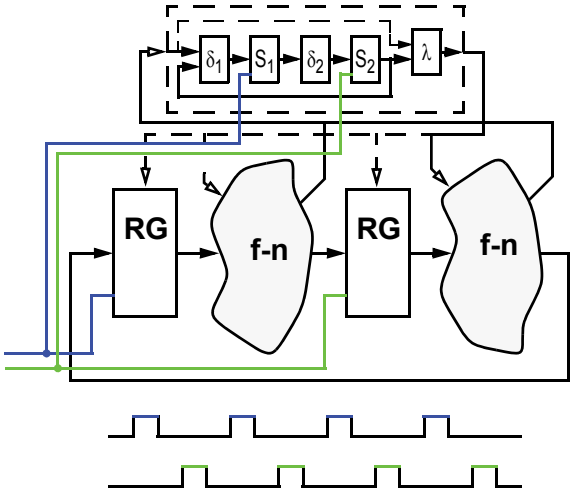
- flip-flop
  - frondist aktiveeruv triger
  - tagaside korral peab olema vähemalt üks triger ahelas

- latch
  - lukustuv triger (lukk-register)
  - taktsignaali aktiivse nivoo korral on triger “läbipaistev” –> tagasiside selliselt, et vähemalt üks suletud triger ahelas

ühefaasiline taktsignaal – flip-flop registrid



kahefaasiline taktsignaal(id) – lukk-registrid



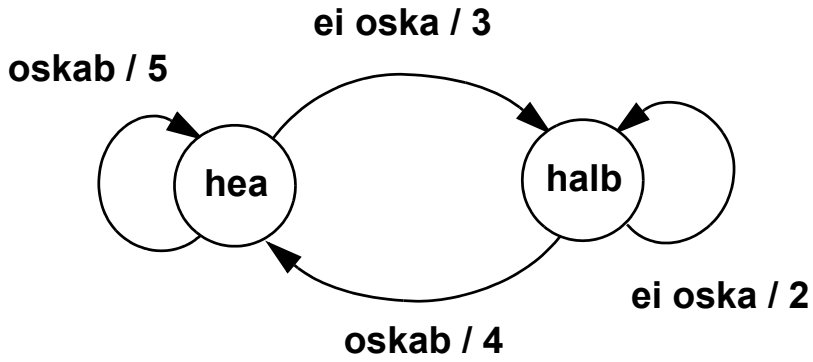
Abstraktne automaat

- Kombinatoorne skeem
  - skeemi väljund sõltub ainult skeemi sisendist
  - atsüklilise topoloogiaga skeem on kombinatoorne
  - tsükliga (tagasisidega) skeemid võivad olla kombinatoorsed
- Mäluga skeem
  - eksisteerivad mäluelemendid
  - asünkroonsetes skeemis võib mäluelement olla mitteilmutatud kujul
  - tagasiside on vajalik
- Automaat – mäluga skeemi erijuht
  - sisendid, väljundid ja olekud – lõplikud hulgad
  - abstraktne automaat, lõplik automaat
  - automaton (pl. automata), sequential machine, finite state machine (FSM)

Näide

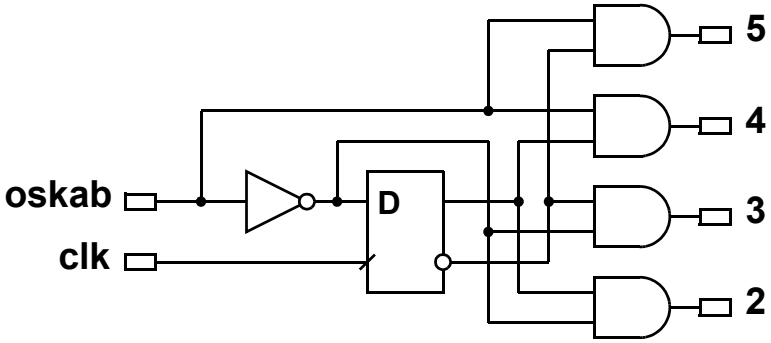
- Õppejõu käitumine eksamil
  - kui õppejõud on heas tujus ja tudeng oskab, siis tudeng saab 5 ning õppejõu hea tuju säilub
  - kui õppejõud on heas tujus ja tudeng ei oska, siis tudeng saab 3 ning õppejõu tuju läheb halvaks
  - kui õppejõud on halvas tujus ja tudeng ei oska, siis tudeng saab 2 ning õppejõu halb tuju säilub
  - kui õppejõud on halvas tujus ja tudeng oskab, siis tudeng saab 4 ning õppejõu tuju läheb heaks

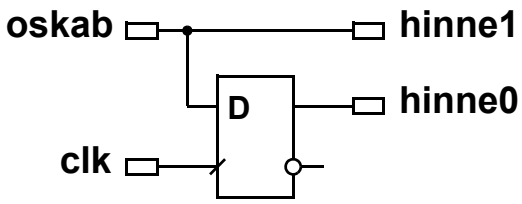
sisend	*olek*	*väljund*	*uus olek*
tudeng	õppejõud	hinne	õppejõud
oskab	hea tuju	5	hea tuju
ei oska	hea tuju	3	halb tuju
ei oska	halb tuju	2	halb tuju
oskab	halb tuju	4	hea tuju



Näide – õppejõu realisatsioon riistvaras

- **Realisatsiooni efektiivsus sõltub kodeeringust!**
  - **Sisend- ja väljundkodeering üldjul teada**
  - **Olekute kodeerimine oluline**
    - **pindala – loogikalülide arv**
    - **viide – funktsioonide keerukus**
    - **võimsustarve – lülituste arv ajaühikus**
  - **Realisatsioon #1**
    sisend: ei oska - 0, oskab - 1
    väljund: 2 - 0001, 3 - 0010, 4 - 0100, 5 - 1000
    olek: hea - 0, paha - 1
  - **Realisatsioon #2**
    sisend: ei oska - 0, oskab - 1
    väljund: 2 - 00, 3 - 01, 4 - 10, 5 - 11
    olek: hea - 1, paha - 0





Abstraktne automaat – definitsioonid

- **Automaat on viisik (quintuple) – M = (S, I, O, δ, λ)**
  - **S: (sise)olekute hulk (states)**
  - **I: sisendite hulk (inputs)**
  - **O: väljundite hulk (outputs)**
  - **δ: siirdefunktsioon (transition) - δ: S $\times$ I $\rightarrow$ S**
  - **λ: väljundfunktsioon - λ: S $\times$ I $\rightarrow$ O**
- **Hulgad on lõplikud ja (üldjuhul) mittetühjad**
  - **hulkade ja funktsioonide erijuhud – automaatide erijuhud**
- **Lähteolek s_0 – M = (S, I, O, δ, λ, s_0)**

Automaatide erijuhud

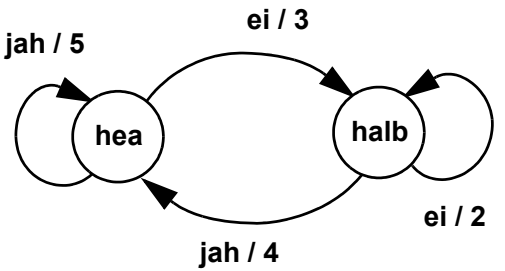
- **Mealy automaat – $M = (S, I, O, \delta, \lambda)$**
  - $S \neq \varnothing$, $I \neq \varnothing$, $O \neq \varnothing$, $\delta: S \times I \rightarrow S$, $\lambda: S \times I \rightarrow O$
- **Moore automaat – $M = (S, I, O, \delta, \lambda)$**
  - $S \neq \varnothing$, $I \neq \varnothing$, $O \neq \varnothing$, $\delta: S \times I \rightarrow S$, $\lambda: S \rightarrow O$
- **Primitiivne automaat – $M = (S, I, \delta)$**
  - $S \neq \varnothing$, $I \neq \varnothing$, $O = \varnothing$ $(O \equiv S)$, $\delta: S \times I \rightarrow S$, $\lambda \equiv \varnothing$
- **Generaator – $M = (S, O, \delta, \lambda)$**
  - $S \neq \varnothing$, $I = \varnothing$, $O \neq \varnothing$, $\delta: S \rightarrow S$, $\lambda: S \rightarrow O$
- **Loogikafunktsioon – $M = (I, O, \lambda)$**
  - $S = \varnothing$, $I \neq \varnothing$, $O \neq \varnothing$, $\delta = \varnothing$, $\lambda: I \rightarrow O$
- **Mikroprogramm automaat – $M = (S, I, O, \delta, \lambda)$**
  - $S \neq \varnothing$, $I = \{0,1\}^L$, $O = \{0,1\}^M$, $\delta: S \times I \rightarrow S$, $\lambda: S \times I \rightarrow O$

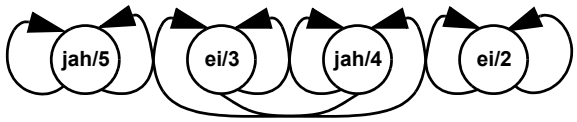
Esitusviisid

- **Tabel**
  - **veerud:**
    sisend (i_t), jooksev olek (s_t),
    väljund (o_t), uus olek (s_{t+1})
  - **read: siire jooksvast olekust uude olekusse:**
    $i_t \times s_t \rightarrow o_t \times s_{t+1}$

i_t	s_t	o_t	s_{t+1}
jah	hea	5	hea
ei	hea	3	halb
ei	halb	2	halb
jah	halb	4	hea

- **Olekudiagramm, olekugraaf (state graph)**
  - **sõlmed: olekud**
  - **kaared: siirded**
- **Siirdediagramm (transition graph)**
  - **sõlmed: siirded**
  - **kaared: olekud**



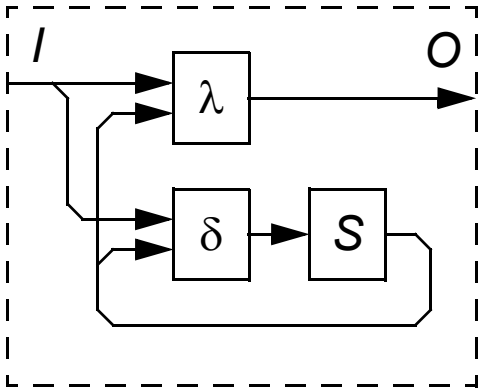


Automaatide omadusi

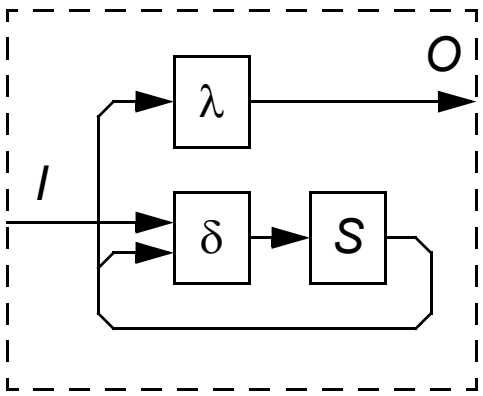
- **Osaliselt määratud automaadid**
  - **leidub olekuid, kus siire pole mingi sisendi puhul määratud**
    - **lihtsustatud kirjapilt – vaikimisi jääb nt. samasse olekusse**
    - **määramatus tuleneb väliskeskkonna iseärasustes – mitte-eksisteerivad sisendkombinatsioonid**
    - **kahendkodeeritud olekud – osa kombinatsioone on kasutamata**
    - **automaadi minimeerimisel vabamad käed – osaliselt määratud loogikafunktsioonid**
- **Mittedeterministlikud automaadid**
  - **leidub olekute ja sisendite kombinatsioone, mille puhul on määratud rohkem kui üks järgmine olek**
    - **kompaktne meetod kirjeldamiseks, kui leidub rohkem kui üks legaalne reaktsioon mingile sisendkombinatsioonile (jadale)**
    - **matemaatilised mudelid**
- **Isomorfism**
  - **üksühene vastavus kahe automaadi komponentide (S,I,O,δ,λ) vahel**
- **Homomorfism**
  - **ühene vastavus kahe automaadi komponentide (S,I,O,δ,λ) vahel ~ “alam-automaat”**

Automaadi struktuur

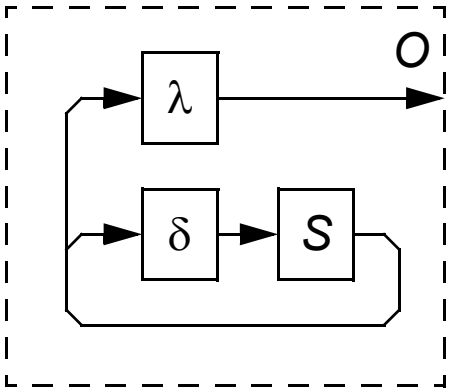
Mealy



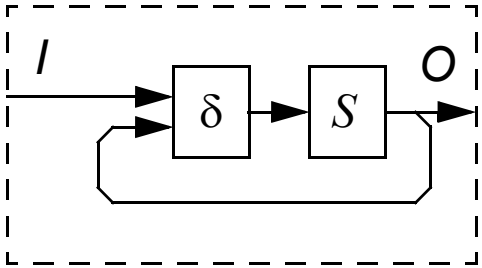
Moore



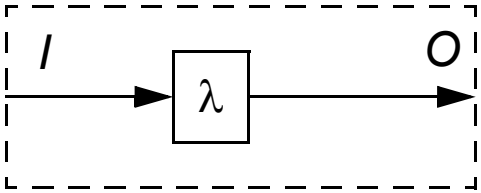
generaator



primitiivne



loogikafunktsioon



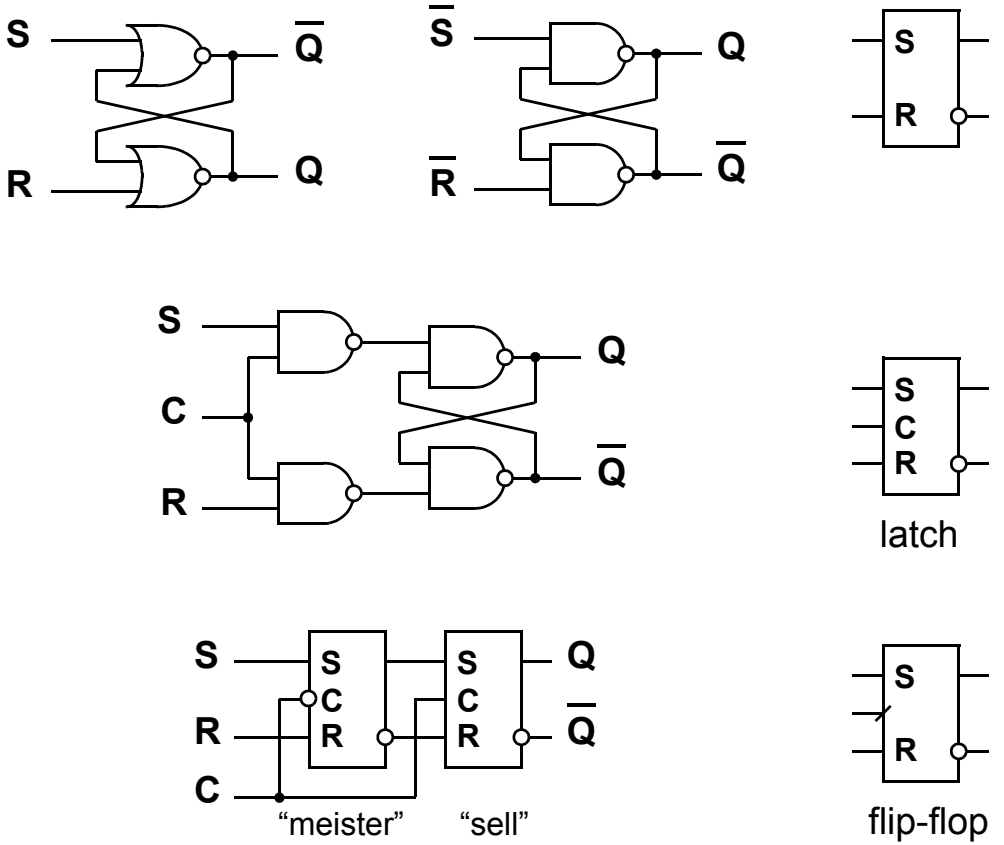
Mäluelemendid

- Salvestavad olekukoodi
- Register
  - kahendvektori salvestamiseks
  - sama tüüpi mäluelemendid
- Mäluelementide tüübid
  - funktsionaalsus – SR, JK, D- ja T-trigerid
  - takteerimine
    - asünkroone – takt puudub
    - latch (lukk-register/-triger) – läbipaistev kui takt on aktiivne
    - flip-flop (frondist aktiveeruv triger) – väljundis muutus ainult taktsignaali frondi korral
      master-slave (meister-sell) – kaks järjestikust lukk-registrit (latch'i)
      frondile reageerivad trigerid – spetsiaalne sise-ehitus

SR-triger (set/reset)

S	R	Q_{t+1}	$\overline{Q}_{t+1}$
0	0	Q_t	$\overline{Q}_t$
0	1	0	1
1	0	1	0
1	1	?	?

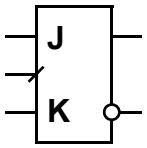
Q_t	Q_{t+1}	S	R
0	0	0	-
0	1	1	0
1	0	0	1
1	1	-	0



JK-triger

J	K	Q_{t+1}	$\overline{Q}_{t+1}$
0	0	Q_t	$\overline{Q}_t$
0	1	0	1
1	0	1	0
1	1	$\overline{Q}_t$	Q_t

Q_t	Q_{t+1}	J	K
0	0	0	-
0	1	1	-
1	0	-	1
1	1	-	0



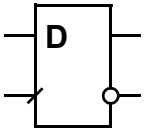
- Määramatus võimaldab loogikafunktsioone efektiivsemalt minimeerida
- Kaks sisendit $\rightarrow$ kaks loogikafunktsiooni

D-triger (delay)

- Kõige enam kasutusel
  - lihtne sise-ehitus
  - väike sisendite arv → vähem loogikafunktsioone

D	Q_{t+1}	$\overline{Q}_{t+1}$
0	0	1
1	1	0

Q_t	Q_{t+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

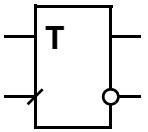


T-triger

- Sobiv loendurites kasutamiseks

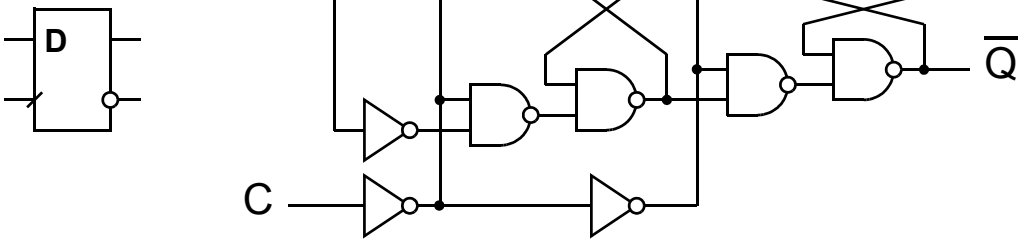
T	Q_{t+1}	$\overline{Q}_{t+1}$
0	Q_t	$\overline{Q}_t$
1	$\overline{Q}_t$	Q_t

Q_t	Q_{t+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

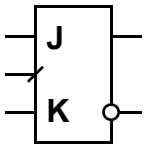


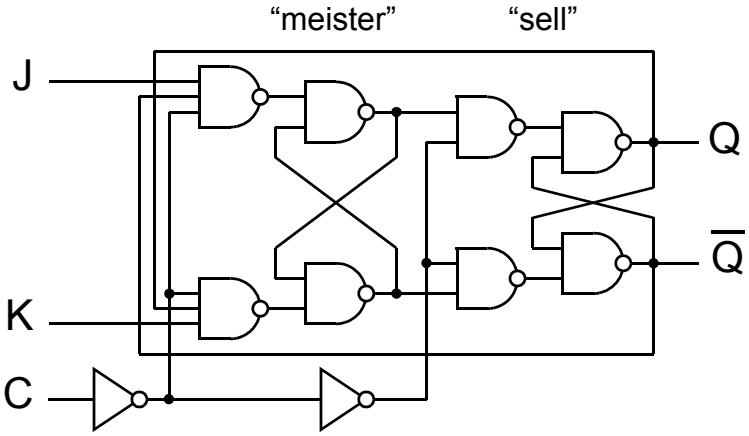
Trigerite sise-ehitus

D-triger

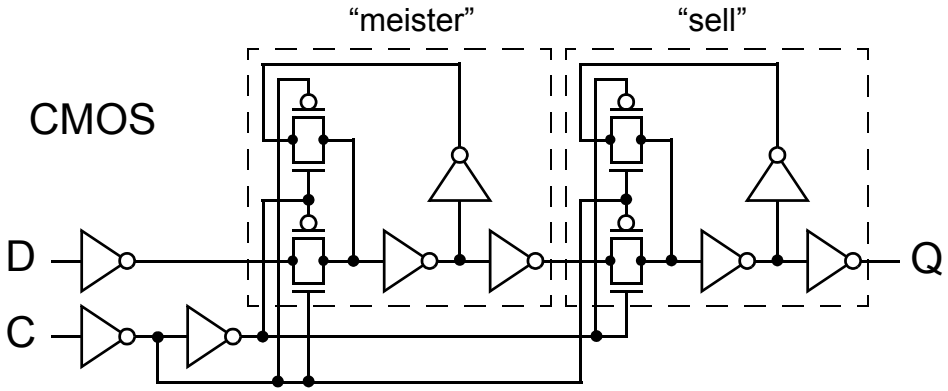


JK-triger





CMOS



- **Trigerite simulaatorid**
  - **http://www.falstad.com/circuit/ – Sequential Logig –> Flip-Flops –> ...**
  - **http://tams-www.informatik.uni-hamburg.de/applets/cmos/cmosdemo.html – CMOS D-latch**

Trigerite ajalised parameetrid

- Sisemised ahelad erinevate viidetega
  - nt. sisendist mäluelementideni võrreldes tagasisidega
  - seadeaeg (setup) – nõutav valmisoleku aeg
  - hoideaeg (hold) – nõutav stabiilsuse aeg
  - nõuete rikkumise korral metastabiilsuse oht

Juhtautomaatide süntees

- Automaadi olekudiagrammi / tabeli genereerimine (süntees)
  - tabeli süntees plokk-skeemist (plokk-diagrammist)
  - plokk-skeemi genereerimine kõrgtaseme keeltest
- Automaadi süntees olekudiagrammist / tabelist
  - eesmärk – automaadi efektiivne realisatsioon
  - sisendite / väljundite kodeerimine; olekute kodeerimine
  - siirde- ja väljundfunktsiooni süntees ja minimeerimine