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, ...**

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)

Disainināide

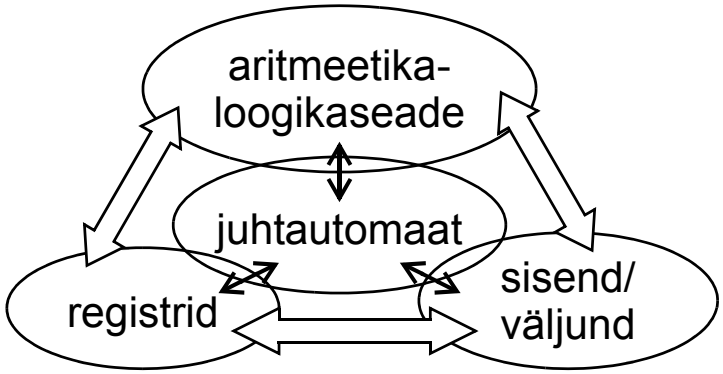
- Neli kahend-sisendit ja -väljundit – nt. 4 lülitit (S1-S4) ja 4 valgusdioodi (L1-L4)
- Sisendite muutumine muudab väljundeid
  - kui S1=1 & S2=0, siis L1←1, muidu L1←0
  - kui S1=0 & S3↑, siis V++ (V[1]=L2, V[0]=L3)
  - kui S1=1 & S2=1 & S4↓, siis L4← ¬L4

- 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**

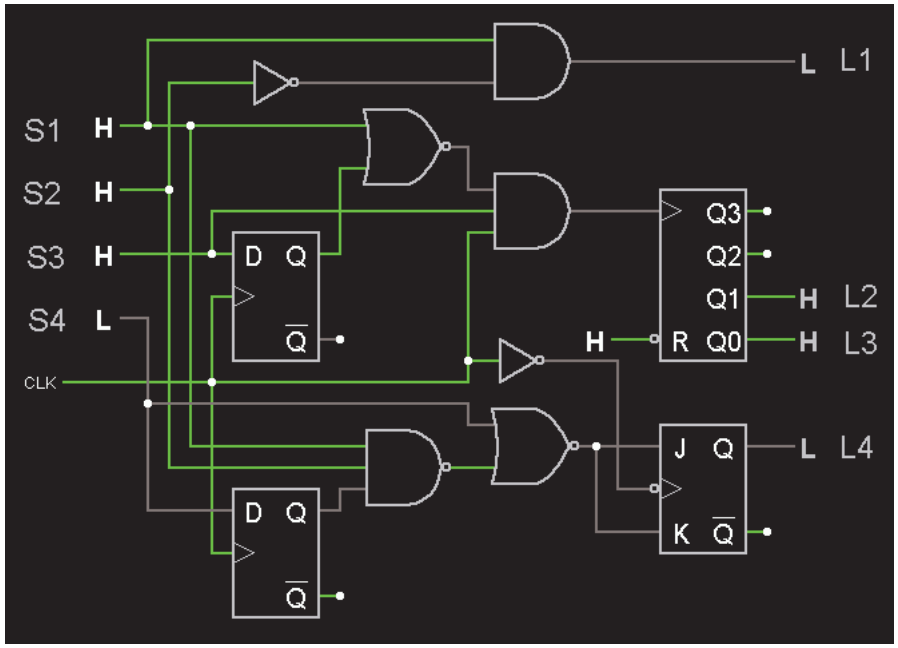
```
char leds=0,dip3p=0,dip4p=0,v=0;     // ROM – 548 / RAM – 81
while (1) {
if (DIP1&&!DIP2) leds|=0b1000; else leds&=0b0111;
  if (!DIP1&&((DIP3^dip3p)&&DIP3)) v++;
  if (v>3) v=0;
  leds=(leds&0b1001)|(v<<1);
  if (DIP1&&DIP2&&((DIP4^dip4p)&&!DIP4)) leds^=0b0001;
  dip3p=DIP3; dip4p=DIP4; led_out(leds);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

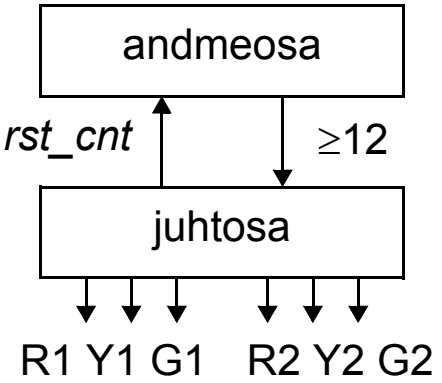
- Tegevused
  - kui S1=1 & S2=0, siis L1←1, muidu L1←0
  - kui S1=0 & S3↑, siis V++ (V[1]=L2, V[0]=L3)
  - kui S1=1 & S2=1 & S4↓, siis L4← ¬L4
- Kolm parallelset osa
- Kombinatsioon-skeem
  - L1 ← S1 & !S2
- Loendur + loogika
  - D-tiger – S3p ← S3
  - v++ ← !S1 & !S3p & S3
  - L2 ← V[1]; L3 ← V[0]
-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



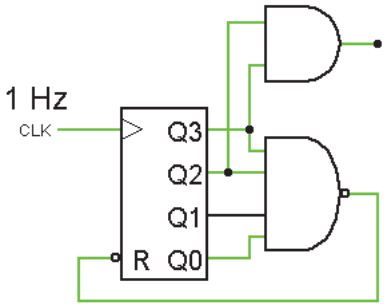
`http://mini.pld.ttu.ee/~lrv/IAY0150/switch4led4.txt`

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”

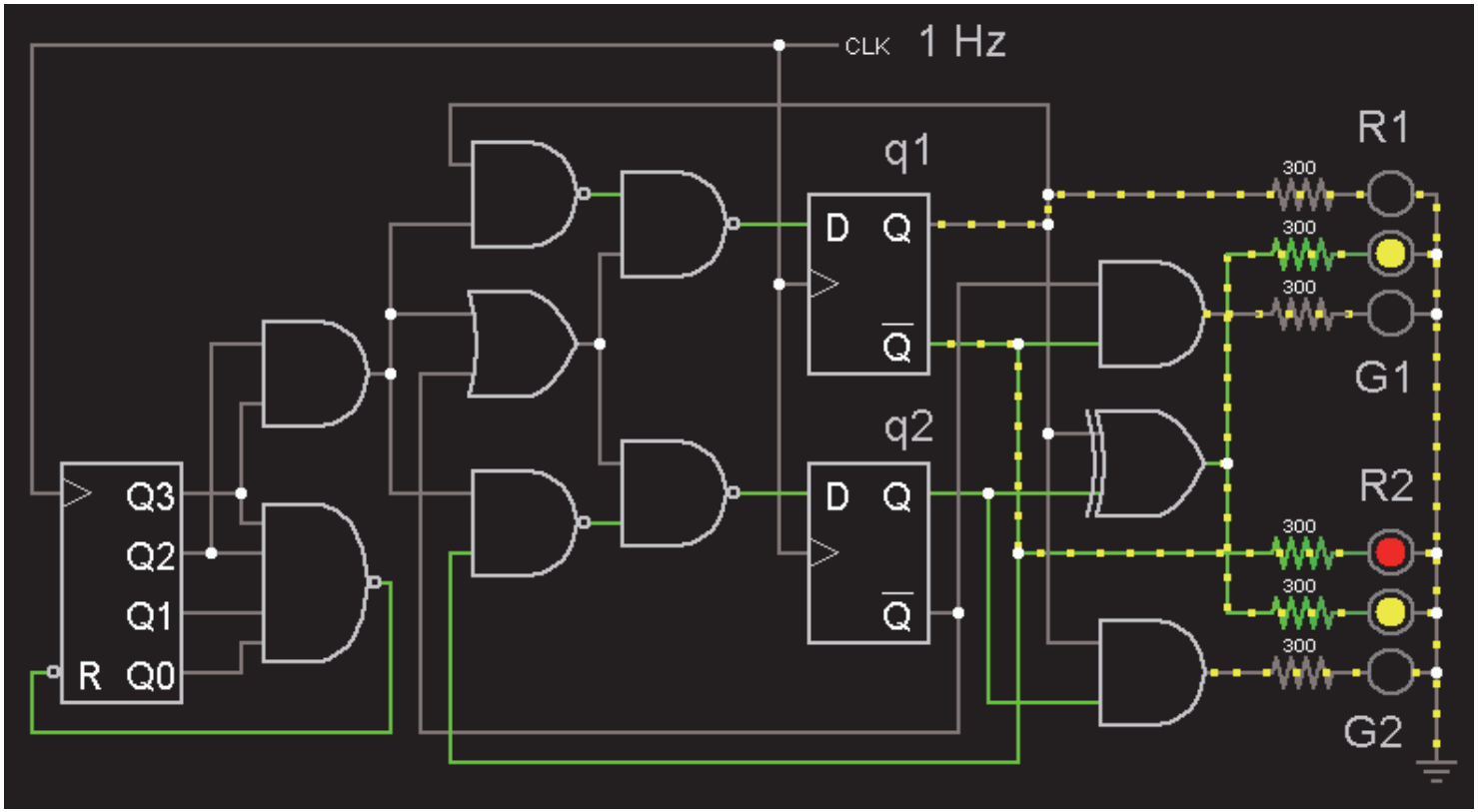


Andmeosa



`http://mini.li.ttu.ee/~lrv/IAY0150/tlc-datapath.txt`

Valgusfoor – tulemus



http://mini.li.ttu.ee/~lrv/IAY0150/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**

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

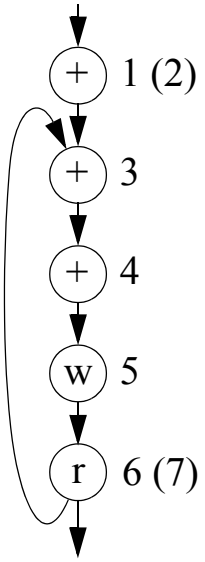
```
{
  sc_fixed<6,10> a,dx,y,x,u,x1,x2,y1;
  while ( true ) {
    wait(); a=inport.read();
    wait(); dx=inport.read();
    wait(); y=inport.read();
    wait(); x=inport.read();
    wait(); u=inport.read();
    while ( true ) {
      for (int i=0;i<7;i++) wait();
      x1 = x + dx;  y1 = y + (u*dx);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

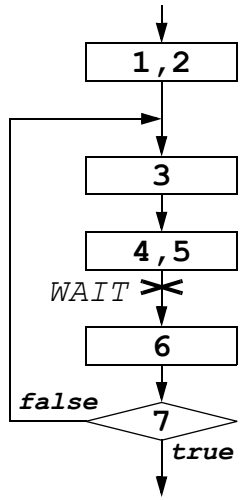
```
# R1:a, R2:dx, R3:y, R4:x, R5:u,
# R6:x1, R7:x2, R8:y1, R9:tmp
    ...
_loop_$32:
    ADD.fx   R6, R4, R2     # x1=x+dx
    MUL.fx   R9, R5, R2     # tmp=u*dx
    ADD.fx   R8, R3, R9     # y1=y+tmp
    MUL.fx   R9, R4, R9     # tmp=x*tmp
    MUL.fx   R9, R9, $5     # tmp=5*tmp
    SUB.fx   R5, R5,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

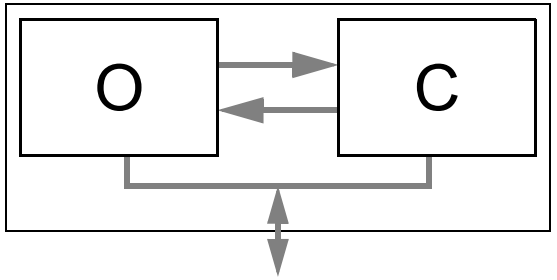
Näide #4 – otsimine mälust

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

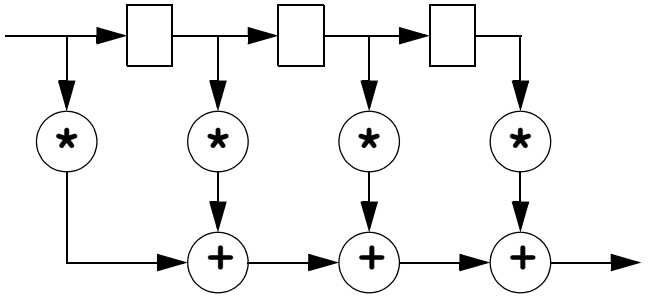


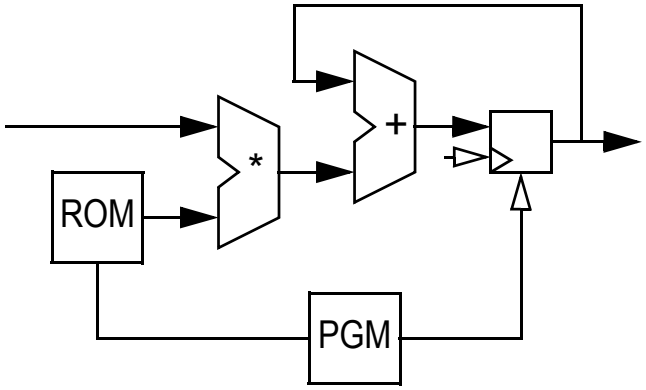


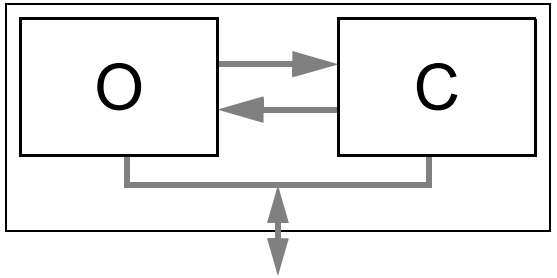


Näide #5 – FIR filter

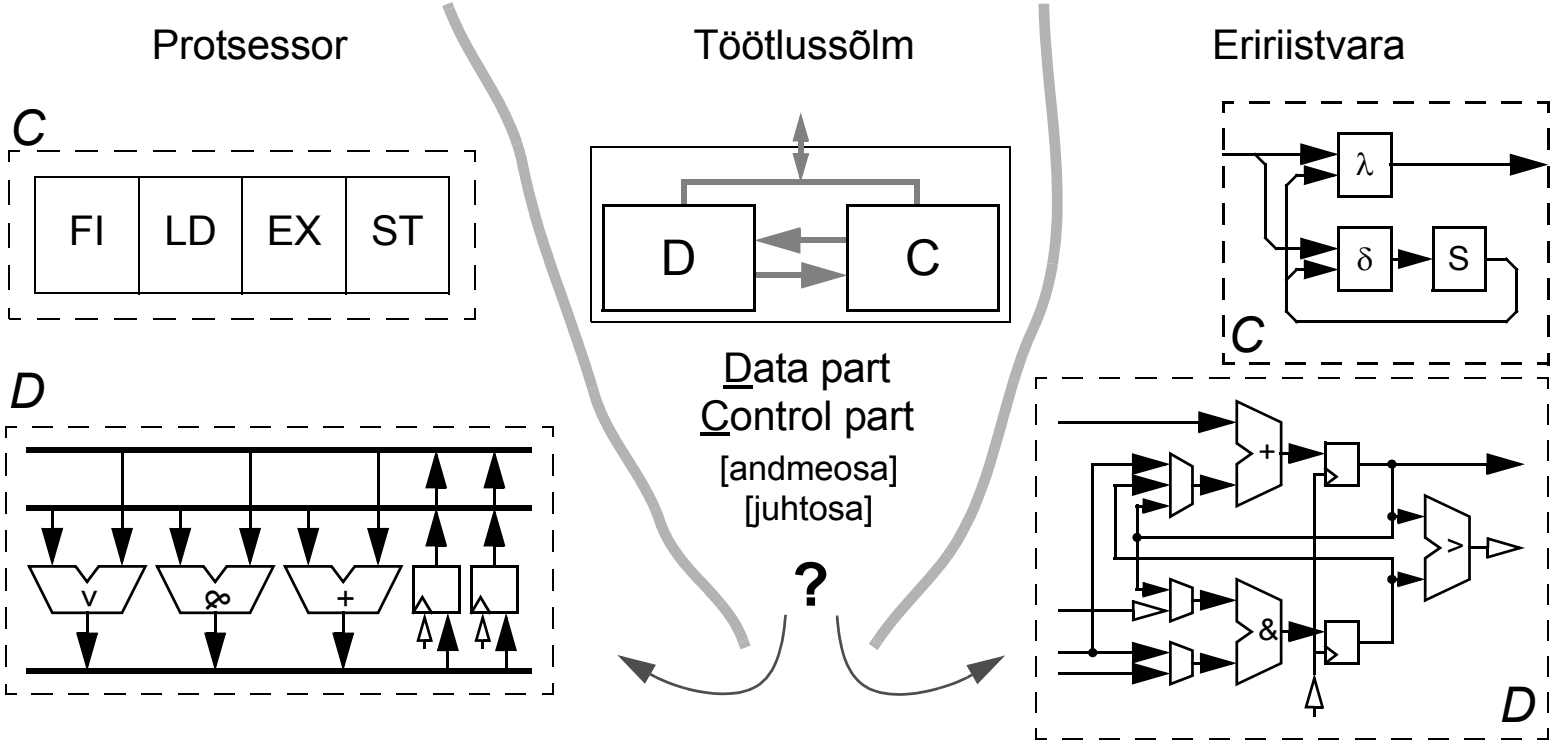
```
...
x=c0*i(0)+c1*i(1)+c2*i(2)+c3*i(3);
...
```







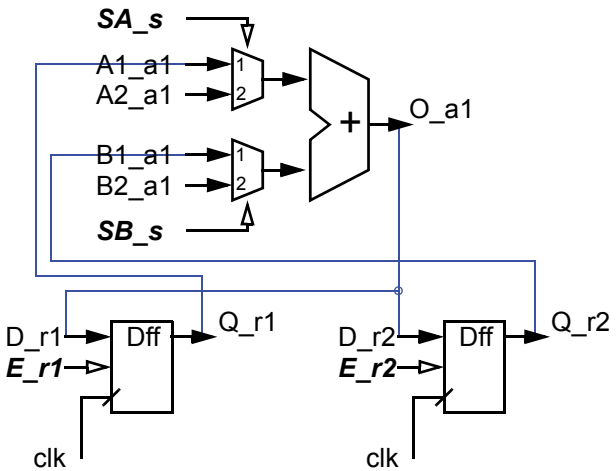
Tarkvara või Riistvara?



- **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

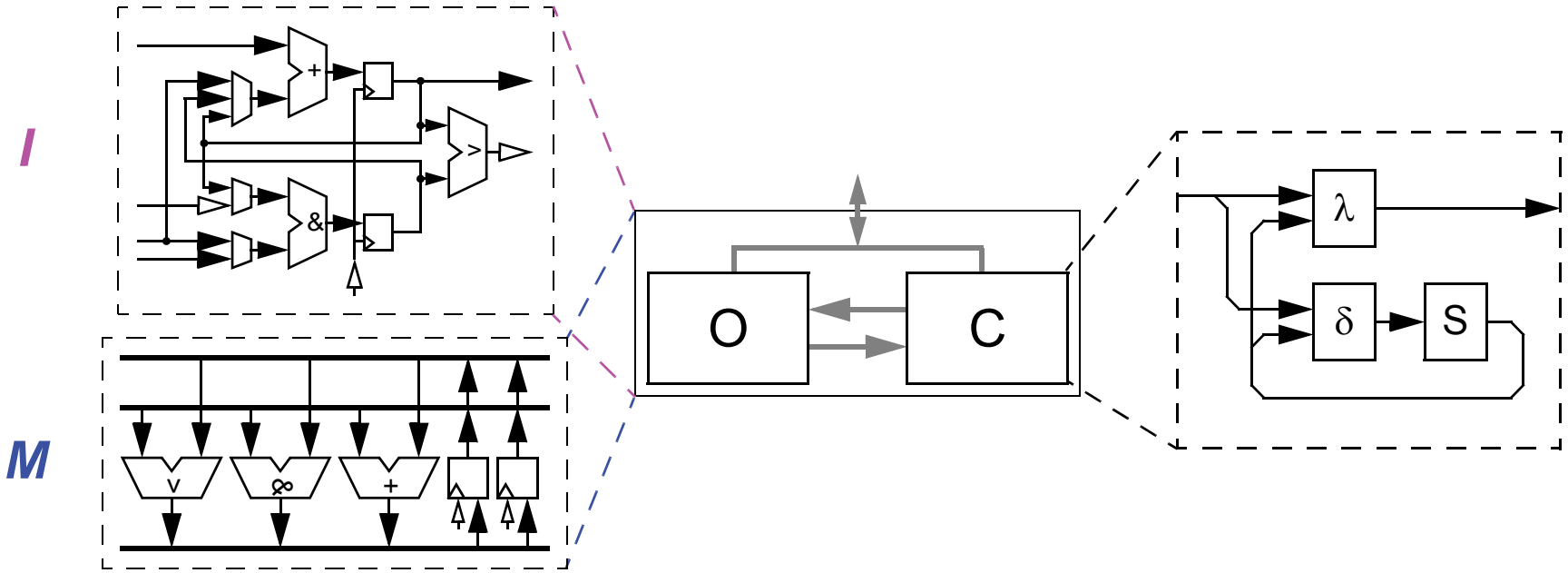
x = 2 * n + 2 * m	SA_s	SB_s	E_r1	E_r2
1: r2 <- n	2	0	0	1
2: r2 <- n + r2	2	1	0	1
3: r1 <- m	0	2	1	0
4: r1<- r1 + m	1	2	1	0
5: r1 <- r1 + r2	1	1	1	0

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

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

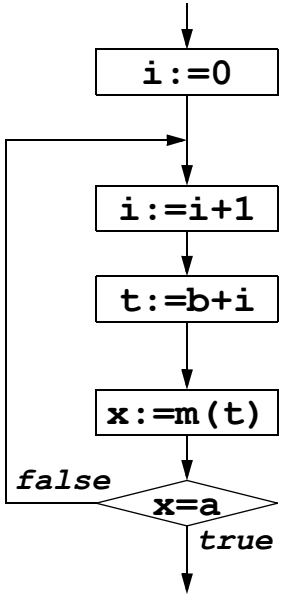
Andmeosa struktuur

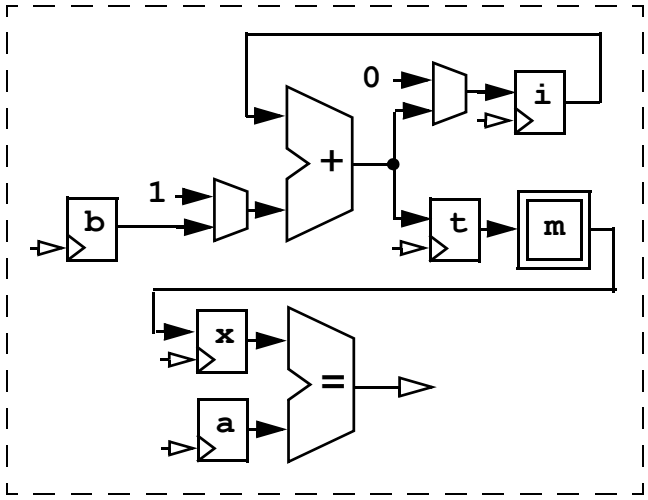
- I-automaat (~~eririistvara)
  - võimaldab üheaegselt sooritada kõiki funktsionaalselt ühitatavaid operatsioone (mikrokäske)
  - iga registri sisendis kombinatsioonskeem
- M-automaat (~~protsessor)
  - iga unikaalse operatsiooni jaoks on oma täitursõlm, üheaegselt võimalik täita tüübilt erinevaid operatsioone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



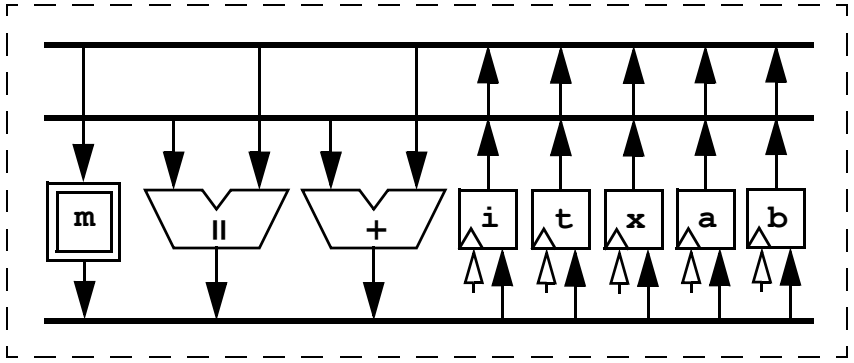
Andmeosa – eridisain või üldotstarbeline?

[Otsimine mälust]





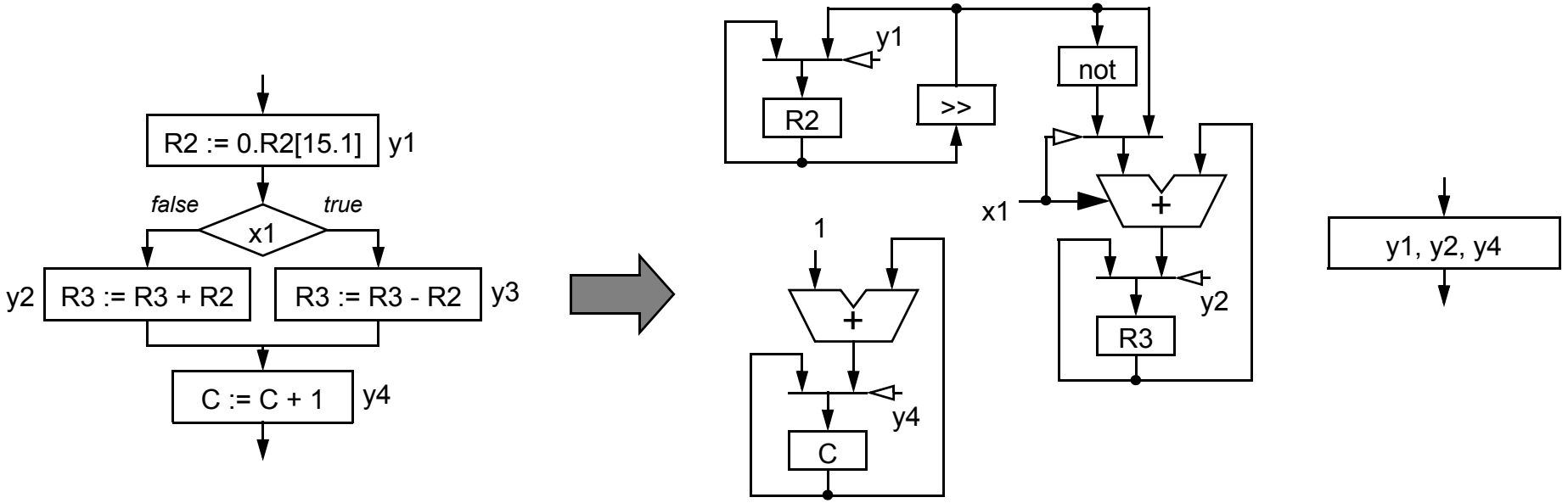
eridisain (e. *I*)
+ kiire
+ väike
-- paindumatu



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

Realisatsiooni optimaalsus?

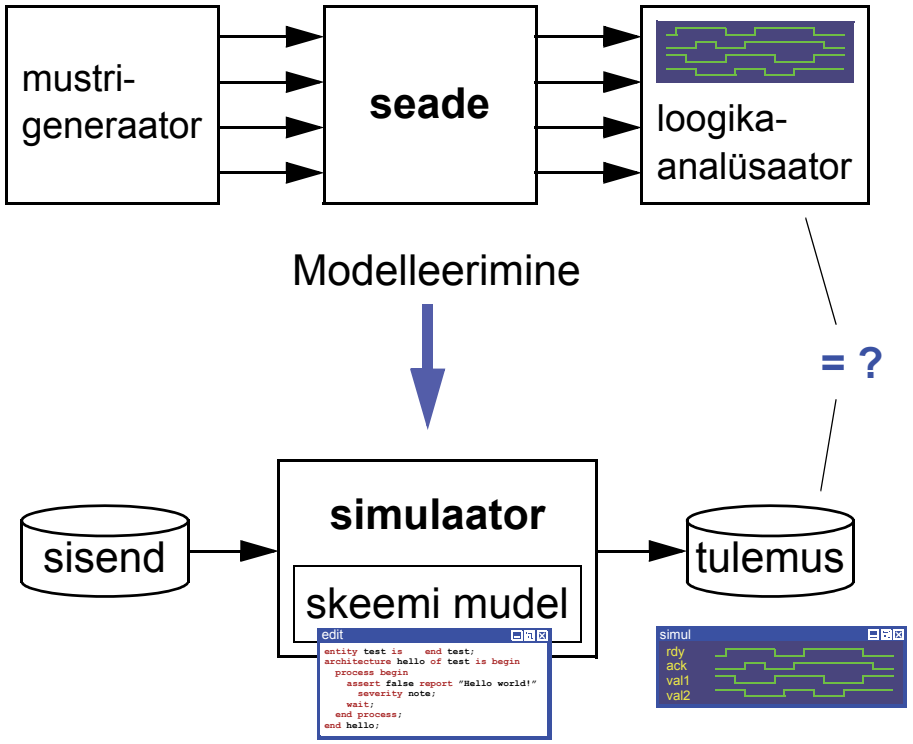
- **operatsiooni hind riistvaras**
  - **nihutamine == traatide teisiti viimine**
  - **liitmist ja lahutamist teostab sama seade – summaator (+ülekanne)**
- **sõltumatud operatsioonid võib täita korraga**
  - **tegelike andmesõltuvuste analüüs – nt. R2 nihutamise tulemus**



Tulemuse kontroll?

Digitaalsüsteemide modelleerimine

- **Modelleerimine**
  - **mudeli korrektsus?**
    - **simuleeritavus**
- **Kasutatav keel?**
  - **C++?**
  - **Paralleelsus!**
    - **C++ lõimed?**
    - **ADA?**
    - **Spetsiaalkeeled?**
    - **Veel üks keel! :-(**



Tarkvara vs. riistvara

- Tarkvara
  - üksik kontrollvoog
  - mitu kontrollvoogu
    - ühel protsessoril / mitmel protsessoril
    - hajutatud ülesanded – protsessorite vaheline kaugus
    - infovahetus – muutujad
    - sünkroniseerimine – semaforid jne.
- Riistvara
  - paralleelselt töötavad moodulid
  - infovahetus – signaalid / kanalid
  - sünkroniseerimine – erisignaalid / sündmused

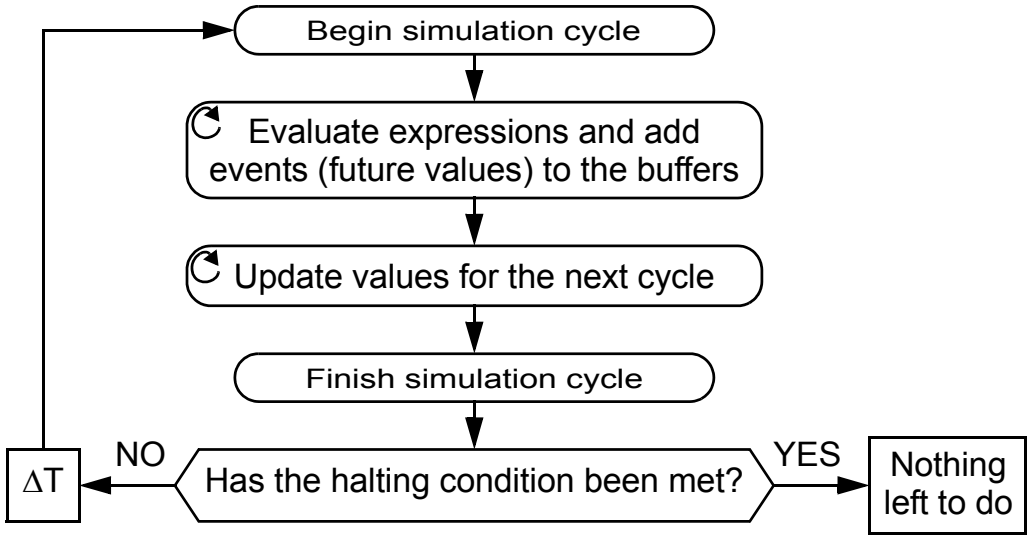
Riistvara modelleerimine

- Paralleelselt töötavad moodulid
- Mudel üksikul protsessoril
- Tulemus peaks olema korratav
- Lihtne põhimõte – signaalil uus ja vana väärtus
  - 1 - arvutatakse kõikide signaalide uued väärtused
  - 2 - signaalidele omistatakse uued väärtused
- Analoogsüsteemid
  - igal simulatsioonitsüklil uus arvutus
- Digitaalsüsteemid
  - uus arvutus ainult muutunud sisendite korral

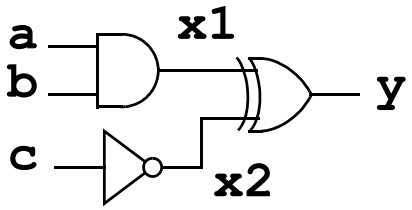
Digitaalsed simulaatorid

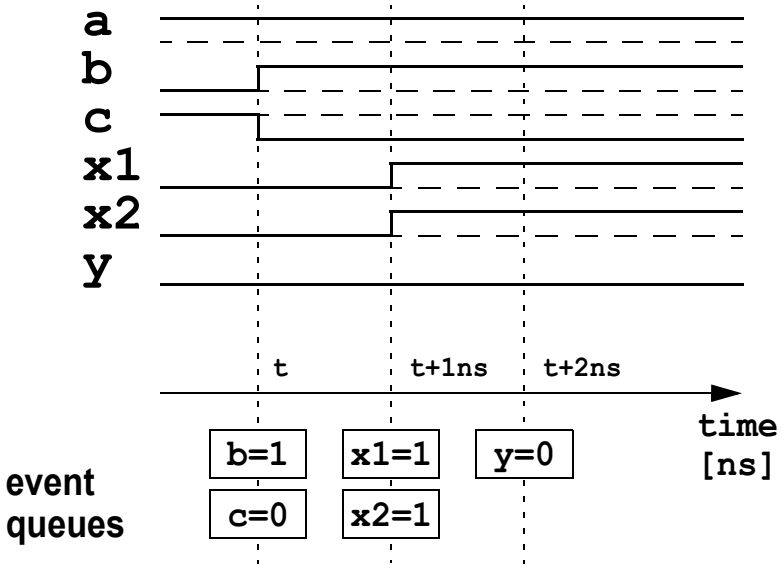
- Sama-aegsed operatsioonid
  - modelleerimine järjestikulises süsteemis
- Simulatsioonitsükkel
  - signaalide järgmiste väärtuste leidmine
  - signaalide jooksvate väärtuste värskendamine
- Tsükkelpõhised ja sündmustepõhised
  - cycle-based vs. event-based
- Viite modelleerimine
  - ühikviide (unit-delay)
  - nullviide (zero-delay)
  - deltaviide (delta-delay)

Ühikviide

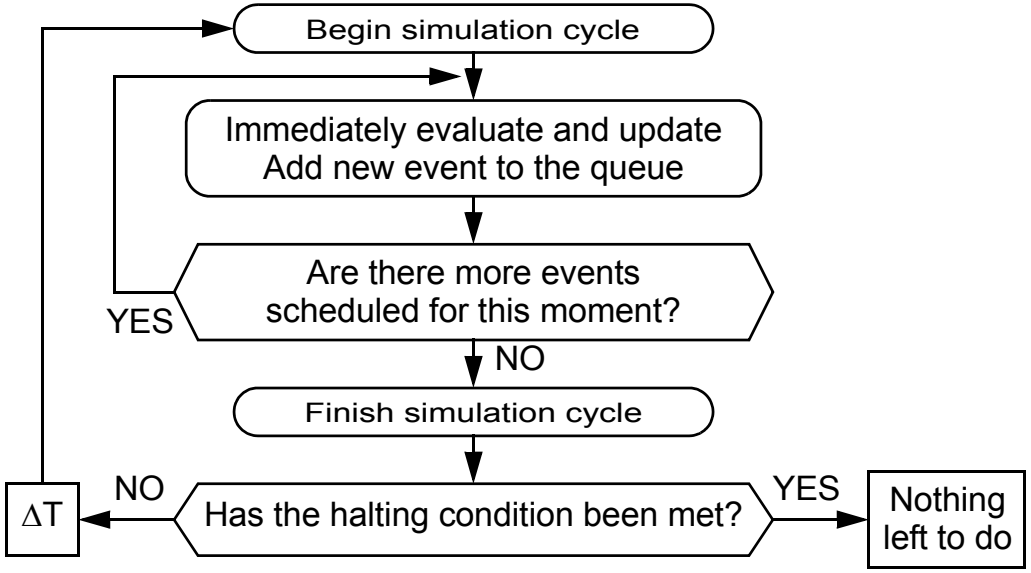


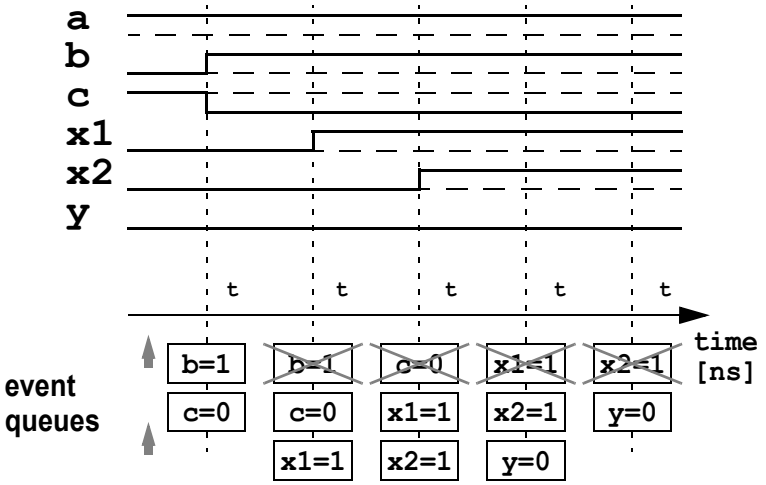

```
x1 <= a and b;
x2 <= not c;
y <= x1 xor x2;
```



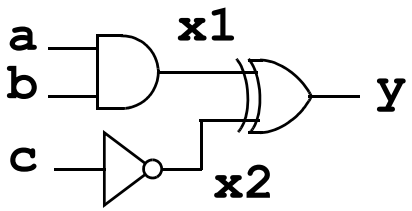


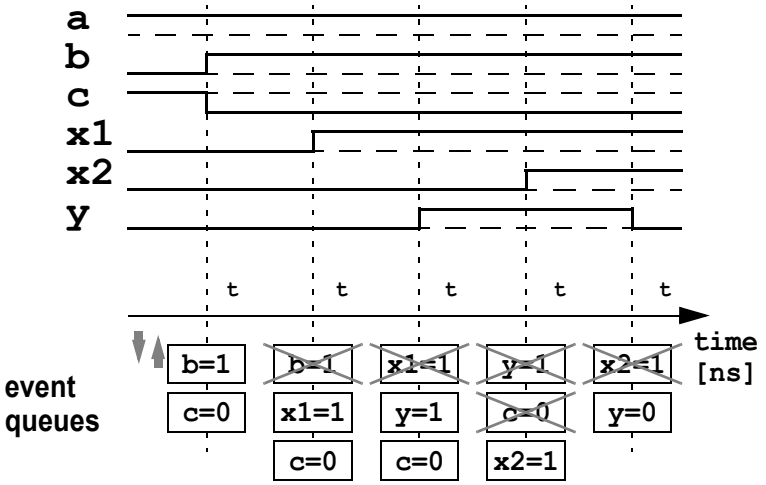
Nullviide



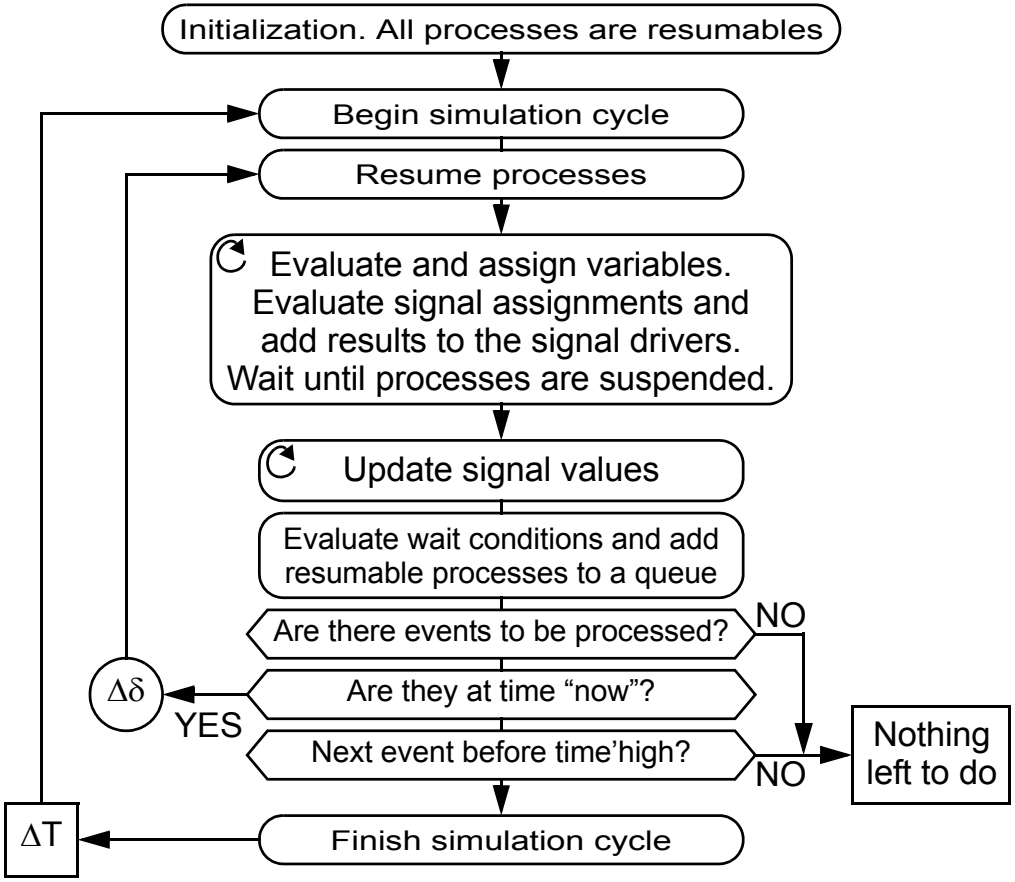



```
x1 <= a and b;
x2 <= not c;
y <= x1 xor x2;
```

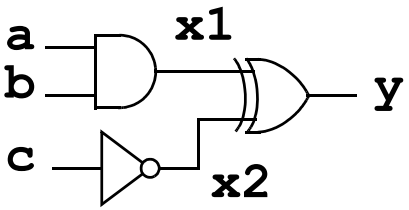


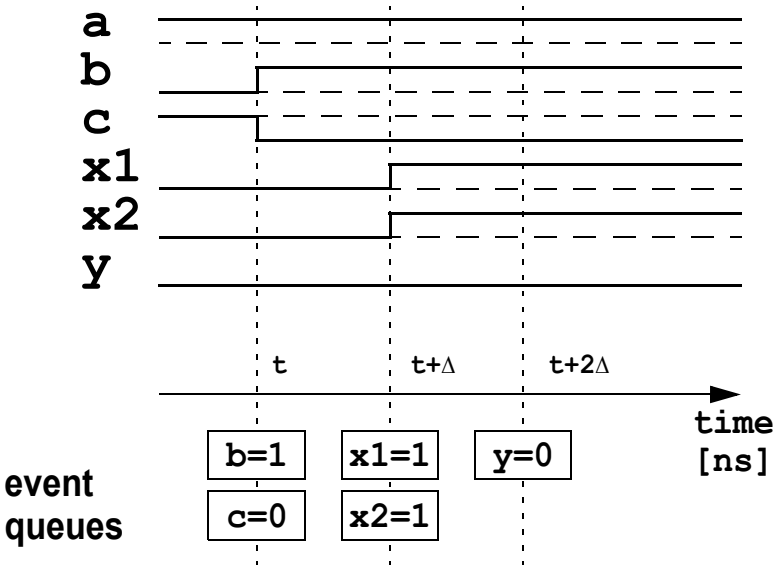


Deltaviide




```
x1 <= a and b;
x2 <= not c;
y <= x1 xor x2;
```



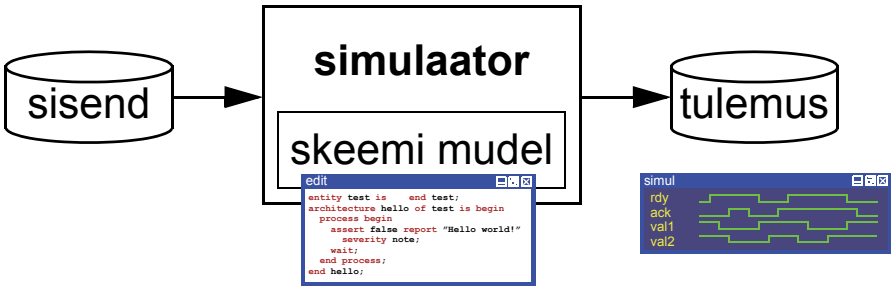


Motivatsioon

- Riistvara kirjelduskeel
  - Hardware Description Language (HDL)
- Erinevad abstraktsiooni-tasemed
  - Sama keel kõikidel projekteerimisetappidel
    - … või vähemalt enamikel projekteerimisetappidel
  - Sama mooduli erinevate kirjelduste võrreldavus
  - Simuleerimine erinevatel kirjeldustasemetel
- Dokumenteerimine
  - Modelleeritav spetsifikatsioon
- Süntees
  - Lähtekirjedus kõrgel abstraktsioonitasemel
  - Formaliseeritav teisendus kõrgemalt abstraktsioonitasemelt madalamale

Modelleerimine – VHDL & kodutöö #1

- **Teisenduste korrektsuse kontroll**
  - Tõeväärtustabel
  - Minimeeritud funktsioonid
  - Optimeeritud funktsioonid
    - pluss vahe-etapid
  - Sisendsignaalide generaator



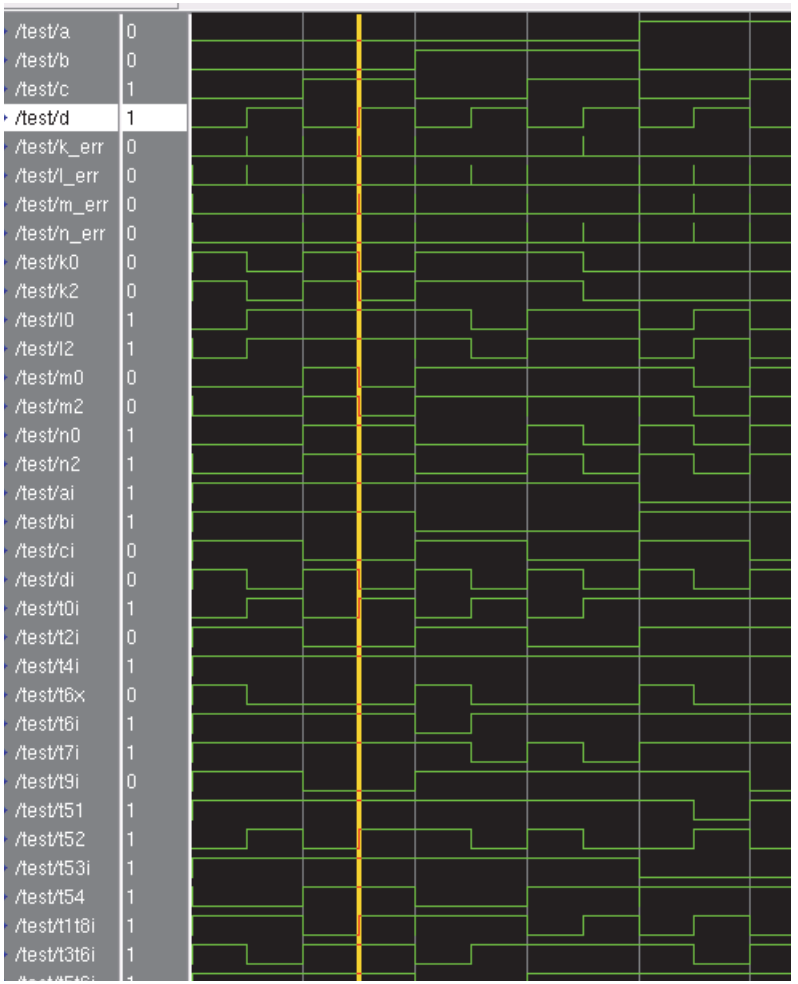

```
-- Sisendsignaalid (sammuga 10 ns)
a <= '0' after 0 ns, '1' after 80 ns, '0' after 160 ns;
b <= '0' after 0 ns, '1' after 40 ns, '0' after 80 ns, '1' after 120 ns;
c <= '0' after 0 ns, '1' after 20 ns, '0' after 40 ns, '1' after 60 ns,
     '0' after 80 ns, '1' after 100 ns, '0' after 120 ns, '1' after 140 ns;
d <= '0' after 0 ns, '1' after 10 ns, '0' after 20 ns, '1' after 30 ns,
     '0' after 40 ns, '1' after 50 ns, '0' after 60 ns, '1' after 70 ns,
     '0' after 80 ns, '1' after 90 ns, '0' after 100 ns, '1' after 110 ns,
     '0' after 120 ns, '1' after 130 ns, '0' after 140 ns, '1' after 150 ns;
```

  - Tulemuste analüüs / võrdlus
    - mitu realisatsiooni korraga?
    - VHDL – entity / architecture paarid

Modelleerimine – VHDL & kodutöö #1

- **Teisenduste korrektsuse kontroll**

```
entity test is   end test;
architecture bench of test is
  signal a, b, c, d, k_err, ...: bit;
  ...
begin
  -- Sisendsignaalid (sammuga 10 ns)
  a <= '0' after 0 ns, '1' after 80 ns, ...;
  ...
  -- Minimeerimise tulemus
  k0 <= ((not a) and (not d)) or (a and c and d)
     or ((not a) and b and (not c)) or
     (b and (not c) and (not d));
  ...
  -- Optimeerimise tulemus
  ai <= not (a and a);      bi <= not (b and b);
  ...
  t51 <= not (a and d);     t52 <= b xor d;
  t5t6i <= not (t51 and b and ci);
  k2 <= not (t0i and t4i and t5t6i);
  ...
  -- Kontrollimine...
  k_err <= k0 xor k2;    l_err <= l0 xor l2;
  ...
end bench;
```



Modelleerimine – VHDL & kodutöö #1

- **Mitme versiooni korraga simuleerimine**
  - **Iga vahe-etapp eraldi komponendina**

```
library IEEE; use IEEE.std_logic_1164.all;
entity f_system is
  port ( a, b, c, d: in std_logic;
         k, l, m, n: out std_logic );
end entity f_system;
architecture tabel of f_system is begin
  process (a, b, c, d)
    variable in_word, out_word:
      std_logic_vector (3 downto 0);
  begin
    in_word := a & b & c & d;
    case  in_word  is
      when "0000" => out_word := "1-00";
      when "0001" => out_word := "01-0";
      when "0010" => out_word := "11-1";
      when "0011" => out_word := "0-01";
      when "0100" => out_word := "1110";
      ...
      when "1110" => out_word := "-000";
      when "1111" => out_word := "1011";
      when others => out_word := "----";
    end case;
    k <= out_word(3);    l <= out_word(2);
    m <= out_word(1);    n <= out_word(0);
  end process;
end architecture tabel;
```

```
architecture espresso of f_system is begin
  k <= ((not a) and (not d)) or (a and c and d) o
       ((not a) and b and (not c)) or
       (b and (not c) and (not d));
  ...
  n <= ((not a) and c and (not d)) or
       (a and c and d) or
       (a and (not b) and (not d)) or
       ((not b) and c);
end architecture espresso;
```

```
architecture opti of f_system is
  signal ai, bi, ci, di: std_logic;
  signal t0i, t2i, ... t7i, t9i: std_logic;
  signal t51, t52, ... t3t6i, t5t6i: std_logic;
begin
  ai <= not (a and a);    bi <= not (b and b);
  ci <= not (c and c);    di <= not (d and d);
  t0i <= a or d;          t2i <= not (ai and c);
  ...
  t1t8i <= not (di and t54);
  t3t6i <= not (t52 and ci);
  t5t6i <= not (t51 and b and ci);
  k <= not (t0i and t4i and t5t6i);
  l <= not (t2i and t3t6i);
  m <= not (t1t8i and t6i and t7i);
  n <= not (t1t8i and t4i and t9i);
end architecture opti;
```

Modelleerimine – VHDL & kodutöö #1

- **Mitme versiooni korraga simuleerimine**

```
library IEEE; use IEEE.std_logic_1164.all;
entity test2 is    end entity test2;
library IEEE; use IEEE.std_logic_1164.all;
architecture bench of test2 is
  signal a, b, c, d, k, l, m, n: std_logic;
  signal k2, k3, l2, l3, m2, m3, n2, n3: std_logic;
  component f_system
    port ( a, b, c, d: in std_logic;
           k, l, m, n: out std_logic );
  end component;
  for U1: f_system use entity work.f_system(tabel);
  for U2: f_system use entity work.f_system(espresso);
  for U3: f_system use entity work.f_system(opti);
begin
  -- Input signaals (after every 10 ns)
  a <= '0' after 0 ns, '1' after 80 ns, '0' after 160 ns;
  b <= '0' after 0 ns, '1' after 40 ns, ...
  c <= '0' after 0 ns, '1' after 20 ns, ...
  d <= '0' after 0 ns, '1' after 10 ns, ...
  -- System of Boolean functions
  U1: f_system port map (a, b, c, d, k, l, m, n);
  U2: f_system port map (a, b, c, d, k2, l2, m2, n2);
  U3: f_system port map (a, b, c, d, k3, l3, m3, n3);
end architecture bench;
```