

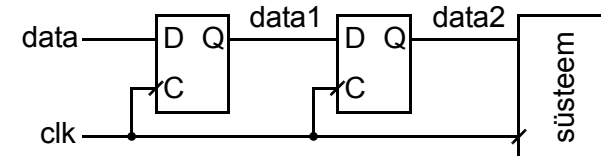
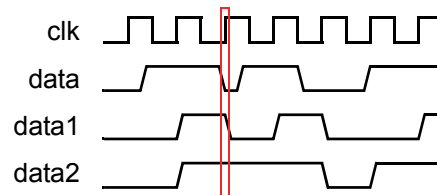
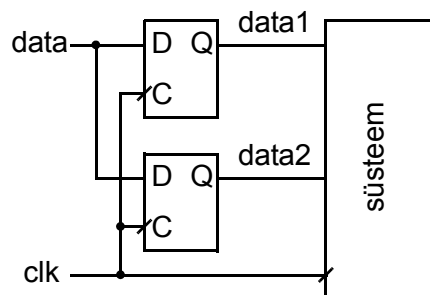
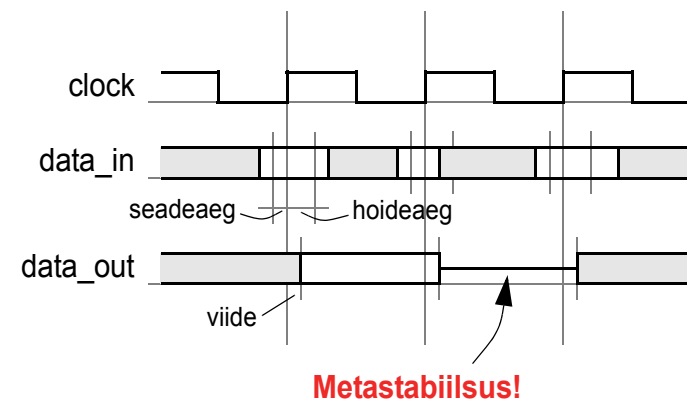


Asünkroonsed süsteemid

- **Asünkroonne andmevahetus**
 - andmevahetus moodulite vahel, mis kasutavad erinevaid taktsignaale
 - signaal peab olema stabiilne takti muutumisel (setup & hold)
 - andmevahetus süsteemis, kus signaali levimise aeg on võrreldav taktsignaali perioodiga
 - hajus-süsteemid
 - süsteem kiibil (SoC)
 - kõikjal sama probleem – signaali stabiilsuse tagamine
 - andmevahetus asünkroonsete (self-timed) moodulite vahel
 - asünkroonsed automaadid
- **Asünkroonseid süsteeme tuleks kasutada seal, kus on neist kasu!**

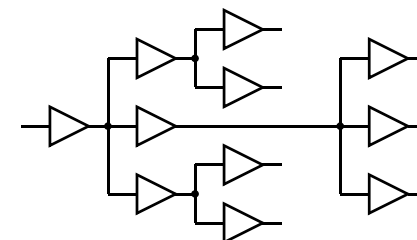
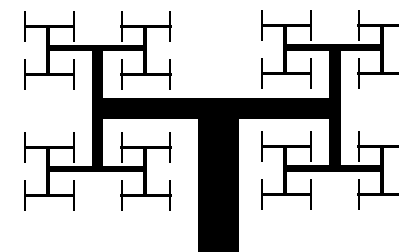
Asünkroonsed sisendid

- Signaalid moodulite vahel, mille taktsignaalid võivad oluliselt erineda
 - erinevate sagedustega taktsignaalid
 - üksteisest kaugel asuvad moodulid
 - füüsikalised efektid – nt. värelemine
- Vajalik on sisendite täiendav puhverdamine
 - vähendab metastabiilsuse levimise tõenäosust



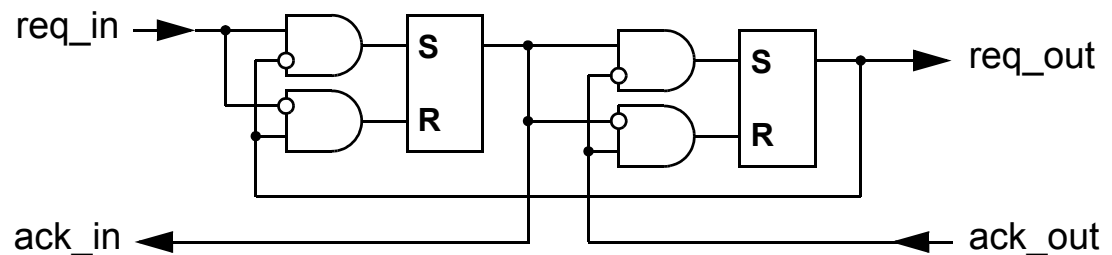
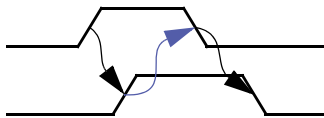
Sünkroonne või asünkroonne nullimine?

- **Sünkroonne nullimine**
 - **kõikidel mäluelementidel samal ajal**
 - taktsignaali aktiivne front viiakse kõikide mäluelementideni (enamvähem) samal ajal
 - **takti olemasolu on vajalik**
 - taktsignaali ei pruugi alguses olemas olla
- **Asünkroonne nullimine**
 - **takti olemasolu ei ole vajalik**
 - mõjub mäluelementidele kindlasti
 - **nullimis-signaal võib eri mäluelementidel lõppeda erineval ajal**
 - signaali levimine võib olla erinev – vaadeldakse tavalise signaalina, kus sama-aegsus pole kriitiline
- **Kombineeritud lähenemine – algus asünkroonne, lõpp sünkroonne**



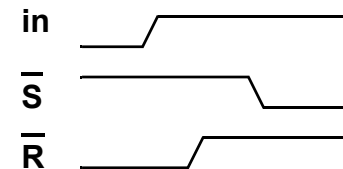
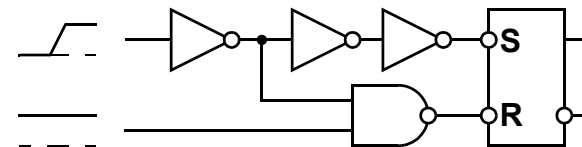
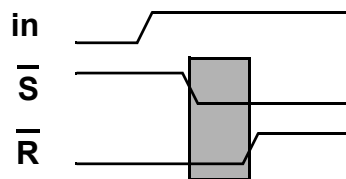
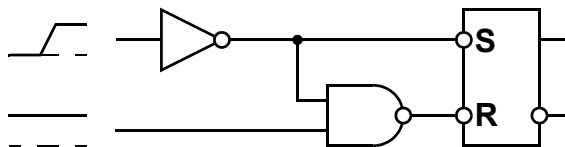
Asünkroonsed automaadid

- Taktsignaali puudub
- Muutus sisendil on sündmuseks → isetakteeruv
- Signaalide võistlus (signal race) – kriitilise tee analüüs väga oluline
 - sündmus saabub eri teid pidi – võib näida kahe või enama sündmusena
- Asünkroonne andmevahetus
- Ilesünkroniseerivad süsteemid



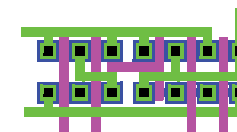
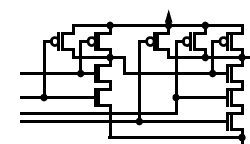
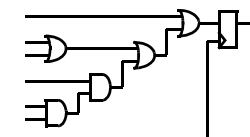
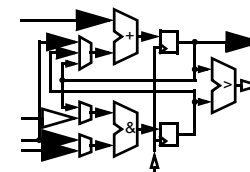
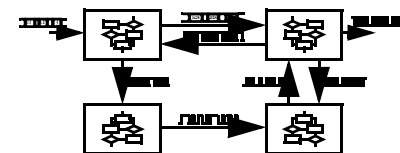
Asünkroonsed automaadid

- Automaadi tabel (GSA, olekudiagramm) aluseks
- Olekute kodeerimine
 - nn. naaberkoodid – erinevus ühes järgus
 - vähendab signaalide võistluse ohtu siirdel

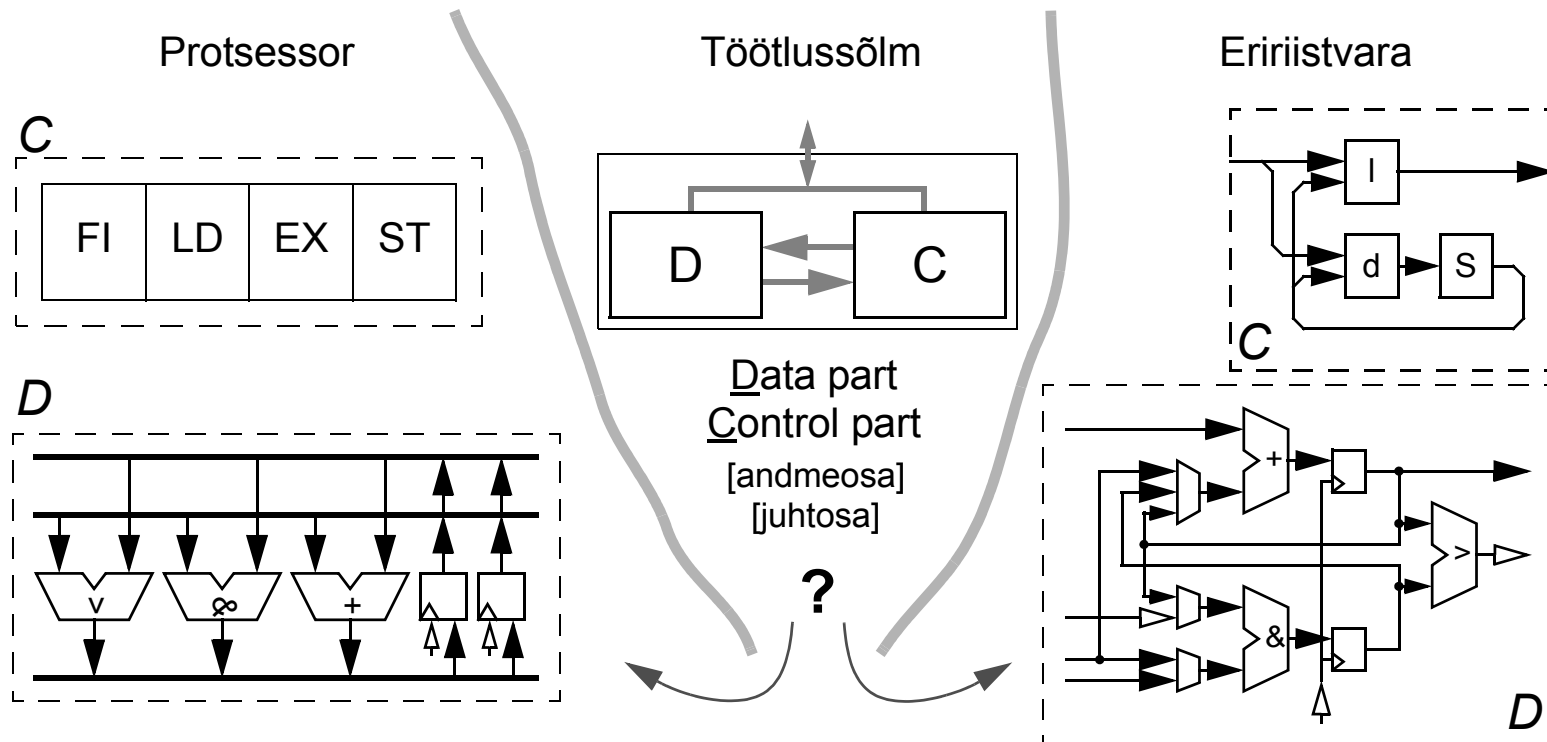


Digitaalsüsteemide realiseerimine

- **Süsteemi disain – System design**
a.k.a. Arhitektuuri süntees – Architectural-level synthesis
 - Süsteemi taseme süntees
 - tükeldamine & liideste süntees
 - Kõrgtaseme süntees
 - planeerimine, eraldamine & sidumine
- **Loogikadisain – Logic Design**
 - Registersiirete taseme süntees
 - andme- & juhtosa süntees
 - Loogikataseme süntees
 - loogika minimeerimine & optimeerimine
- **Füüsiline disain – Physical design**
a.k.a. Geomeetria süntees – Geometrical-level synthesis
 - Füüsikalise taseme süntees
 - teisendus loogikaelementideks
 - paigaldamine & trasseerimine/ruutimine



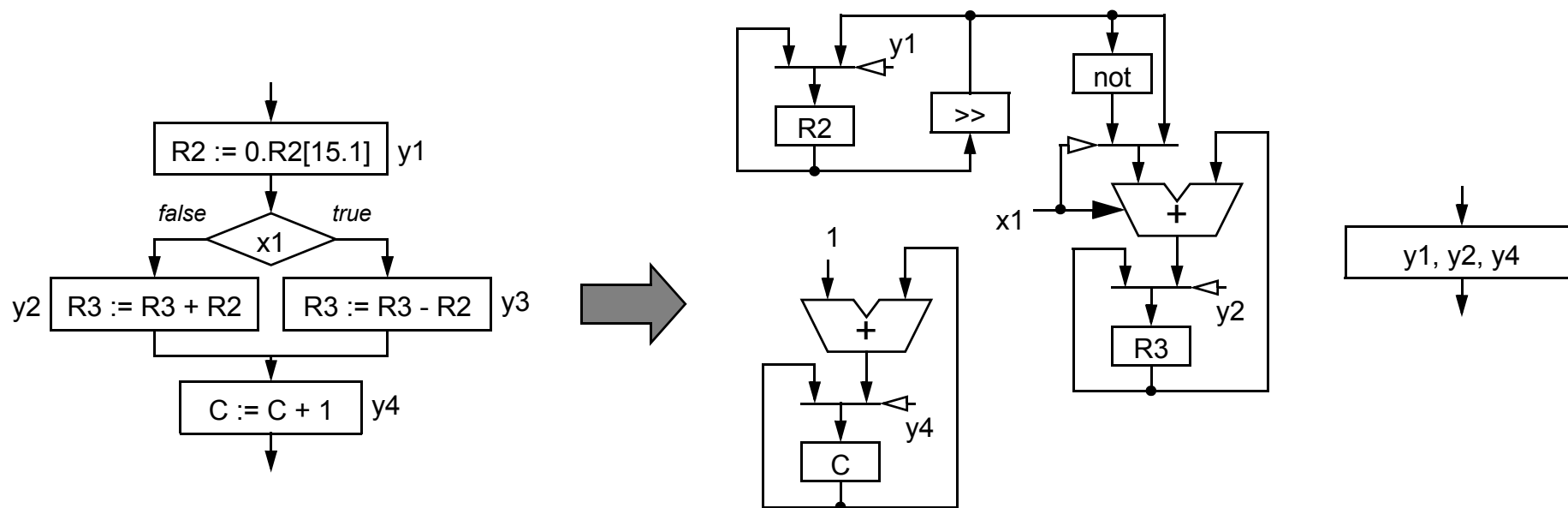
Tarkvara või riistvara / Üldotstarbeline või eridisain ?



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

Realisatsiooni optimaalsus?

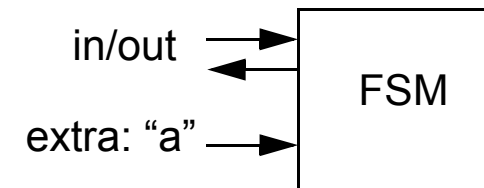
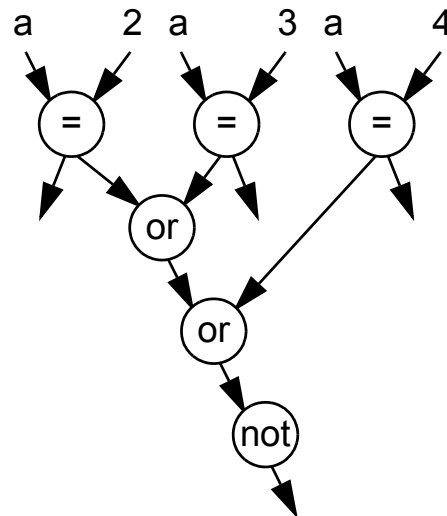
- operatsiooni hind riistvaras
 - nihutamine == traatide teisiti viimine
 - liitmist ja lahutamist teostab sama seade – summaator (+ülekanne)
- sõltumatud operatsioonid võib täita korraga
 - tegelike andmesõltuvuste analüüs – nt. R2 nihutamise tulemus



Andmeosa → juhtosa

- Osa operatsioonidest andmeosas teisendatakse operatsioonideks juhtosas
 - võrdlused konstantidega
 - siirdetingimuste avaldised (*if-then-else*, *case*, tsüklid)
 - optimeeritakse koos automaadiga – võib toimuda olekute arvu plahvatuslik kasv

```
case a is
when 2 => ...
when 3 => ...
when 4 => ...
when others => ...
end case;
```

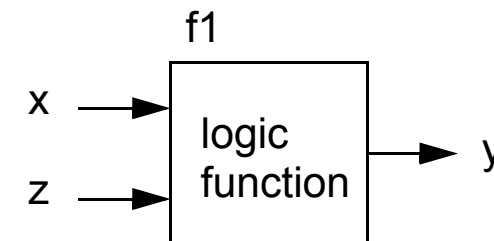
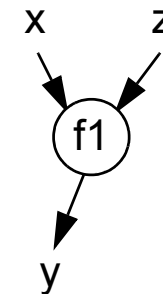
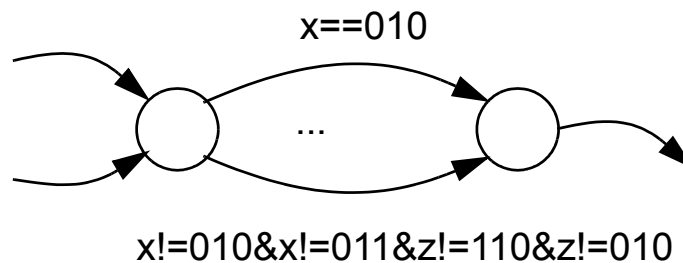


a&state:
010:000
011:000
100:000
.....

Juhtosa → andmeosa

- Osa operatsioonidest juhtosas teisendatakse operatsioonideks andmeosas
 - siirdetingimuste arvutuste grupeerimine
 - lihtsustab automaadi sünteesi
 - parem (osade) loogikafunktsioonide minimeerimine

```
if x="010" then y:="0110";
elsif x="011" and
      z="110" then y:="0101";
else
  if z="010" then y:="1001";
  else
    y:="1100";
  end if;
end if;
```



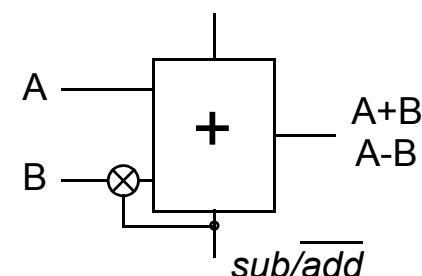
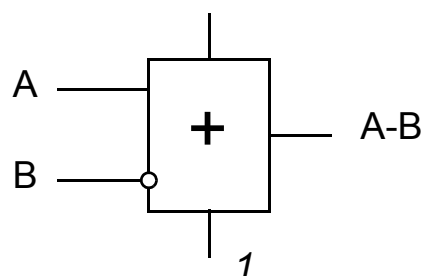
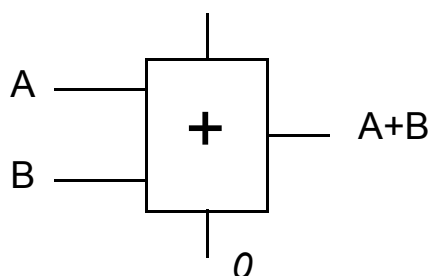
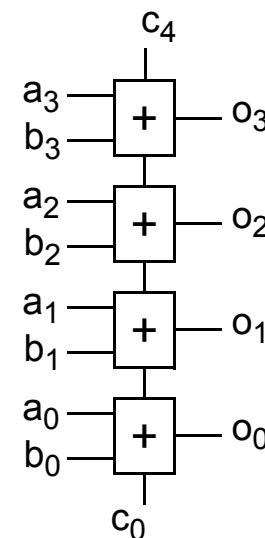
Aritmeetika-operatsioonide realiseerimine

Summator e. liitja

- poolsummator – $(c_i, o_i) = a_i + b_i$
 - $o_i = a_i \oplus b_i$ $c_i = a_i \& b_i$
- täis-summator – $(c_i, o_i) = a_i + b_i + c_{i-1}$
 - $o_i = a_i \oplus b_i \oplus c_i$ $c_i = (a_i \& b_i) \mid (a_i \& c_{i-1}) \mid (b_i \& c_{i-1})$

Lahutaja

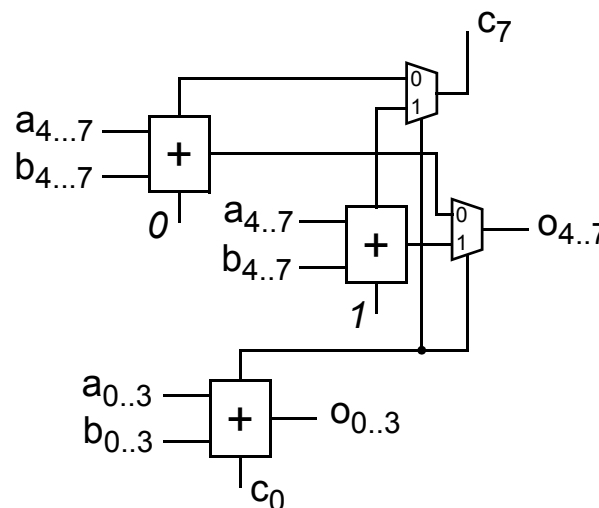
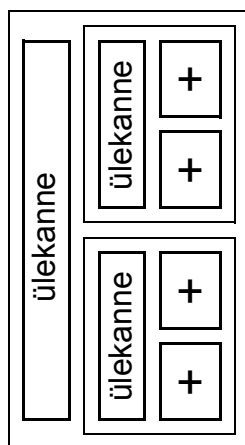
- täiendkood – $O = A - B = A + \overline{B} + 1$
- liitja-lahutaja – $O = n ? (A - B) : (A + B)$



Aritmeetika-operatsioonide realiseerimine

Ülekande kiirendamine

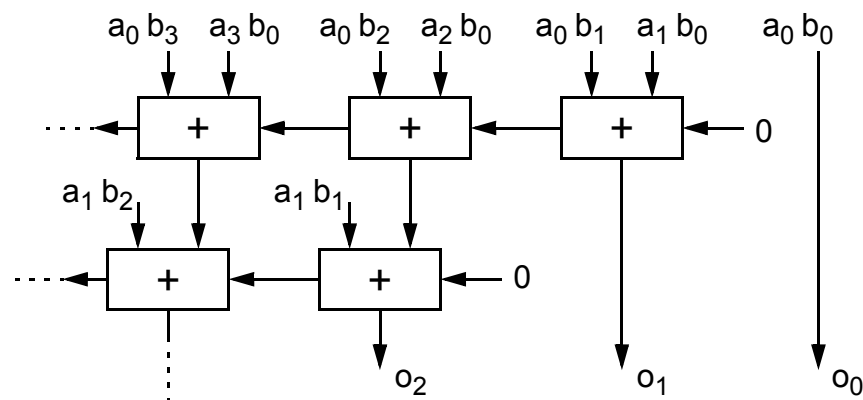
- jagatud gruppideks, grupi sees ülekanne eraldi funktsioonina
- “carry-lookahead”
 - “generate” – $g_i = a_i \& b_i$ “propagate” – $p_i = a_i \mid b_i$ “carry” – $c_i = g_i \mid (p_i \& c_{i-1})$
 - $g_{i+1} = a_{i+1} \& b_{i+1}$; $p_{i+1} = a_{i+1} \mid b_{i+1}$; $c_{i+1} = g_{i+1} \mid (p_{i+1} \& c_i)$
 - $c_{i+1} = g_{i+1} \mid (p_{i+1} \& (g_i \mid (p_i \& c_{i-1}))) \dots$
- “carry-select”
 - spekulatiivne arvutus – vanemates järkudes leitakse summa ja ülekanne edasi nii 0 kui 1 jaoks, tulemus valitakse multipleksoriga, kui ülekanne alt “lõpuks kohale jõuab”



Aritmeetika-operatsioonide realiseerimine

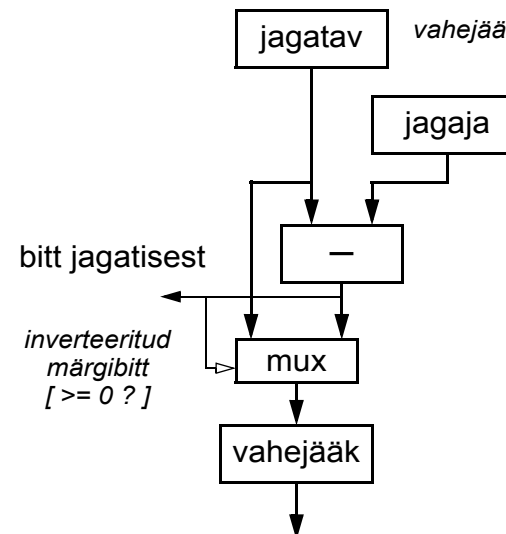
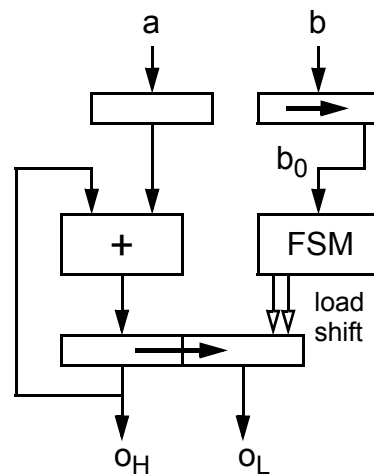
Korrutamine

- $A = a_{n-1} \cdot 2^{n-1} + a_{n-2} \cdot 2^{n-2} + \dots + a_2 \cdot 2^2 + a_1 \cdot 2^1 + a_0 \cdot 2^0$
- $B = b_{n-1} \cdot 2^{n-1} + b_{n-2} \cdot 2^{n-2} + \dots + b_2 \cdot 2^2 + b_1 \cdot 2^1 + b_0 \cdot 2^0$
- $A \cdot B = a_{n-1} \cdot 2^{n-1} \cdot (b_{n-1} \cdot 2^{n-1} + b_{n-2} \cdot 2^{n-2} + \dots + b_2 \cdot 2^2 + b_1 \cdot 2^1 + b_0 \cdot 2^0) +$
 $a_{n-2} \cdot 2^{n-2} \cdot (b_{n-1} \cdot 2^{n-1} + b_{n-2} \cdot 2^{n-2} + \dots + b_2 \cdot 2^2 + b_1 \cdot 2^1 + b_0 \cdot 2^0) + \dots$
 $a_2 \cdot 2^2 \cdot (b_{n-1} \cdot 2^{n-1} + b_{n-2} \cdot 2^{n-2} + \dots + b_2 \cdot 2^2 + b_1 \cdot 2^1 + b_0 \cdot 2^0) +$
 $a_1 \cdot 2^1 \cdot (b_{n-1} \cdot 2^{n-1} + b_{n-2} \cdot 2^{n-2} + \dots + b_2 \cdot 2^2 + b_1 \cdot 2^1 + b_0 \cdot 2^0) +$
 $a_0 \cdot 2^0 \cdot (b_{n-1} \cdot 2^{n-1} + b_{n-2} \cdot 2^{n-2} + \dots + b_2 \cdot 2^2 + b_1 \cdot 2^1 + b_0 \cdot 2^0) =$
 $a_{n-1} \cdot b_{n-1} \cdot 2^{2n-2} + a_{n-1} \cdot b_{n-2} \cdot 2^{2n-3} + \dots + a_0 \cdot b_2 \cdot 2^2 + a_0 \cdot b_1 \cdot 2^1 + a_0 \cdot b_0 \cdot 2^0$



Aritmeetika-operatsioonide realiseerimine

- **Korrutamine**
 - **paralleelne – kombinatsiooniskeem**
 - $m \cdot n$ bitti – m n -bitist summaatorit ja $m \cdot n$ JA-elementi
 - kombinatoorskeem liiga suure viitega – abiregistrid & konveier (pipeline)
 - **järjestikuline – järk-järgu kaupa tsükliliselt**
 - akumulaator ja juhtautomaat
 - mitu järku korraga – juhtloogika lisaks
- **Jagamine – tavaliselt tsükliliselt – akumulaator, juhtloogika ja juhtautomaat**



<http://www.ecs.umass.edu/ece/koren/arith/simulator/>

Ripple Carry Adder With Timing - Netscape

File Edit View Go Bookmarks Tools Window Help

http://www.ecs.umass.edu/ece/koren/arith/simulator/Add/ripple/ripple.html

Home Search Bookmarks

Ripple Carry Adder With Timing

Ripple Carry Adder

A: 00101110
B: 01100010

bin dec

Number of Bits: 8

Signal Delays

A_i to C_{i+1}	2.2	C_i to C_{i+1}	1.9
C_i to S_i	2.5	A_i to S_i	2.6

Compute

Reset

Time taken to generate all Sum bits - (16.1)units
Time taken to generate all Carryout - (15.5)units

Sum 10010000 :144

Delays to calculate Sum in each bit position

BitNumber	7	6	5	4	3	2	1	0
SumDelay	(16.1)units	(14.2)units	(12.3)units	(10.4)units	(8.5)units	(6.6)units	(4.7)units	(2.6)units

Carry Look Ahead Adder With Timing - Netscape

File Edit View Go Bookmarks Tools Window Help

http://www.ecs.umass.edu/ece/koren/arith/simulator/Add/lookahead/

Home Search Bookmarks

Carry Look Ahead Adder With Timing

Carry Look Ahead Adder

A: 00101110
B: 01100010

bin dec

Total Bits: 8 Group Size: 4

Signal Delays

AND/OR Gate Delay	1.0
XOR Gate Delay	1.6
Maximum Fan-in	4

Compute

Reset

Time taken to generate all Sum bits - (10.2)units

Sum 10010000 :144

Array Multiplier With Timing - Netscape

File Edit View Go Bookmarks Tools Window Help

http://www.ecs.umass.edu/ece/koren/arith/simulator/ArMMV/

Home Search Bookmarks

Array Multiplier With Timing

2's Complement Array Multiplier

A: 00101
X: 11001

bin dec

Number of Bits (>2): 5

Signal Delays

A_i to C_{i+1}	1.5	C_i to C_{i+1}	1.0
C_i to S_i	1.5	A_i to S_i	2.0

Compute

Reset

Help

Product 1111011101 :-35

Total Delay for Multiplier=12



Algoritmi realiseerimine

- **Algoritm kui programm**
 - operatsioonid – andmeosa
 - järjestus ja tingimuse – 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?
 - $A < B \implies A - B < 0$ / $A \neq B \implies A - B \neq 0$ – kõike saab teha lahutajaga?!
 - sama riistvara kõikidele operatsioonidele või korraga arvutamine?
 - kiirus või suurus? [ja kas see ongi probleem?]



Näide #1 – GCD (Greatest Common Divisor)

- Spetsifikatsioon ~~ käitumuslik kirjeldus
 - sisendi/väljundi ajastus fikseeritud – juht- ja takt-signaalid

```

process -- gcd-bhv.vhdl
  variable x, y: unsigned(15 downto 0);
begin
  -- Wait for the new input data
  wait on clk until clk='1' and rst='0';
  x := xi;    y := yi;    rdy <= '0';
  wait on clk until clk='1';
  -- Calculate
  while x /= y loop
    if x < y then y := y - x;
    else        x := x - y;    end if;
  end loop;
  -- Ready
  xo <= x;    rdy <= '1';
  wait on clk until clk='1';
end process;
  
```

Probleemid

- tsüklis puudub takt
- keerukas “wait” käsk
- (- mitu “wait” käsku)

Mida proovida?

- erinevad süntesaatorid
- suuruse minimeerimine
- kiiruse tõstmine

Tehnoloogiad – ASIC, FPGA

VHDL kood & testpingid

<http://mini.pld.ttu.ee/~lrv/gcd/>

GCD – sünteesitav kood?

- Takteeritud käitumuslik stiil

```
process -- gcd-bhvc.vhdl
    variable x, y: unsigned(15 downto 0);
begin
    -- Wait for the new input data
    while rst = '1' loop
        wait on clk until clk='1';
    end loop;
    x := xi;    y := yi;    rdy <= '0';
    wait on clk until clk='1';
    -- Calculate
    while x /= y loop
        if x < y then y := y - x;
        else          x := x - y;    end if;
        wait on clk until clk='1';
    end loop;
    -- Ready
    xo <= x;    rdy <= '1';
    wait on clk until clk='1';
end process;
```

ASIC: sünteesitav

961 e.g. / 20.0 ns

2 lahutajat, 2 võrdlejat

FPGA: mitte-sünteesitav

“wait” käsud tsüklis :(
juhtautomaat vajalik :(:(

Võimalikud valikud

- funktsioonide jagamine
- universaalsed funktsioonid
- spekulatiivne arvutamine

GCD – käitumuslik automaat (behavioral FSM)

```
process begin -- gcd-bfsm.vhdl
    wait on clk until clk='1';
    case state is
        -- Wait for the new input data
    when S_wait =>
        if rst='0' then
            x<=xi; y<=yi; rdy<='0'; state<=S_start;
        end if;
        -- Calculate
    when S_start =>
        if x /= y then
            if x < y then y <= y - x;
            else x <= x - y; end if;
            state<=S_start;
        else
            xo<=x; rdy<='1'; state<=S_ready;
        end if;
        -- Ready
    when S_ready => state<=S_wait;
    end case;
end process;
```

ASIC: sünteesitav

911 e.g. / 19.4 ns

2 lahutajat, 2 võrdlejat

FPGA: sünteesitav

108 SLC / 9.9 ns

2 lahutajat, 2 võrdlejat

Kas saab teha paremaks?

Jälle need võimalikud valikud

- funktsioonide jagamine
 - üks operatsioon taktis
- universaalsed funktsioonid
 - $A < B \implies A - B < 0$ / $A \neq B \implies A - B \neq 0$
- spekulatiivne arvutamine
 - lahutamine enne otsustamist

GCD – universaalsed funktsioonid?

- $A < B == A - B < 0$ / $A \neq B == A - B \neq 0$

```
-- Three operations:
-- subtraction, and
-- comparisons not-equal &
-- less-than
xo <= xi - yi;

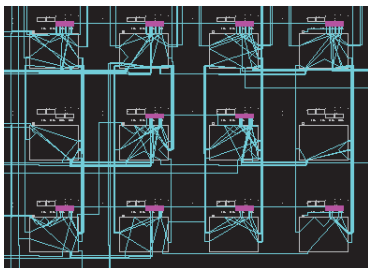
ne <= '1' when xi /= yi else '0';

lt <= '1' when xi < yi else '0';
```

ASIC: 209 e.g. / 19.9 ns

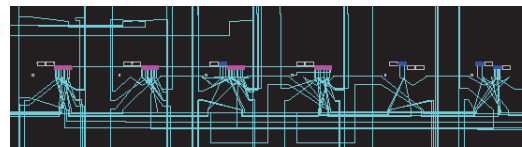
FPGA: 20 SLC / 12.1 ns

kolm liitjat!



```
-- ALU - subtracting and then comparing
x_out <= xi - yi;    xo <= x_out;
process (x_out)
    variable or_tmp: unsigned(15 downto 0);
begin
    or_tmp(15) := x_out(15);
    for i in 14 downto 0 loop
        or_tmp(i) := or_tmp(i+1) or x_out(i);
    end loop;
    ne <= to_bit(or_tmp(0));
end process;
lt <= to_bit(x_out(15));
```

ASIC: 148 e.g. / 21.8 ns / FPGA: 12 SLC / 14.6 ns



GCD – disainiruumi uurimine

- Erinevad lahendused – <http://mini.pld.ttu.ee/~lrv/gcd/>
 - gcd-bhv.vhdl – puhas käitumuslik kirjeldus, mitte-sünteesitav
 - gcd-bhvc.vhdl – takteeritud käitumuslik kirjeldus, osad süntesaatorid OK
 - gcd-bfsm.vhdl – nn. käitumuslik automaat (automaat & käitumuslik andmetee), sünteesitav (aga kui efektiivne see on?)
 - gcd-rtl1.vhdl – üks ALU, 3 takti iteratsiooni kohta –
1) “pole võrdne?”, 2) “väiksem kui?”, 3) lahutaja

```
#1#    a=x-y;    a!=0 ? #2# : #ready#
#2#    a=x-y;    a<0 ? #3# : #4#
#3#    y=y-x    /    #4#    x=x-y
```

- gcd-rtl2.vhdl – üks ALU, 2 takti iteratsiooni kohta –
1) “pole võrdne?” ja “väiksem kui?”, 2) lahutaja

```
#1#    a=x-y;    a!=0 ? ( a<0 ? #2# : #3# ) : #ready#
#2#    y=y-x    /    #3#    x=x-y
```



GCD – disainiruumi uurimine (2)

- Erinevad lahendused – <http://mini.pld.ttu.ee/~lrv/gcd/>
- gcd-rtl3.vhdl – võrdleja (väiksem kui) kontrollib lahutajat, 1 takt iteratsiooni kohta – väike kuid aeglane (jadamisi) andmetee


```
#1#      x<y ? y=y-x : ( a=x-y;   a!=0 ? x=x-y : #ready# )
```
- gcd-rtl4.vhdl – spekuleeriv arvutamine – mõlemad lahutamised tehakse kõigepealt, siis toimub otsustamine (üks lahutaja võrdleb “väiksem kui”, teine “pole võrdne”)


```
#1#      a1=x-y;   a2=y-x;   a2!=0 ? ( a1<0 ? y=a2 : x=a1 ) : #ready#
```
- gcd-rtl5.vhdl – spekuleeriv arvutamine – mõlemad lahutamised tehakse kõigepealt, siis toimub otsustamine (üks lahutaja võrdleb “väiksem kui”, kuid eraldi “pole võrdne”)


```
#1#      a1=x-y;   a2=y-x;   x!=y ? ( a1<0 ? y=a2 : x=a1 ) : #ready#
```

GCD – üksik ALU, 2 takti iteratsiooni kohta

gcd-rtl2.vhdl

```
-- Next state function of the FSM
process (state, rst, alu_ne, alu_lt) begin
    ena_x <= '0';      ena_y <= '0';      ena_r <= '0';
    set_rdy <= '0';    xi_yi_sel <= '0';    sub_y_x <= '0';
    next_state <= state;
    case state is
        when S_wait =>      -- Wait for the new input data
            if rst='0' then
                xi_yi_sel <= '1';    ena_x <= '1';    ena_y <= '1';
                next_state <= S_start;
            end if;
        when S_start =>      -- Loop: ready?
            if alu_ne='1' then
                if alu_lt='1' then    next_state <= S_sub_y_x;
                else                  next_state <= S_sub_x_y;    end if;
            else                  next_state <= S_ready;
            end if;
        when S_sub_y_x =>      -- Loop: y-x
            ena_y <= '1';    sub_y_x <= '1';
            next_state <= S_start;
        when S_sub_x_y =>      -- Loop: x-y
            ena_x <= '1';    sub_y_x <= '0';
            next_state <= S_start;
        when S_ready =>      -- Ready
            ena_r <= '1';    set_rdy <= '1';
            next_state <= S_wait;
    end case;
end process;

-- ALU: subtract / less-than / not-equal
alu_o <= alu_1 - alu_2;
alu_lt <= to_bit(alu_o(15));
process (alu_o)
    variable or_tmp: unsigned(15 downto 0);
begin
    or_tmp(15) := alu_o(15);
    for i in 14 downto 0 loop
        or_tmp(i) := or_tmp(i+1) or alu_o(i);
    end loop;
    alu_ne <= to_bit(or_tmp(0));
end process;

-- Multiplexers
x_i <= xi when xi_yi_sel='1' else alu_o;
y_i <= yi when xi_yi_sel='1' else alu_o;
alu_1 <= y when sub_y_x='1' else x;
alu_2 <= x when sub_y_x='1' else y;

-- Registers
process begin
    wait on clk until clk='1';
    state <= next_state;
    if ena_x='1' then    x <= x_i;    end if;
    if ena_y='1' then    y <= y_i;    end if;
    if ena_r='1' then    xo <= x;    end if;
    rdy <= set_rdy;
end process;
```

GCD – komparaator+lahutaja, 1 takt

gcd-rtl3.vhdl

```
-- Next state function of the FSM
process (state, rst, alu_ne) begin
    ena_xy <= '0';    ena_r <= '0';
    set_rdy <= '0';    xi_yi_sel <= '0';
    next_state <= state;
    case state is
    when S_wait =>    -- Wait for the new input data
        if rst='0' then
            xi_yi_sel <= '1';    ena_xy <= '1';
            next_state <= S_start;
        end if;
    when S_start =>    -- Calculate
        if alu_ne='1' then
            ena_xy <= '1';    next_state <= S_start;
        else
            ena_r <= '1';    set_rdy <= '1';
            next_state <= S_ready;
        end if;
    when S_ready =>    -- Ready
        ena_r <= '1';    set_rdy <= '1';
        next_state <= S_wait;
    end case;
end process;

-- Comparator (less-than)
sub_y_x <= '1' when x < y else '0';

-- Subtractor (+not-equal)
alu_o <= alu_1 - alu_2;
process (alu_o)
    variable or_tmp: unsigned(15 downto 0);
begin
    or_tmp(15) := alu_o(15);
    for i in 14 downto 0 loop
        or_tmp(i) := or_tmp(i+1) or alu_o(i);
    end loop;
    alu_ne <= to_bit(or_tmp(0));
end process;

-- Multiplexers
x_i <= xi when xi_yi_sel='1' else alu_o;
y_i <= yi when xi_yi_sel='1' else alu_o;
alu_1 <= y when sub_y_x='1' else x;
alu_2 <= x when sub_y_x='1' else y;
ena_x <= '1' when (sub_y_x='0' and ena_xy='1') or
    xi_yi_sel='1' else '0';
ena_y <= '1' when (sub_y_x='1' and ena_xy='1') or
    xi_yi_sel='1' else '0';

-- Registers
process begin
    wait on clk until clk='1';
    state <= next_state;
    if ena_x='1' then    x <= x_i;    end if;
    if ena_y='1' then    y <= y_i;    end if;
    if ena_r='1' then    xo <= x;    end if;
    rdy <= set_rdy;
end process;
```


GCD – spekuleeriv arvutamine (2 lahutajat)

gcd-rtl5.vhdl

```
-- Next state function of the FSM
process (state, rst, alu_ne) begin
    ena_xy <= '0';      ena_r <= '0';
    set_rdy <= '0';     xi_yi_sel <= '0';
    next_state <= state;
    case state is
        -- Wait for the new input data
        when S_wait =>
            if rst='0' then
                xi_yi_sel <= '1';      ena_xy <= '1';
                next_state <= S_start;
            end if;
        -- Calculate
        when S_start =>
            if alu_ne='1' then
                ena_xy <= '1';
                next_state <= S_start;
            else
                ena_r <= '1';      set_rdy <= '1';
                next_state <= S_ready;
            end if;
        -- Ready
        when S_ready =>
            ena_r <= '1';      set_rdy <= '1';
            next_state <= S_wait;
        end case;
    end process;
```

```
-- Subtractor (x-y) / comparator (x<y)
alu_o1 <= x - y;
sub_y_x <= '1' when alu_o1(alu_o1'high)='1' else '0';

-- Subtractor (y-x)
alu_o2 <= y - x;

-- Comparator (y/=x)
alu_ne <= '1' when x /= y else '0';

-- Multiplexers
x_i <= xi when xi_yi_sel='1' else alu_o1;
y_i <= yi when xi_yi_sel='1' else alu_o2;
ena_x <= '1' when (sub_y_x='0' and ena_xy='1') or
    xi_yi_sel='1' else '0';
ena_y <= '1' when (sub_y_x='1' and ena_xy='1') or
    xi_yi_sel='1' else '0';

-- Registers
process begin
    wait on clk until clk='1';
    state <= next_state;
    if ena_x='1' then      x <= x_i;      end if;
    if ena_y='1' then      y <= y_i;      end if;
    if ena_r='1' then      xo <= x;      end if;
    rdy <= set_rdy;
end process;
```

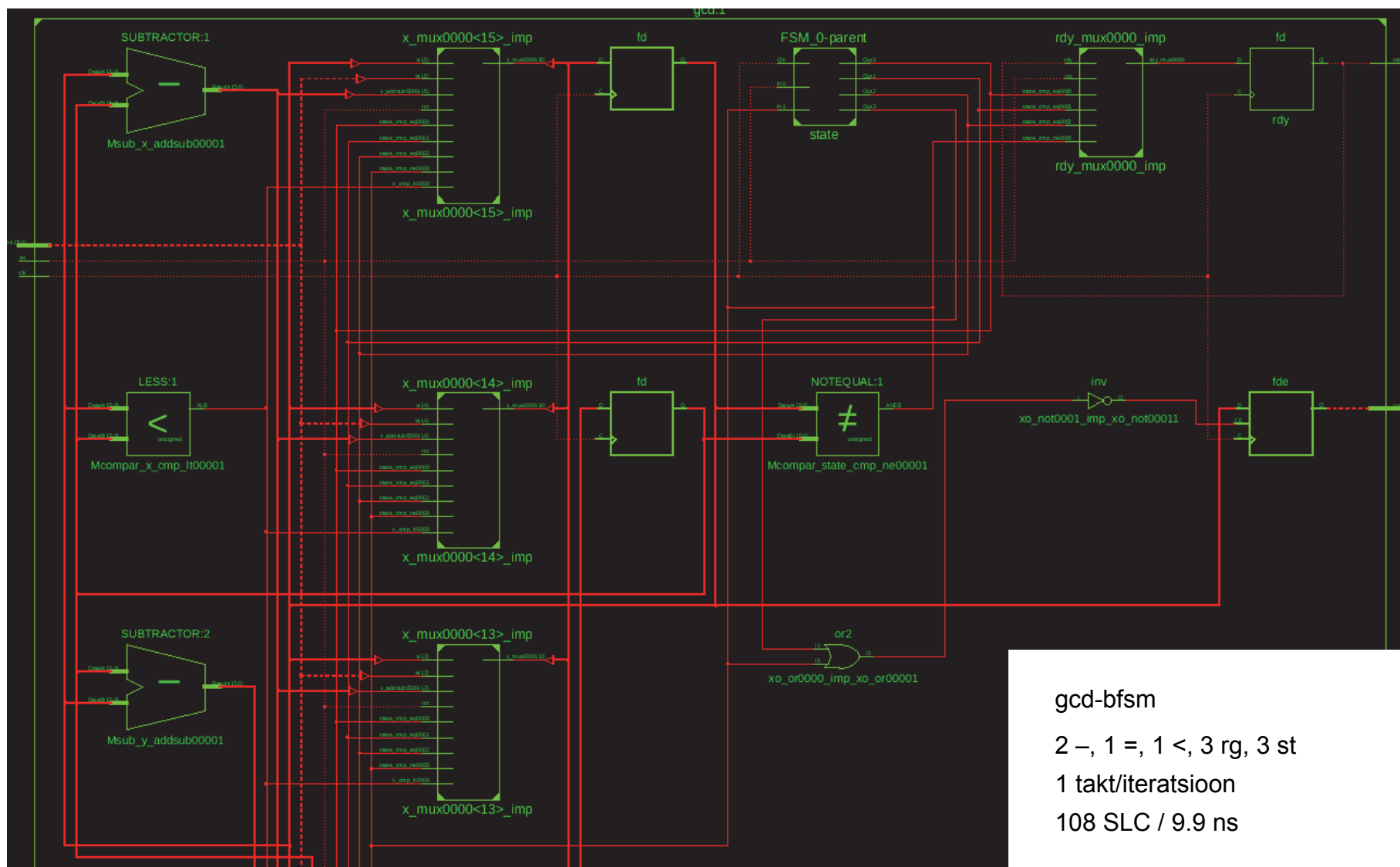
GCD – sünteeside tulemused

Tehnoloogia		FPGA				ASIC			
Piirangud ¹⁾		50 MHz		100 MHz		50 MHz		25 MHz	
	clk / FU	[SLC]	[ns]	[SLC]	[ns]	[e.g.]	[ns]	[e.g.]	[ns]
<i>gcd-bhv</i> ²⁾	? / ?	93	17.3	-	-	1141	20.0	-	-
gcd-bhvc	1 / 2+2	-	-	-	-	961	20.0	977	31.1
gcd-bfsm	1 / 2+2	108	9.9	108	9.4	911	19.4	984	30.8
gcd-rtl1	3 / 1	50	10.8	50	9.7	986	19.8	883	32.4
gcd-rtl2	2 / 1	48	10.8	48	10.0	931	19.9	882	32.3
gcd-rtl3	1 / 1+1	58	17.0	58	14.6	1134	20.0	928	40.0
gcd-rtl4	1 / 2	78	12.6	78	9.0	976	19.9	928	29.0
gcd-rtl5	1 / 2+1	58	8.0	58	7.6	915	20.0	932	26.9

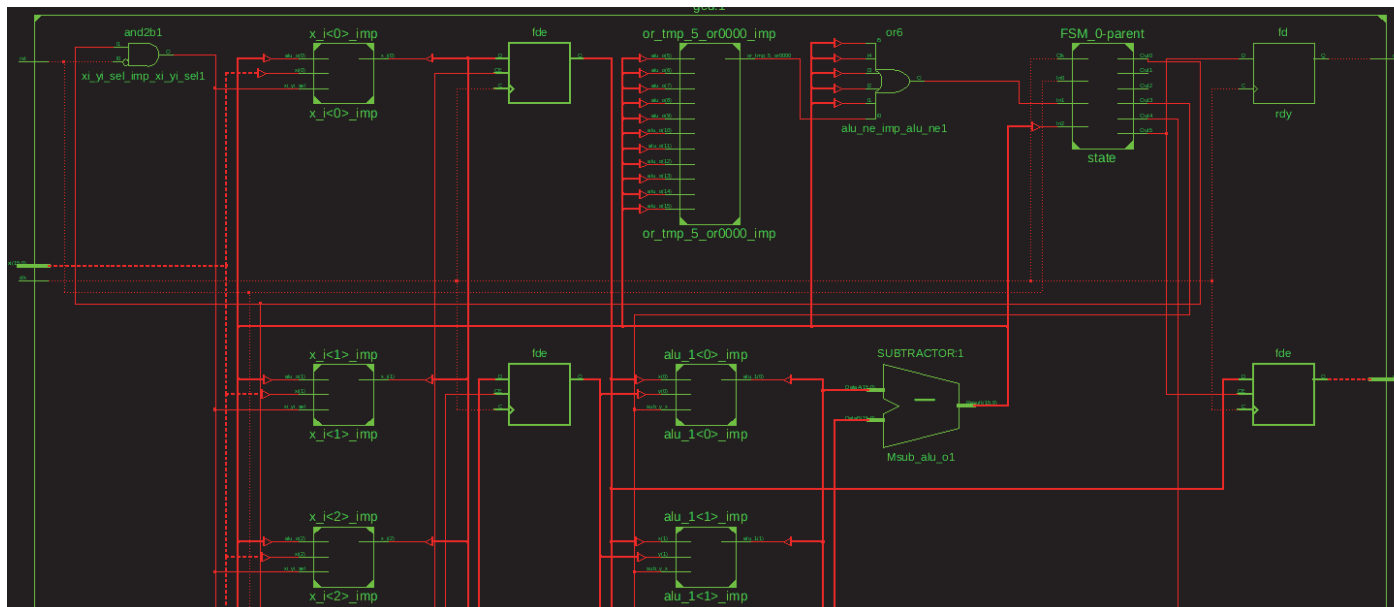
¹⁾ Taktiperiood oli ainuke piirang

²⁾ gcd-bhv sünteesiti prototüüp kõrgtasemesüntesaatoriga xTractor

GCD – kust tulevad need erinevused?



GCD – kust tulevad need erinevused?



gcd-rtl1

1 ALU, 3 rg, 6 st

3 takti/iteratsioon

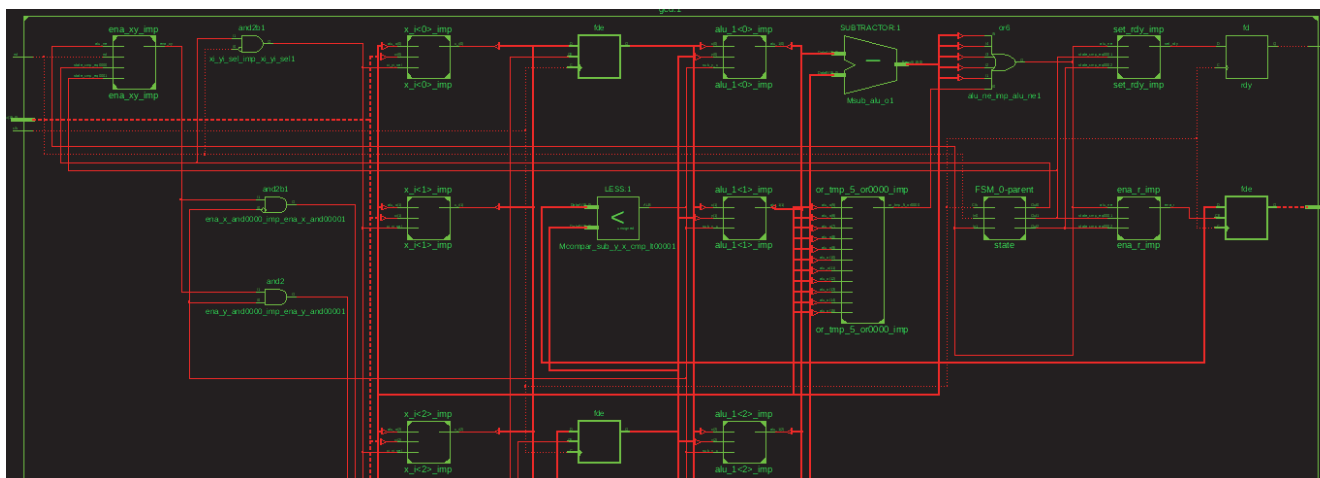
50 SLC / 10.8 ns

gcd-rtl2

1 ALU, 3 rg, 5 st

2 takti/iteratsioon

48 SLC / 10.8 ns



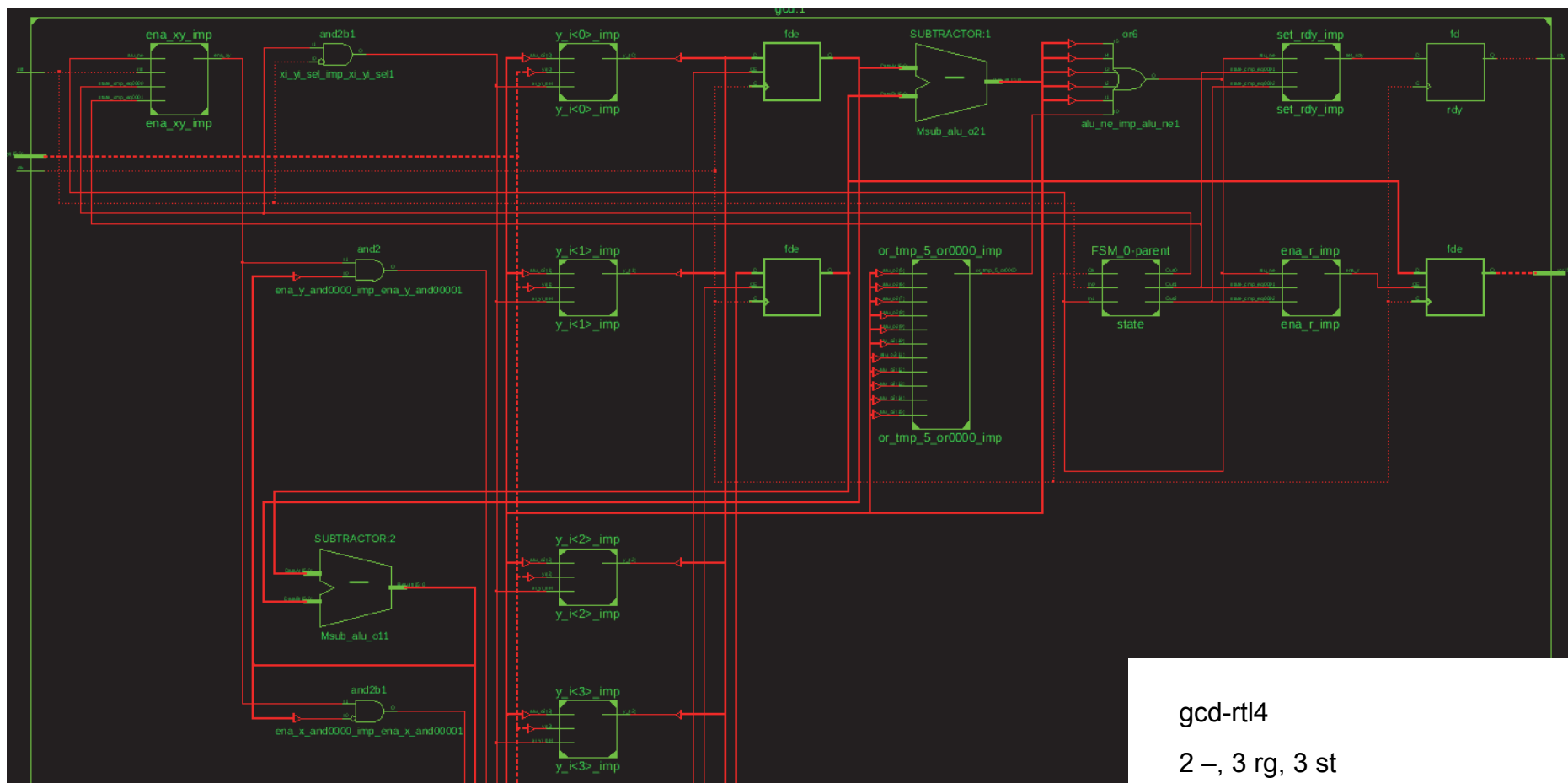
gcd-rtl3

1 –, 1 <, 3rg, 3st

1 takt/iteratsioon

58 SLC / 17.0 ns

GCD – kust tulevad need erinevused?



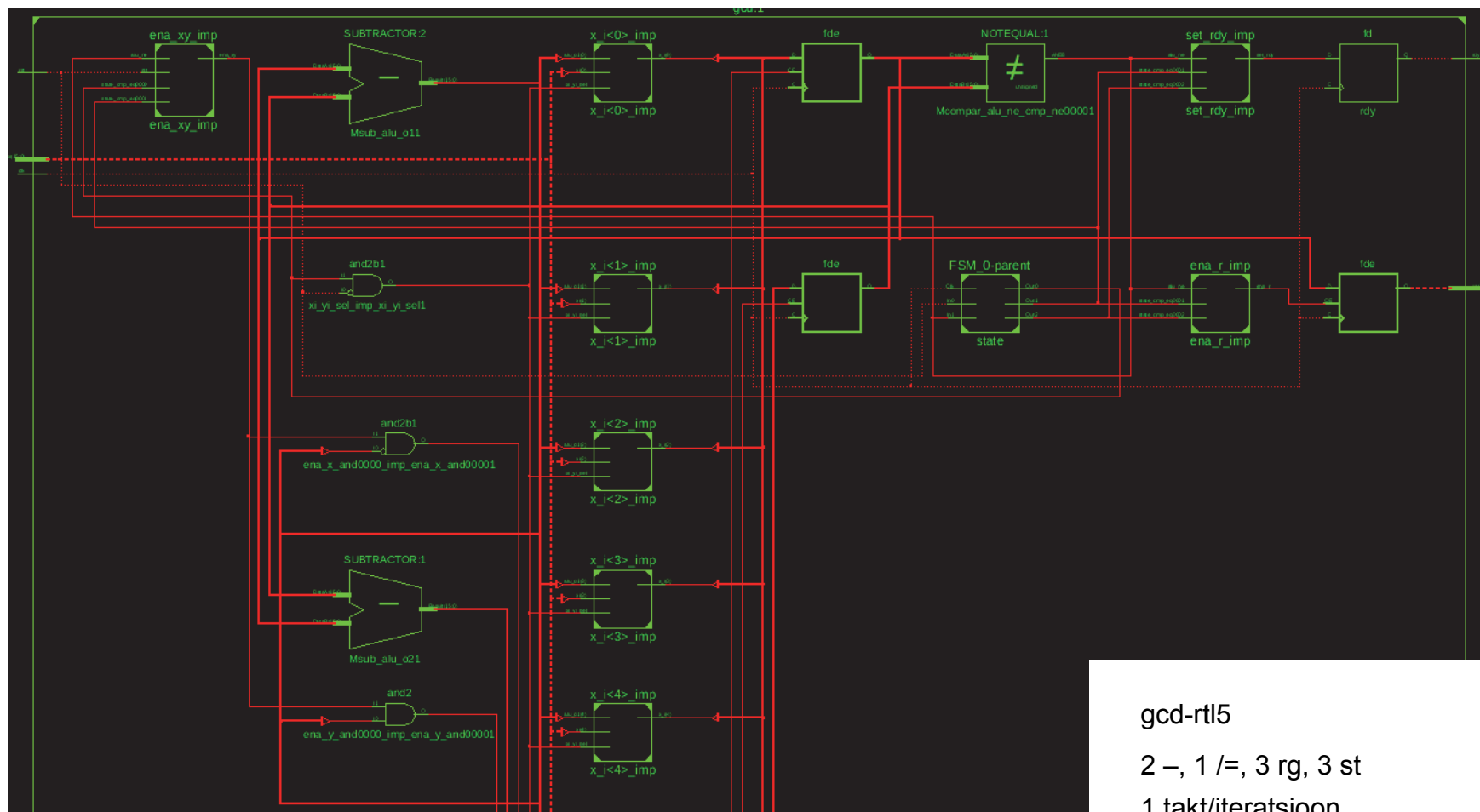
gcd-rtl4

2 –, 3 rg, 3 st

1 takt/iteratsioon

78 SLC / 12.6 ns

GCD – kust tulevad need erinevused?



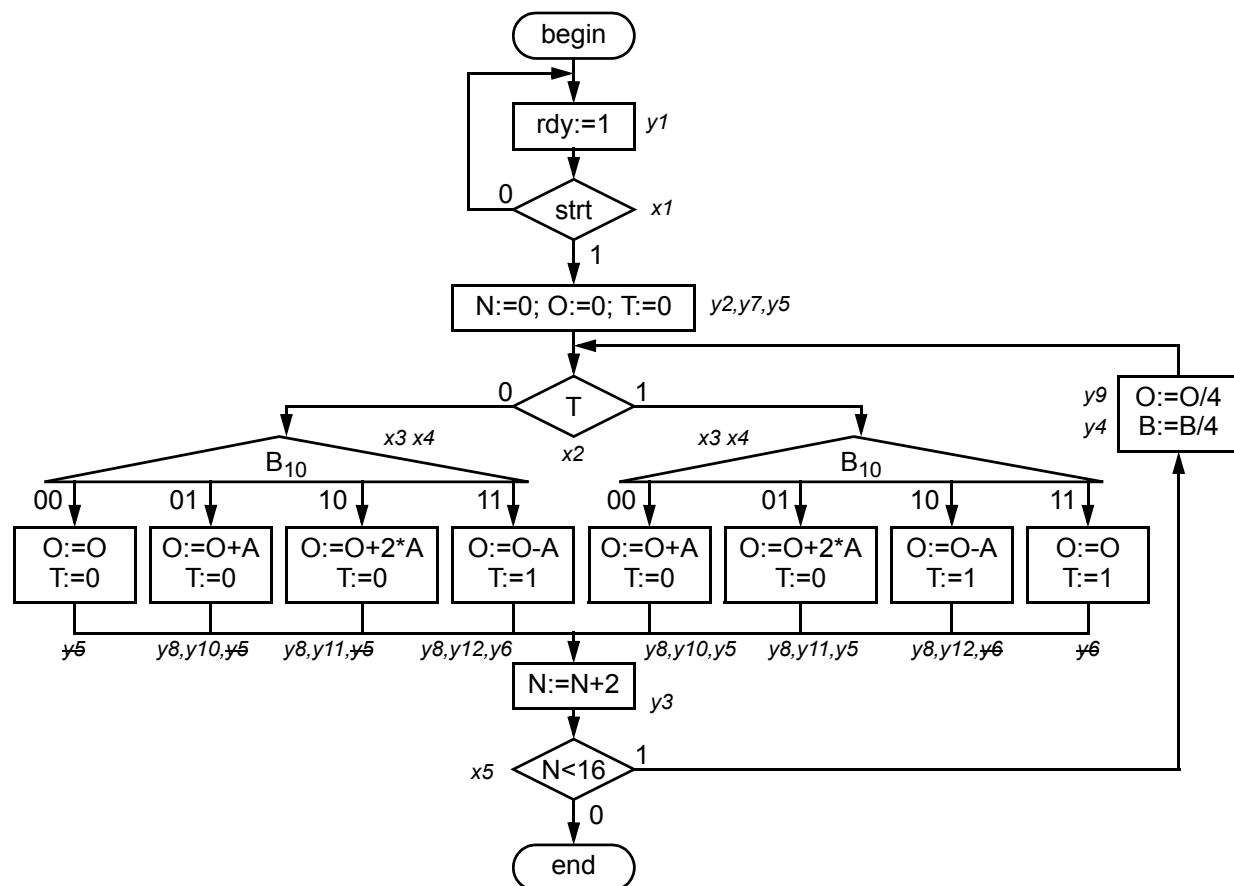
gcd-rtl5

2 –, 1 /=, 3 rg, 3 st

1 takt/iteratsioon

58 SLC / 8.0 ns

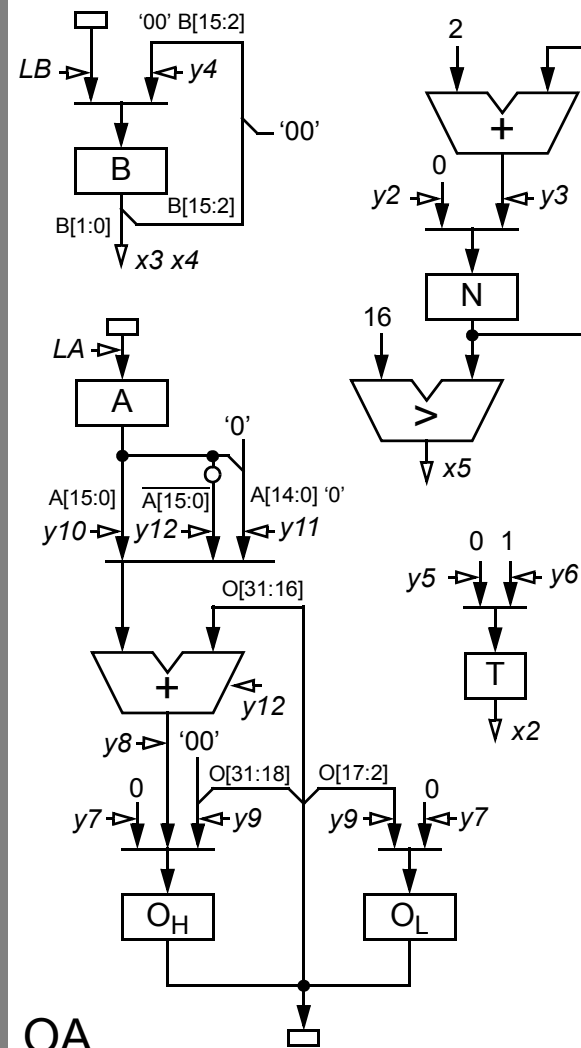
Realisatsiooni näide (ver. 1)



JA Moore: i 5, o 12, s 12, t 28; Mealy: i 5, o 12, s 4, t 13 (tagasiside OK)

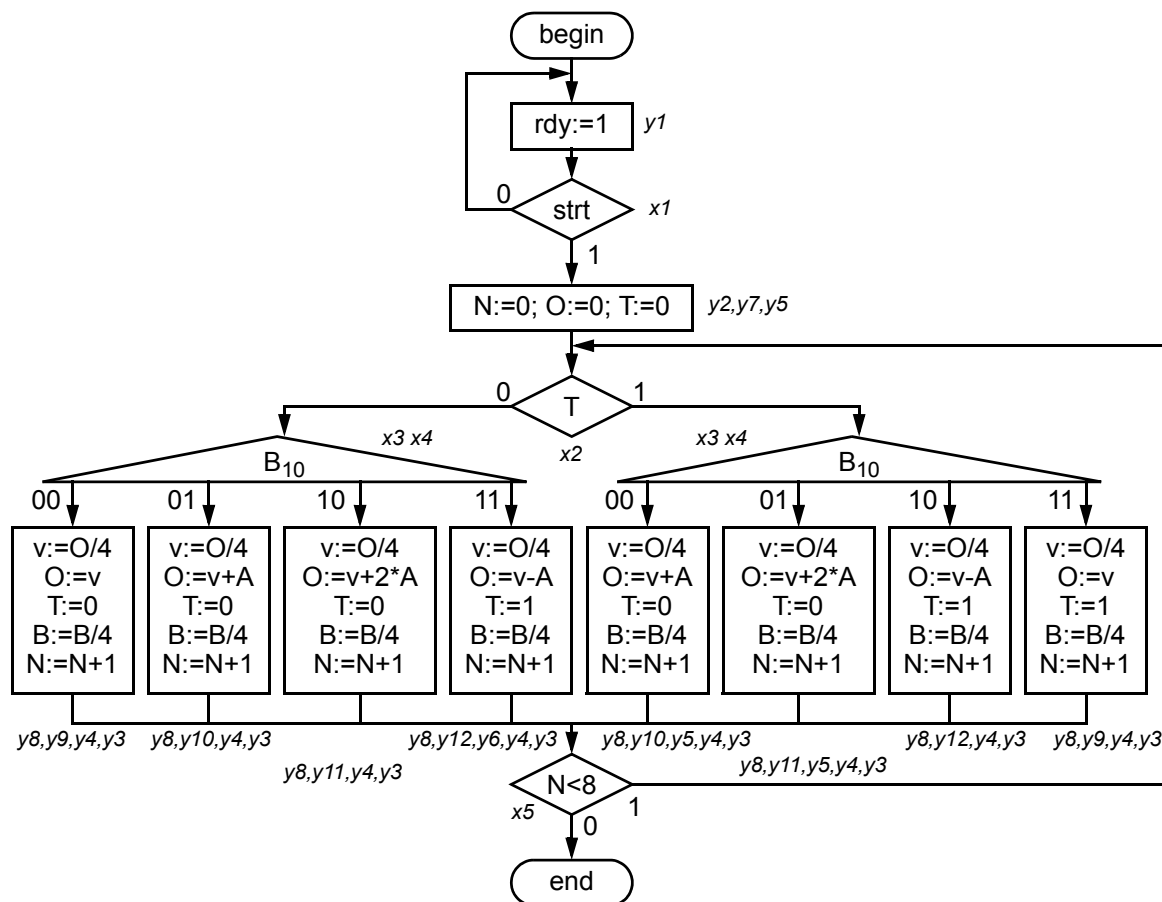
OA 2 sum., 1 võrdl., 5 (6) mux, 5 (6) reg.

3 takti iteratsiooni kohta, kokku 24 (+1) takti



Realisatsiooni näide (ver. 2)

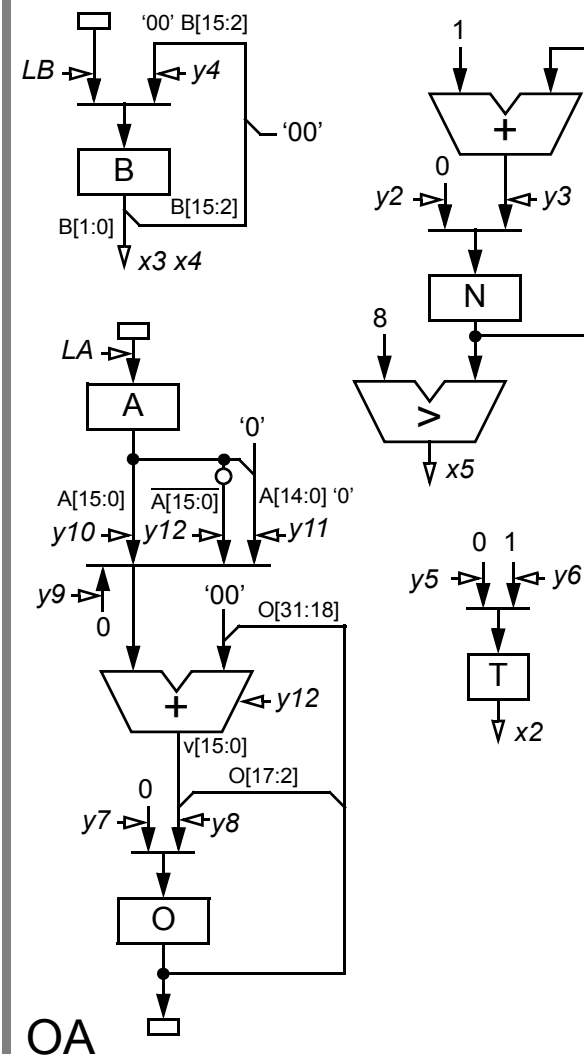
JA



JA Moore: i 5, o 12, s 10, t 82; Mealy: i 5, o 12, s 3, t 17

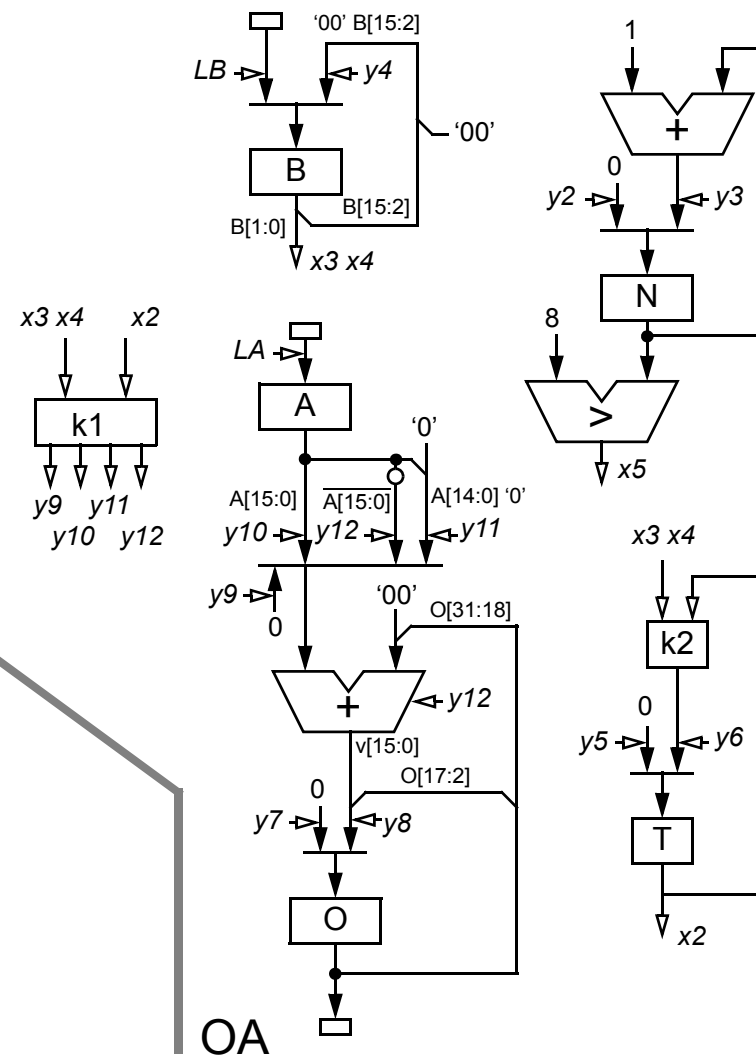
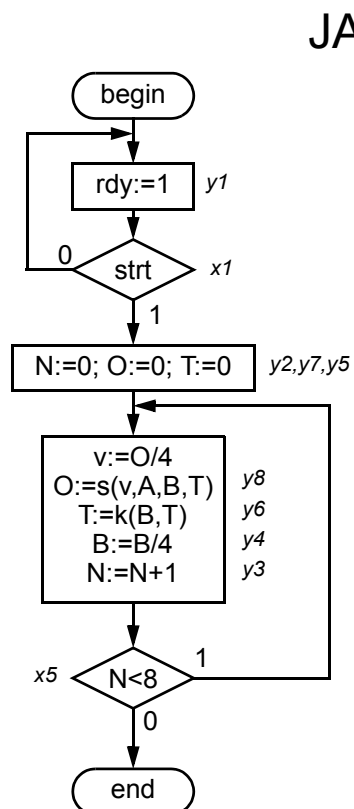
OA 2 sum., 1 vördl., 5 mux, 5 reg.

1 takt iteratsiooni kohta, kokku 9 (+1) takti



Realisatsiooni näide (ver. 3)

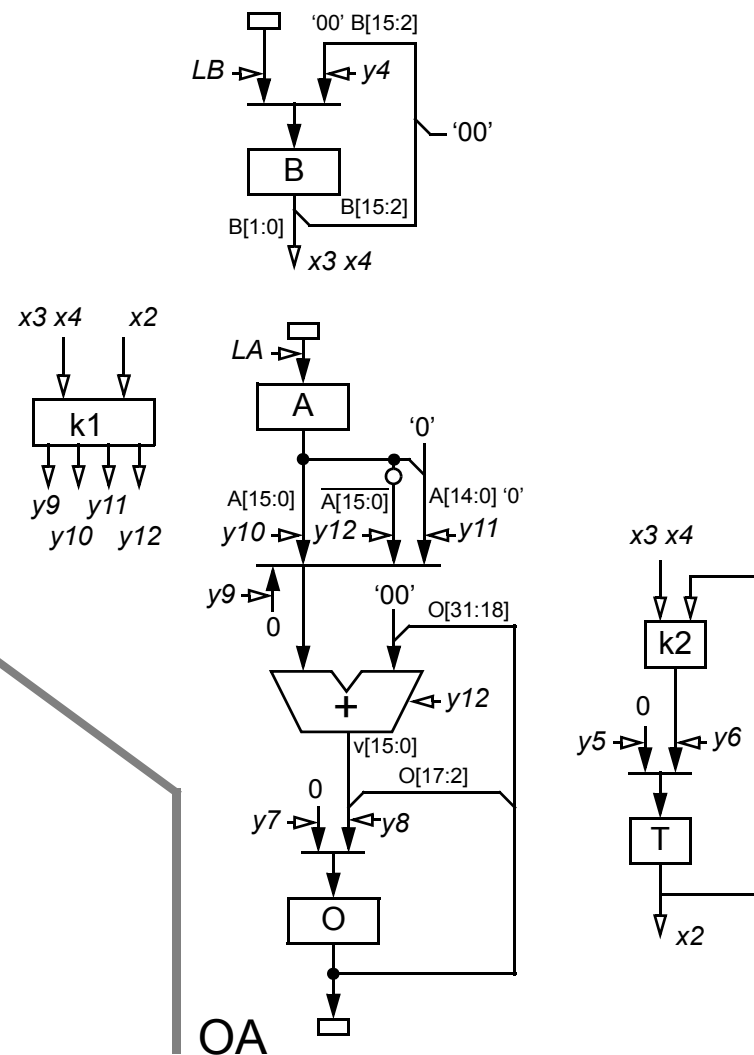
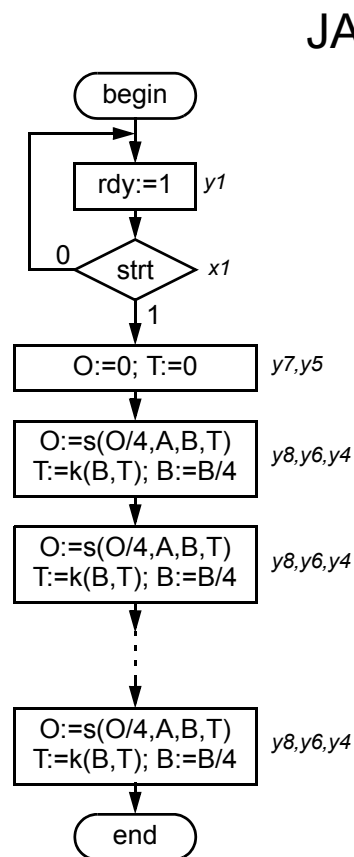
- Keerukus: JA $\rightarrow$ OA
- siirded $\sim$
kombinatsioonskeem



JA Moore: i 2, o 8, s 3, t 5; Mealy: i 2, o 8, s 3, t 5
 OA 2 sum., 1 võrdl., 5 mux, 5 reg., 1 (2) f-n
 1 takt iteratsiooni kohta, kokku 9 (+1) takti

Realisatsiooni näide (ver. 3)

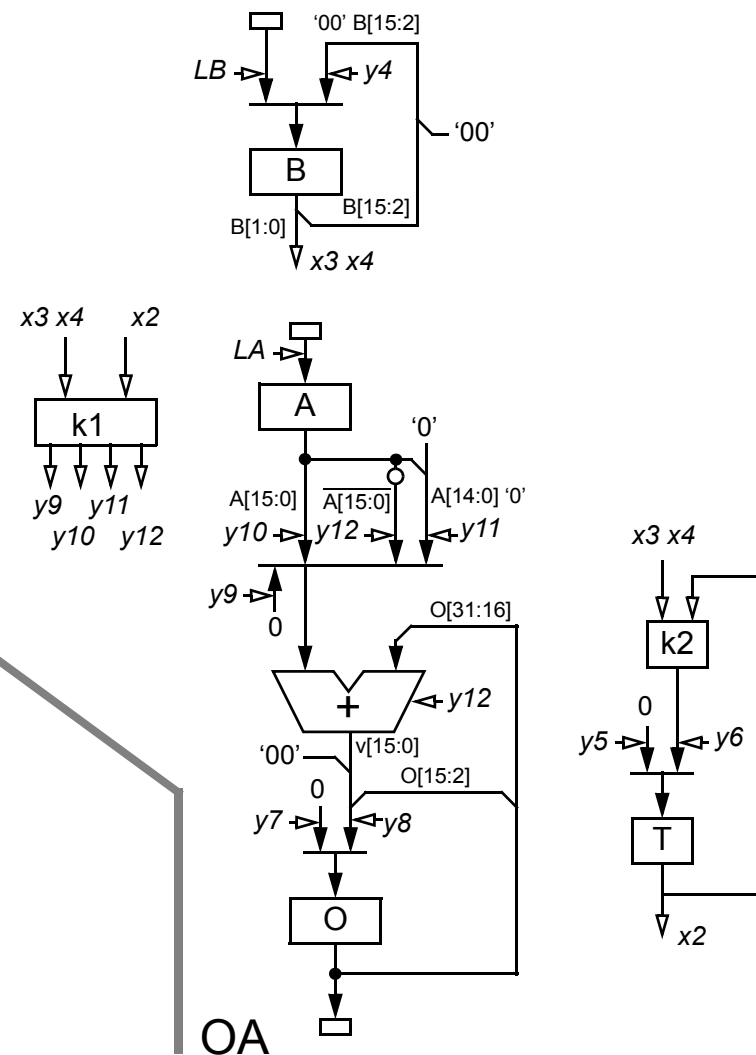
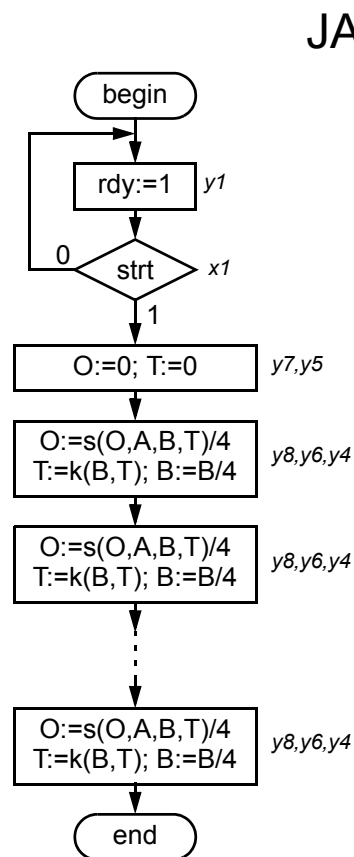
- Keerukus: OA → JA
- loendur ~ automaat
- Veelgi kiirem:
 - ühitada nullimine (y7,y5) ja esimene iteratsioon
 - OA: peaaegu sama
 - JA: 9 olekut
 - kiirus: 8 (+1) takti



JA Moore: i 1, o 6, s 10, t 11; Mealy: i 1, o 6, s 10, t 11
 OA 1 sum., 4 mux, 4 reg., 1 (2) f-n
 1 takt iteratsiooni kohta, kokku 9 (+1) takti

Realisatsiooni näide (ver. 4)

- **Eelmised variandidid**
 - 2 noorimat järku kasutamat
 - korrektse tulemuse asukoht?
- **Nihutamine pärast liitmist/lahutamist**
 - 2 vanimat järku alati 0-d!!!
- **Pisi-erinevused OA-s**



JA Moore: i 1, o 6, s 10, t 11; Mealy: i 1, o 6, s 10, t 11

OA 1 sum., 4 mux, 4 reg., 1 (2) f-n

1 takt iteratsiooni kohta, kokku 9 (+1) takti

Realisatsioonide võrdlused

- **Seitse erinevat korrutaja realisatsiooni**
 - **paralleelne, bitt korraga, 2 bitti korraga (versioonide 3 ja 4 erinevad VHDL koodid)**
 - **VHDL koodide erinevused põhjustatud sünteesi juhtimise vajadusest**
 - “what you write is what you get...”
 - **VHDL failid – <http://mini.pld.ttu.ee/~lrv/IAY0150/multiplier/>**

Tüüp	nr.	Synopsys DC		Xilinx ISE	
		comb. & reg. [e.g.]	[ns]	[slices]	[ns]
paralleelne	0	685 + 112 = 797	20.0	0 (1/24 mult.)	~10
radix-2 (bhv-rtl)	1	227 + 508 = 735	15.5	46 (7680)	6.7
radix-4 (v.4, bhv-rtl)	2	402 + 511 = 913	20.0	52 (7680)	7.8
radix-4 (v.4, ~90% rtl)	3	303 + 511 = 814	15.9	37 (7680)	7.1
radix-4 (v.4, 100% rtl)	4	179 + 511 = 690	19.9	34 (7680)	7.3
radix-4 (v.3, bhv-rtl)	5	397 + 514 = 911	20.0	51 (7680)	8.2
radix-4 (v.3, 100% rtl)	7	182 + 511 = 693	18.7	36 (7680)	6.4

Seos füüsilise maailmaga

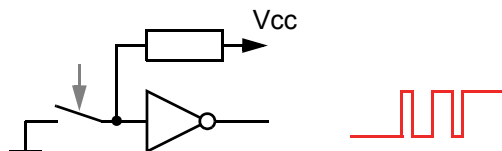
- Sisendid, väljundid jms.**

- vt. nt. John F. Wakerly, "Digital design: principles and practices." Pearson/Prentice Hall

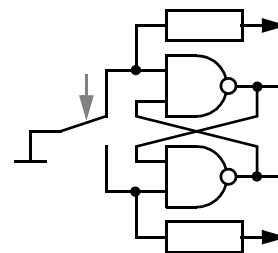
Sisendid

1) lülitid

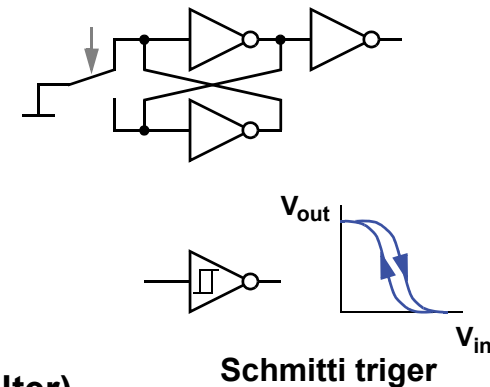
põhiprobleemiks kontaktide *värelemine*



võimalikud lahendused



... ja loomulikult ka RC-ahel (madalpääsfilter)

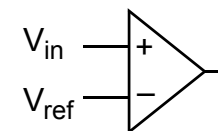


2) ADC – analoog-digitaal muundurid

pinge -> kood
vool -> kood
sagedus -> kood
jne.

põhi-elemendiks komparaator
võrreldakse erinevust sisend-
ja tugipinge vahel

paralleelne – n bitti -> 2^n komparaatorit
järjestikuline ~ FSM, DAC + komparaator

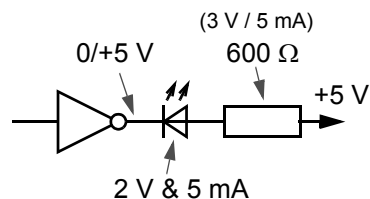


Seos füüsilise maailmaga

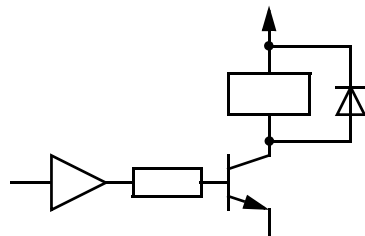
- Sisendid, väljundid jms.

Väljundid

1) indikaatorid (nt. LED)

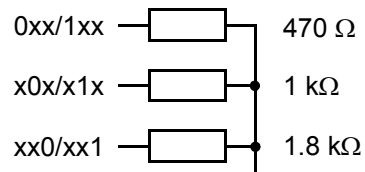


2) täiturid (nt. relee)



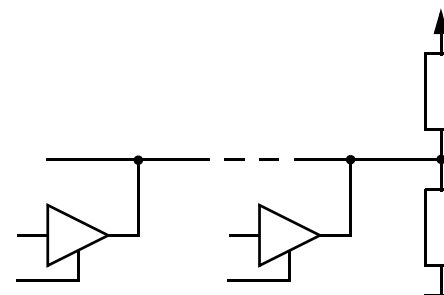
3) DAC – digitaal-analoog muundurid

nt. XSA-3S1000
www.xess.com

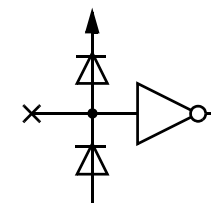


Siinid

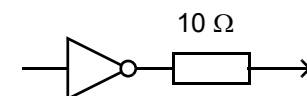
sobitus



kaitsediodid



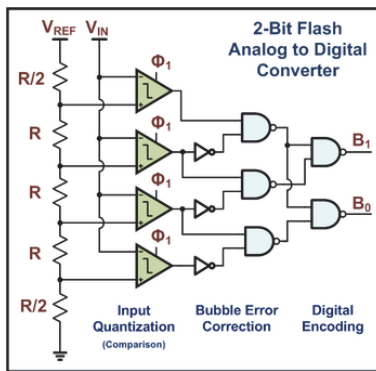
lühisekaitse



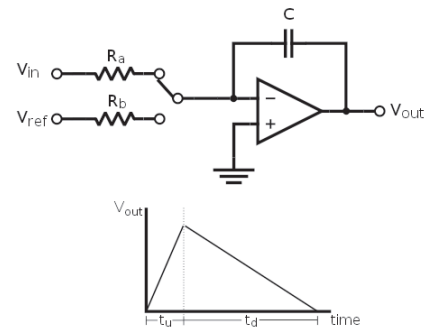
ADC / DAC

Analog-to-Digital-Converter / Digital-to-Analog-Converter

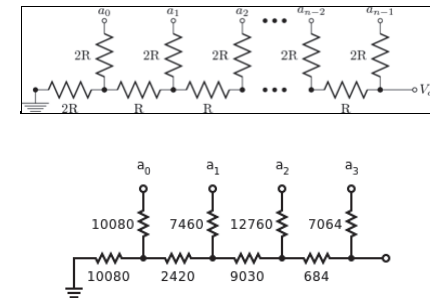
ADC



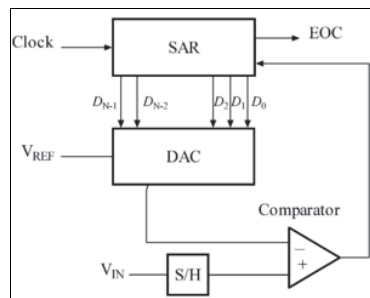
Integrating ADC



DAC



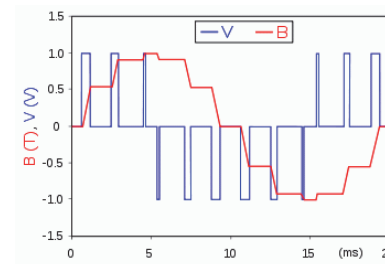
Resistor ladders



Successive approximation ADC

www.wikipedia.org

Pulse-width modulation



H-bridge

www.talkingelectronics.com

