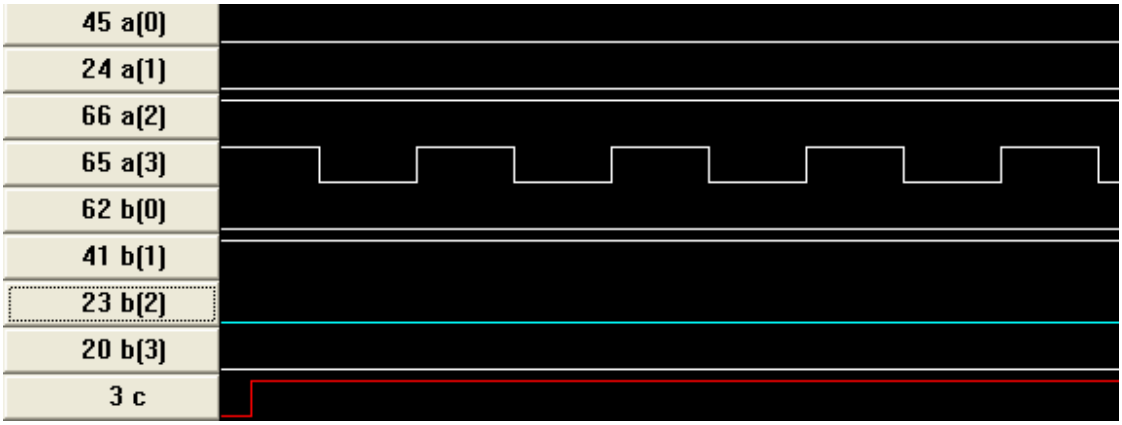
Opdracht 1:

```
--Opdracht 1

library ieee;
use ieee.std_logic_1164.all;
use work.std_arith.all;

entity ab is port(
 a,b : in std_logic_vector(3 downto 0);
 c : out std_logic);
end ab;

architecture main of ab is
begin
 process(a,b)
  begin
   if a > b then
    c <= '1';
   else
    c <= '0';
   end if;
 end process;
end main;
```



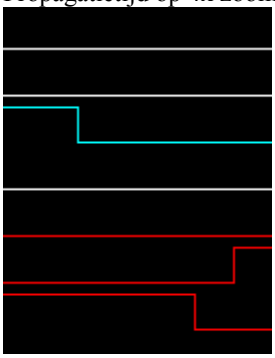
Gezien de simulatieplot is std_logic standaard unsigned. Als het namelijk signed was geweest dan had de klokpuls (sign bit a(3)) hem om en om negatief en dus kleiner dan signaal b gemaakt waardoor het resultaat niet constant hoog zou zijn.

Opdracht 2

```
--Opdracht 2

--4 bits
library ieee;
use ieee.std_logic_1164.all;
use work.numeric_std.all;
use work.std_arith.all;
-- use work.numeric_bit.all;

entity add is port(
 a,b : in  std_logic_vector(3 downto 0);
 c   : out std_logic_vector(4 downto 0)); -- plus carry
end add;
```

```
architecture main of add is
 begin
  process (a,b)
   begin
         c <= (("0" & a) + ("0" & b));
   end process;
end main;
```

Aantal gebruikte macrocellen: 4
Aantal unieke producttermen: 0
Propagatietijd op 4x zoom: ca. 2cm



--8 bits

```
entity add is port(
 a,b : in  std_logic_vector(7 downto 0);
 c   : out std_logic_vector(8 downto 0)); -- plus carry
end add;
```

Aantal unieke macrocellen: 5
Aantal unieke producttermen: 0
propagatietijd bij 4x zoom: ongeveer gelijk (2 cm)

--16 bits

```
entity add is port(
 a,b : in  std_logic_vector(15 downto 0);
 c   : out std_logic_vector(16 downto 0)); -- plus carry
end add;
```

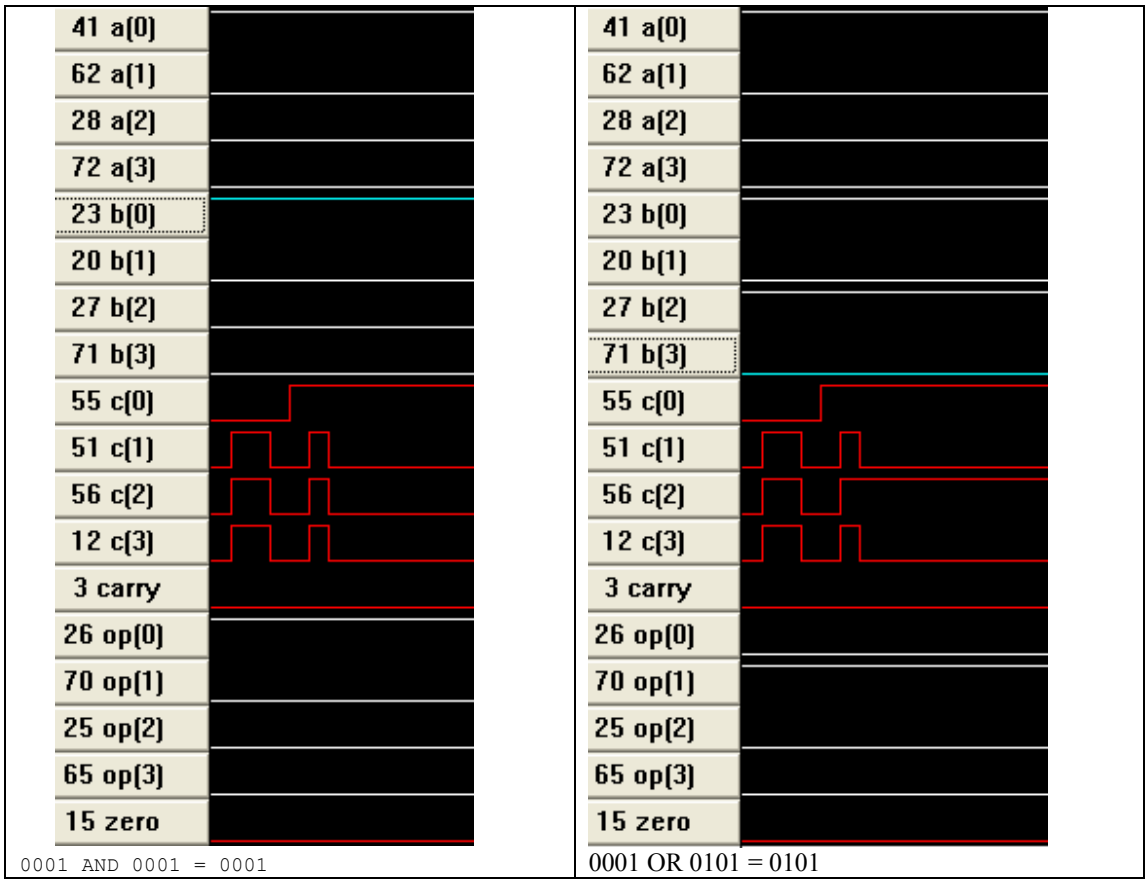
Aantal macrocellen: 12
Aantal unieke producttermen: 41
Propagatietijd: ca 5,5 cm (ongeveer 3 keer zo lang).

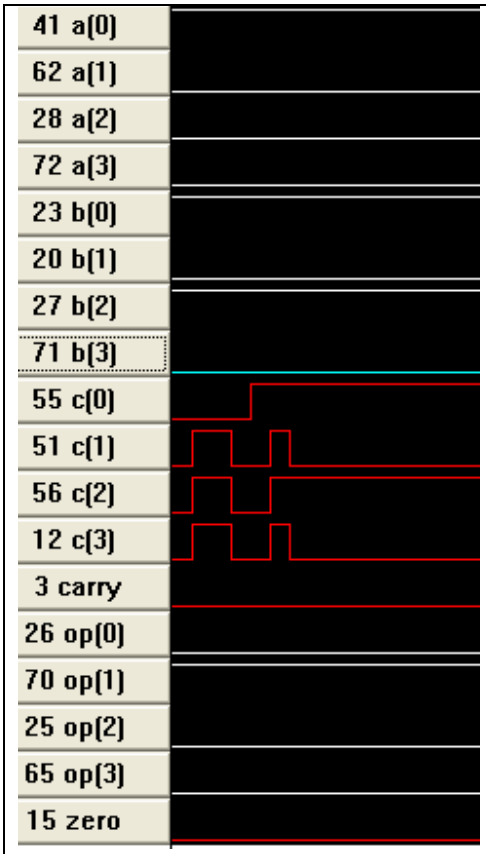
Conclusie: De propagatietijd is in een grote mate afhankelijk van het aantal producttermen.

Opdracht 3

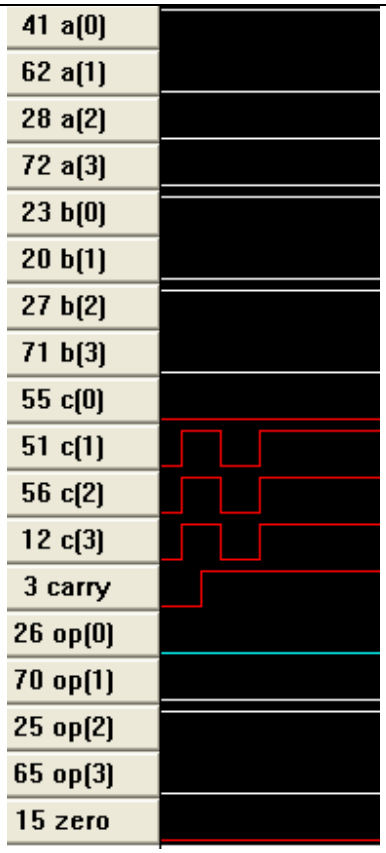
```
-- Opdracht 3 | 4-bits ALU
-- 21 oktober 05:34:32
-- Marinus van Heuvelen | Roderik Emmerink
-- ELVN4

-- coding specification:
        --0001 -> and
        --0010 -> or
        --0011 -> xor
        --0100 -> not
        --0101 -> +
        --0110 -> -
        --0111 -> *
        --1000 -> shl
        --1001 -> shr

-- Begin of Listing
library ieee;
use ieee.std_logic_1164.all;
use work.std_arith.all;
use work.numeric_std.all;

--   4 bits ALU
--  ____ ____
--  \ a \ / b /
-- c-\       /
-- z- \__c_ /
--

entity alu is port(
 a,b,op : in unsigned(3 downto 0); -- input signals
 c : out unsigned(3 downto 0);     -- alu output
 zero, carry : out std_logic);     -- flags
end alu;

architecture Main of alu is
 signal a1, b1, c1: unsigned(4 downto 0); -- 1 bitje meer
 signal c10 : unsigned(9 downto 0);
begin
 process (a,b,op,a1,b1,c1,c10)
  begin
   a1 <= ('0', a(3), a(2), a(1), a(0)); -- bitje toevoegen.
   b1 <= ('0', b(3), b(2), b(1), b(0));
   case op is
    when "0001" => c1 <= a1 AND b1;     -- and
    when "0010" => c1 <= a1 OR b1;      -- or
    when "0011" => c1 <= a1 XOR b1;     -- xor
    when "0100" => c1 <= NOT a1;        -- not
```

```
    when "0101" => c1 <= a1 + b1;        -- +
    when "0110" => c1 <= a1 - b1;        -- -
    when "0111" => c10 <= a1 * b1;       -- *
    when "1000" => c1 <= a1 sll 1;        -- shl
    when "1001" => c1 <= a1 srl 1;        -- shr
    when others => c <= "0000";
   end case;
  zero <= '0';
  case op is
   when "0111" => c <= c10(3 downto 0);
   when others => c <= c1(3 downto 0); -- vlaggetje niet meenemen.
   end case;
   carry <= c1(4);       -- 4de bitje is het vlaggetje
  end process;
end Main;
```

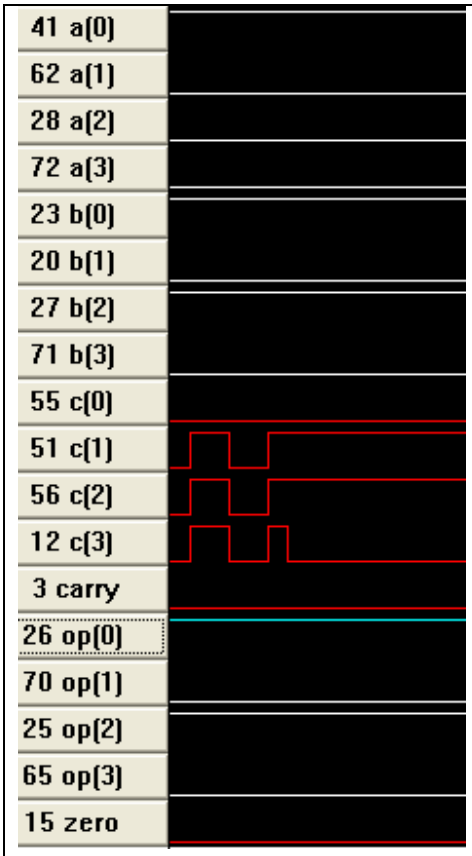




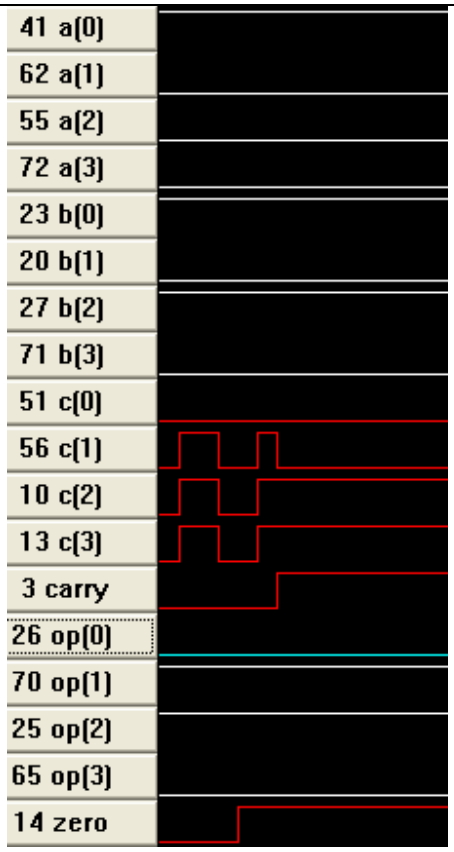
0001 OR 0101 = 0101



NOT 0001 = 1110

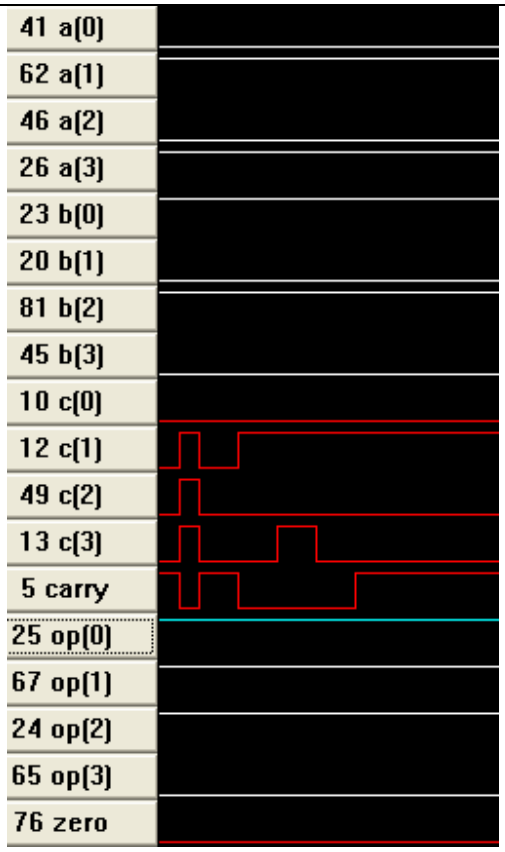


0001 + 0101 = 0110

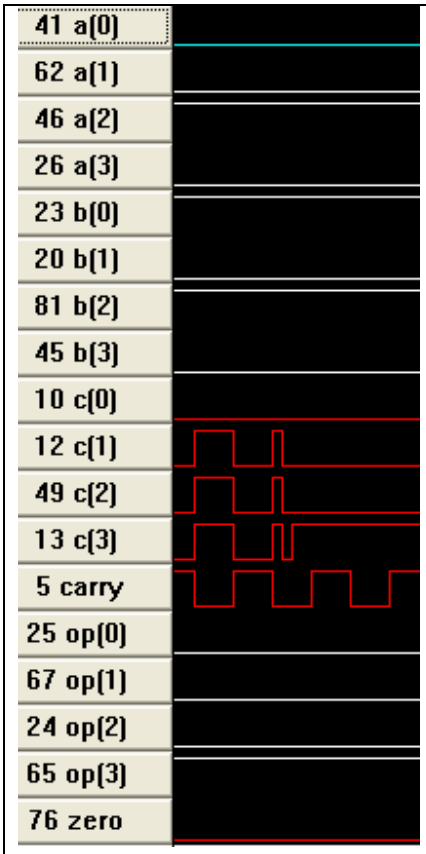


0001 – 0101 = Carry hoog, zero hoog

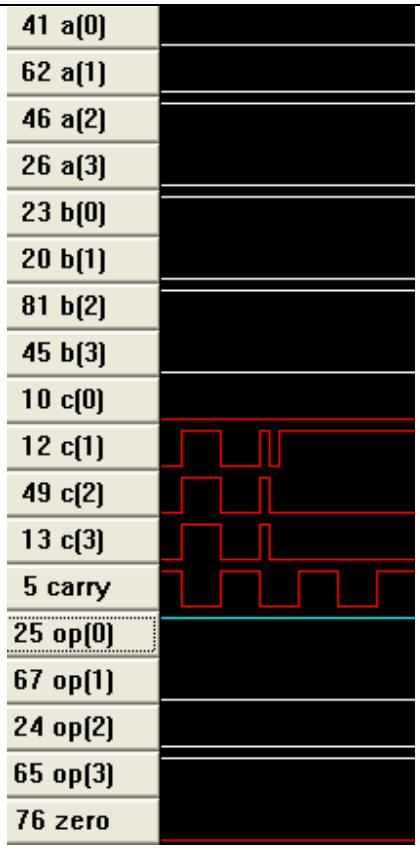
0010 * 0101 = 1010



1010 * 0101 = Carry out!



SHL 0100 = 1000 (da's logisch)



SHR 0100 = 0010

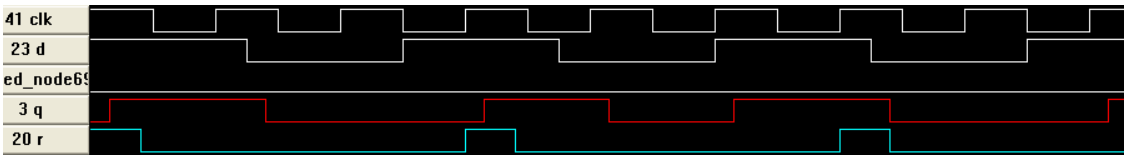
Opdracht 4

D-latch met asynchrone overheersende reset:

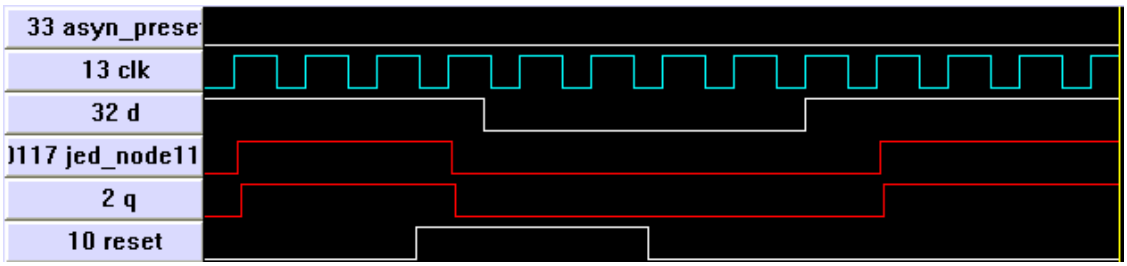
```
-- Week4 Opdracht 4a
-- D-latch met een asynchrone, overheersende reset
-- Marinus van Heuvelen & Roderik Emmerink
-- 10 december 2003 20:03
-- ELVN4

library ieee;
use ieee.std_logic_1164.all;

entity dlatch is port (
d, r, clk : in std_logic;
q : out std_logic);
end dlatch;

architecture main of dlatch is
begin
 process (clk,r,d) begin
  if (r = '1') then
        q <= '0';
  end if;
  if (clk = '1') then
    q <= d;
```

```
  end if;
 end process;

end main;
```

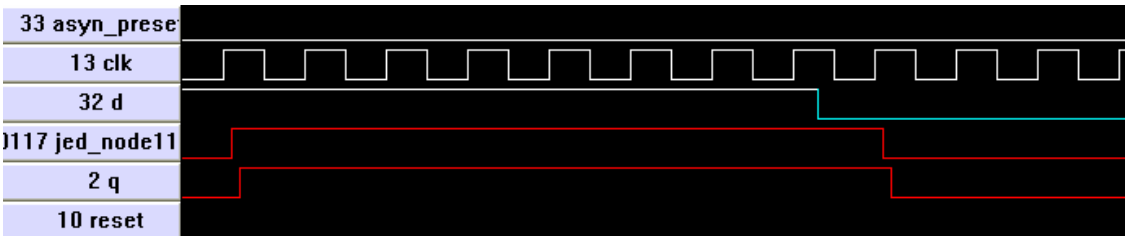
```
--
```



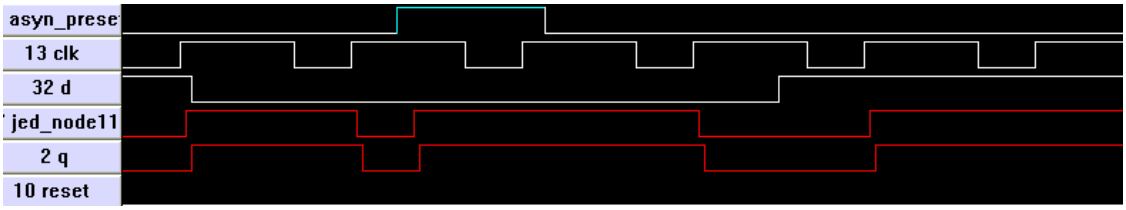

```
--D-flipflop met asyn preset en syn reset
library ieee;
use ieee.std_logic_1164.all;

entity eDflip is port(asyn_preset,reset,d,clk: in  std_logic; q:out std_logic);
end eDflip;
architecture aDflip of eDflip is
begin
process (clk, asyn_preset) begin
if asyn_preset = '1' then
                        q <= '1';
                elsif clk'event and clk = '1' then
                        if reset = '1' then
                                q <= '0';
                        else
                                q <= d;
                        end if;
                end if;
        end process;
end aDflip;
.
```



Figuur: functioneren van synchrone reset



Figuur: Doorgifte van data (zonder reset, zonder preset)



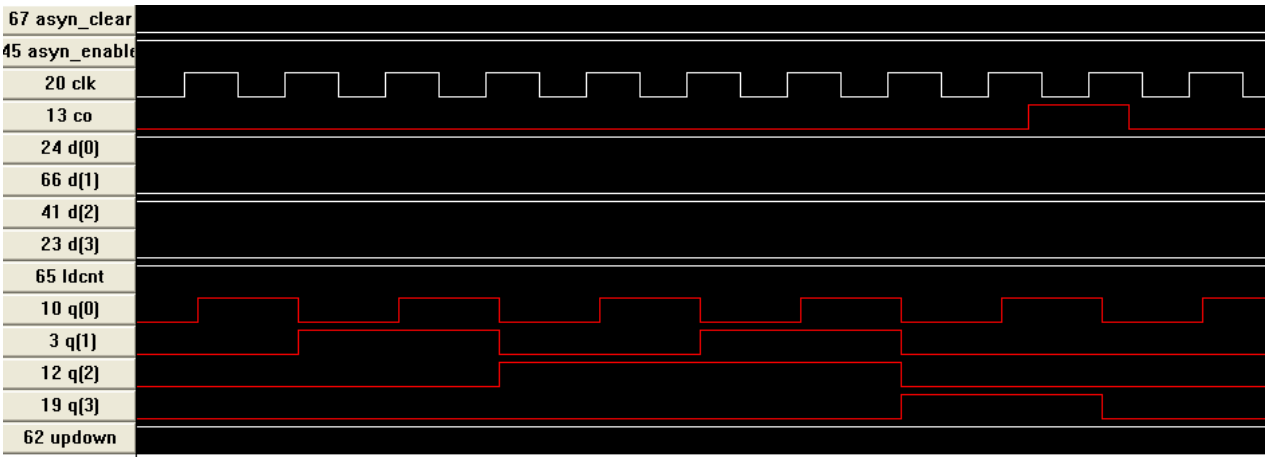
Figuur: Functioneren van de asynchrone preset.

Opdracht 5

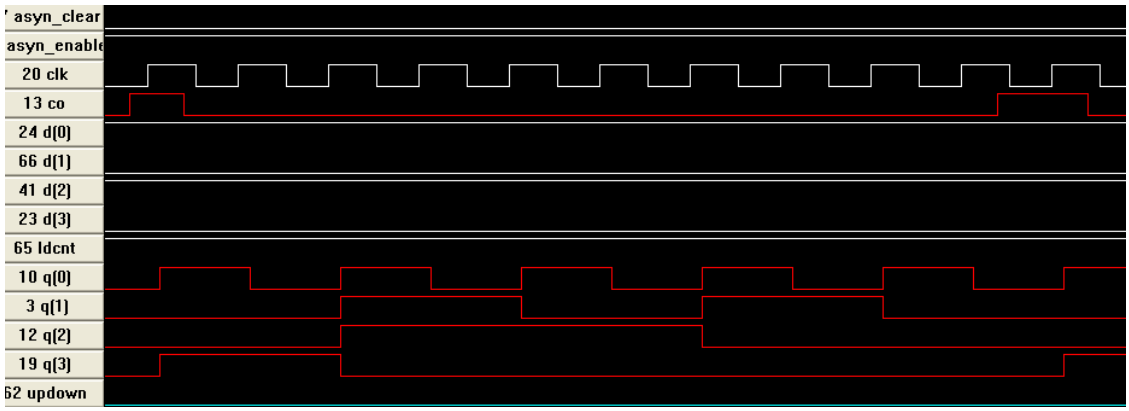
```
-- Week 3, opdracht 5 BCD teller
-- Roderik Emmerink, Marinus van Heuvelen
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity bcd_count_e is
        port (
                clk    : in  std_logic;              -- kloksignaal
                asyn_clear  : in  std_logic;         -- asynchrone clear
                asyn_enable : in  std_logic;         -- enable signaal voor load en count
                updown : in  std_logic;              -- 1 -> optellen, 0 -> aftellen
                ldcnt  : in  std_logic;              -- 1 -> tellen, 0 -> laden van d
                d      : in  unsigned(3 downto 0);  -- data-load ingangen
                q      : out unsigned(3 downto 0);  -- telleruitgangen
                co     : out std_logic        -- carry out, 1 als:
                                                                -- teller 9 en
optellen of
                                                                -- teller 0 en
aftellen.
        );
end bcd_count_e;

architecture gedrag of bcd_count_e is
begin
        process(clk,asyn_clear,ldcnt,asyn_enable)
        begin
                if asyn_clear = '1' then -- clear, dan alle output 0
                        q <= (others =>'0');
                elsif clk'event and clk='1' then -- clock flank
                   if asyn_enable = '1' then     -- outputs enabled
                        if ldcnt = '1' then
                                if q = "1001" and updown = '1' then -- 9 naar 0
                                        q <= (others =>'0');
                                elsif q = "0000" and updown = '0' then -- 0 naar 9
                                        q <= "1001";
                                else
                                        if updown = '1' then
                                                q <= q + 1; -- optellen
                                        else
                                                q <= q - 1; -- aftellen
                                        end if;
                                end if;
                        else
                                q <= d;
                        end if;
                   end if;
                end if;
        end process;
        -- check carry out
        co <= '1' when q = "1001" and asyn_enable='1' and updown = '1' else
                  '1' when q = "0000" and asyn_enable='1' and updown = '0' else
```

```
                '0';
end gedrag;
```

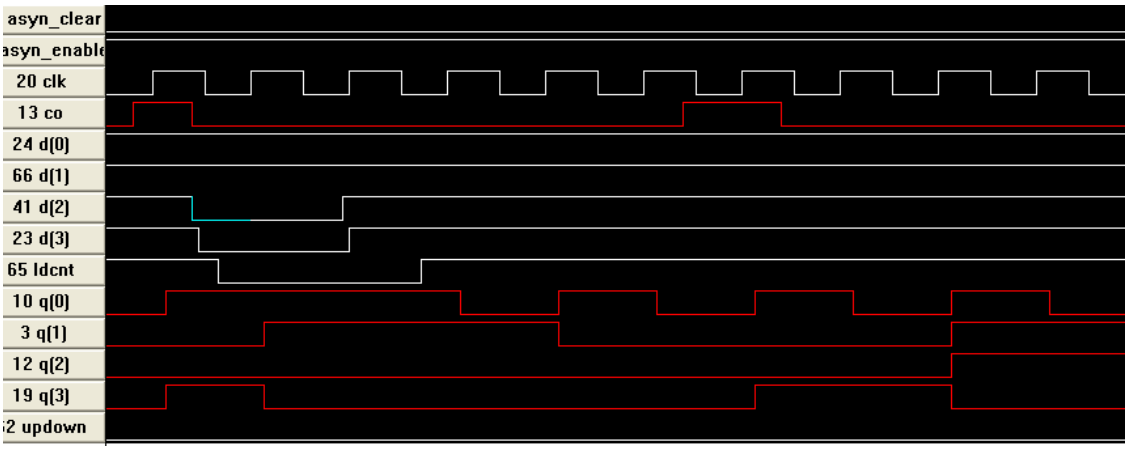
Figuur: code 4 bits bcd counter.



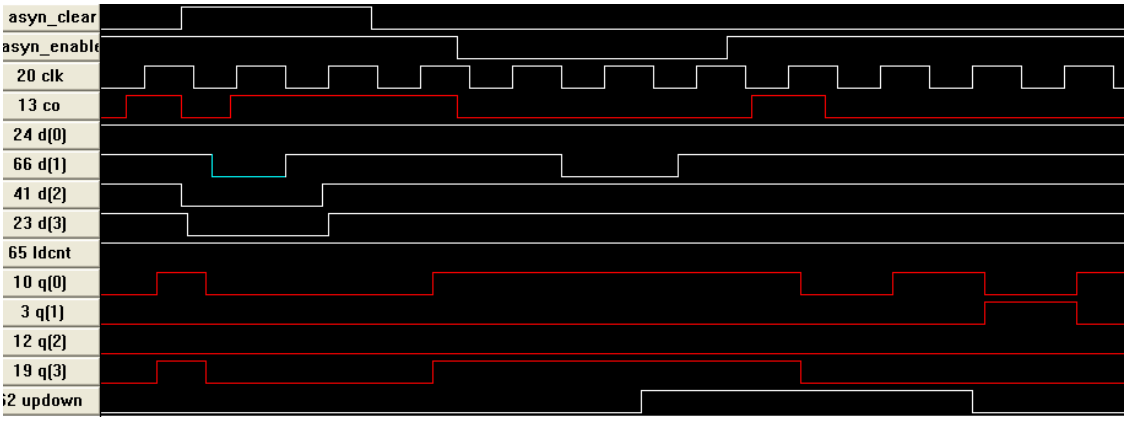
Figuur: optellen



Figuur: aftellen



Figuur: aftellen en de teller laden met 3.



Figuur: demonstratie clear en enable bij wisselende omstandigheden van updown, ldcnt en data.

Package bestand:

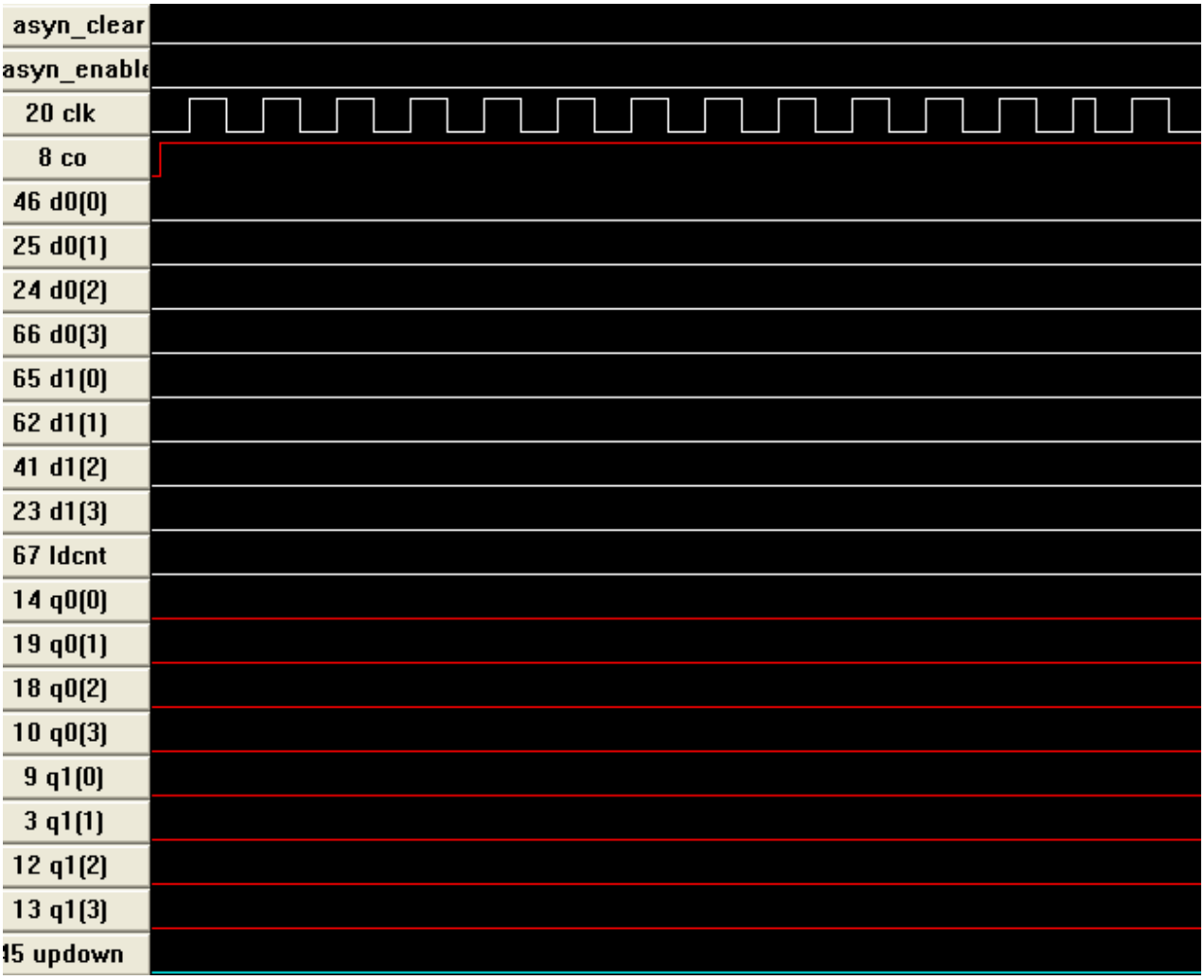
```
-- Week 3, opdracht 5 2 BCD teller gecascadeerd
-- Roderik Emmerink, Marinus van Heuvelen
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity bcd_count_e is
        port (
                clk    : in  std_logic;              -- kloksignaal
                asyn_clear  : in  std_logic;         -- asynchrone clear
                asyn_enable : in  std_logic;         -- enable signaal voor load en count
                updown : in  std_logic;              -- 1 -> optellen, 0 -> aftellen
                ldcnt  : in  std_logic;              -- 1 -> tellen, 0 -> laden van d
                d      : in  unsigned(3 downto 0);  -- data-load ingangen
                q      : out unsigned(3 downto 0);  -- telleruitgangen
                co     : out std_logic        -- carry out, 1 als:
                                                                -- teller 9 en
optellen of
                                                                -- teller 0 en
aftellen.
        );
end bcd_count_e;

architecture gedrag of bcd_count_e is
begin
        process(clk,asyn_clear,ldcnt,asyn_enable)
        begin
                if asyn_clear = '1' then -- clear, dan alle output 0
                        q <= (others =>'0');
                elsif clk'event and clk='1' then -- clock flank
                   if asyn_enable = '1' then     -- outputs enabled
                        if ldcnt = '1' then
                                if q = "1001" and updown = '1' then -- 9 naar 0
                                        q <= (others =>'0');
                                elsif q = "0000" and updown = '0' then -- 0 naar 9
                                        q <= "1001";
                                else
                                        if updown = '1' then
                                                q <= q + 1; -- optellen
                                        else
                                                q <= q - 1; -- aftellen
                                        end if;
                                end if;
                        else
                                q <= d;
                        end if;
```

```
                end if;
            end if;
        end process;
        -- check carry out
        co <= '1' when q = "1001" and asyn_enable='1' and updown = '1' else
              '1' when q = "0000" and asyn_enable='1' and updown = '0' else
              '0';
end gedrag;
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
package bcd_count_pkg is
        component bcd_count_e port(
                clk    : in  std_logic;              -- kloksignaal
                asyn_clear  : in  std_logic;         -- asynchrone clear
                asyn_enable : in  std_logic;         -- enable signaal voor load en count
                updown : in  std_logic;              -- 1 -> optellen, 0 -> aftellen
                ldcnt  : in  std_logic;              -- 1 -> tellen, 0 -> laden van d
                d      : in  unsigned(3 downto 0);  -- data-load ingangen
                q      : out unsigned(3 downto 0);  -- telleruitgangen
                co     : out std_logic        -- carry out
        );
        end component;
end bcd_count_pkg;
```

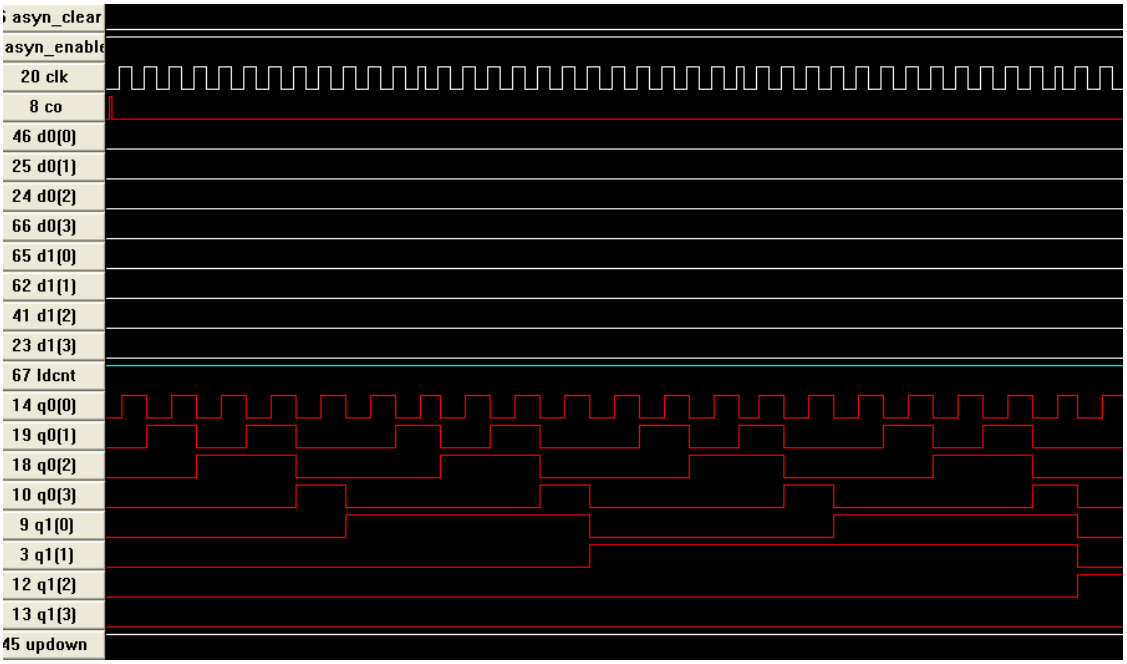
cascade bestand:

```
library ieee;
use work.std_arith.all;
use work.numeric_std.all;
use ieee.std_logic_1164.all;
use work.bcd_count_pkg.all;

entity e_bcd_2seg is port(
        asyn_clear, asyn_enable, updown, ldcnt, clk:        in std_logic;
        q0,q1:                                  out    unsigned(3 downto 0);
        d0,d1:                                  in unsigned(3 downto 0);
        co:                                             out    std_logic);
end e_bcd_2seg;

architecture a_bcd_2seg of e_bcd_2seg is
signal co0, asyn_enable0:     std_logic;
begin
        u0: bcd_count_e port map (clk,asyn_clear,asyn_enable,updown,ldcnt,d0,q0,co0);
        asyn_enable0 <= co0 or (not ldcnt);
        u1: bcd_count_e port map (clk,asyn_clear,asyn_enable0,updown,ldcnt,d1,q1,co);
end a_bcd_2seg;
```

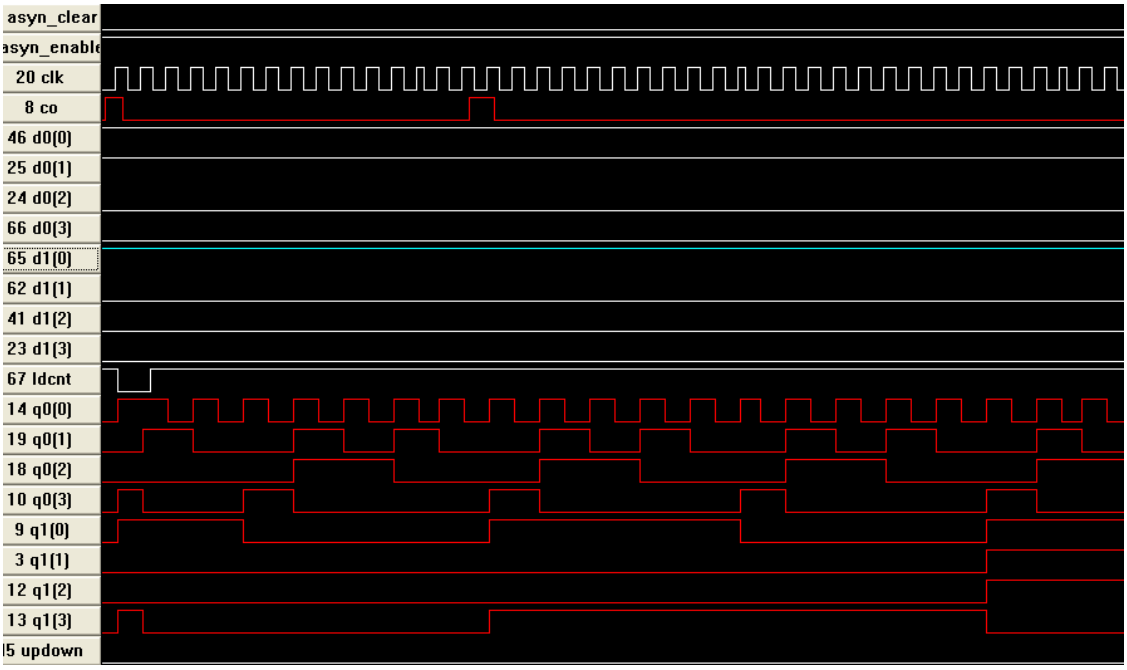
Simulaties van de gecascadeerde BCD counter:



Figuur: kloksignaal aangeboden, alle input is laag. Carry out wordt hoog omdat de counter in aftelmode staat, maar het telt niet omdat het niet enabled is en omdat het in laadmode staat.



Figuur: teller is enabled, staat in telmode en telt omhoog. Als het eerste segment van 0 tot 9 heeft geteld hoogt het via haar enable het tweede segment één op.



Figuur: Ook aftellen functioneert naar behoren. De teller laadt het getal 13. Van 10 gaat het over op 09, zoals het hoort en bij 0 aangekomen continueert het naar 99. Bovendien wordt zoals verwacht de carry out dan even hoog gemaakt.