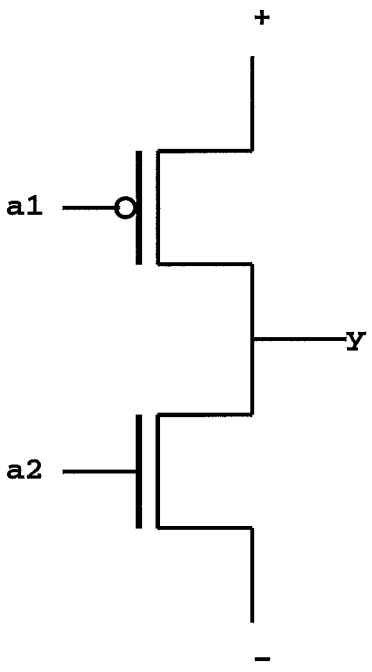
'Standaard' VHDL types:

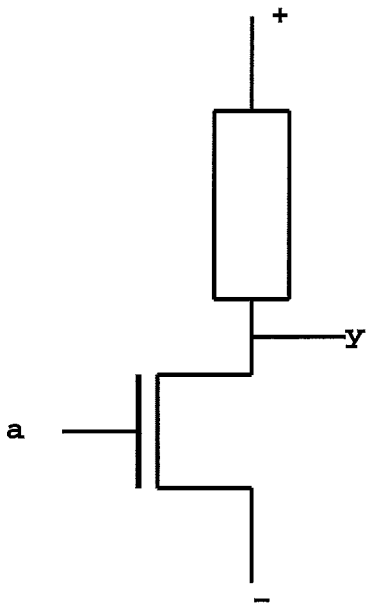
- .integer
- .boolean
- .time
- .real
- .bit
- .bit_vector

Vraag:

Kunnen hiermee alle logische schakelingen worden gemodelleerd ?

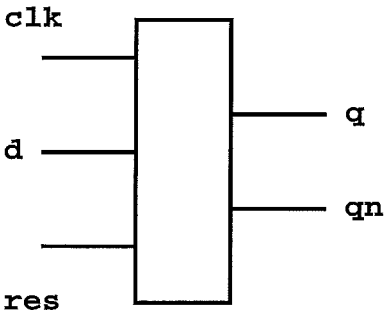


a1	a2	y
0	0	1
0	1	?
1	0	?
1	1	0



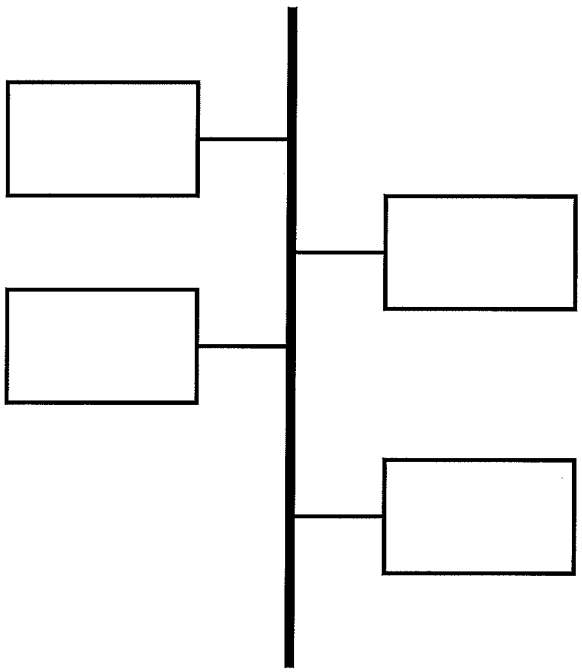
a	y
0	?
1	0

d_flipflop



Wat is de waarde van q en qn direct na het inschakelen ?

bus_structuur



meerdere sources op een signaal

Nieuw type met 9-waardige logica: std_ulogic

'U' -- uninitialized

'X' -- Forcing unknown

'0' -- Forcing 0

'1' -- Forcing 1

'Z' -- High impedance

'W' -- Weak unknown

'L' -- Weak 0

'H' -- Weak 1

'-' -- Don't care

subtype: std_logic is resolved std_ulogic

type: std_ulogic_vector

type: std_logic_vector

gedefinieerd in het package std_logic_1164

in de IEEE library

```
-------------------------------------------------------------------
-- resolution function
-------------------------------------------------------------------
CONSTANT resolution_table : stdlogic_table := (
--      ---------------------------------------------------------
--      |  U    X    0    1    Z    W    L    H    -        |    |
--      ---------------------------------------------------------
        ( 'U', 'U', 'U', 'U', 'U', 'U', 'U', 'U', 'U' ), -- | U |
        ( 'U', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X' ), -- | X |
        ( 'U', 'X', '0', 'X', '0', '0', '0', '0', 'X' ), -- | 0 |
        ( 'U', 'X', 'X', '1', '1', '1', '1', '1', 'X' ), -- | 1 |
        ( 'U', 'X', '0', '1', 'Z', 'W', 'L', 'H', 'X' ), -- | Z |
        ( 'U', 'X', '0', '1', 'W', 'W', 'W', 'W', 'X' ), -- | W |
        ( 'U', 'X', '0', '1', 'L', 'W', 'L', 'W', 'X' ), -- | L |
        ( 'U', 'X', '0', '1', 'H', 'W', 'W', 'H', 'X' ), -- | H |
        ( 'U', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X' )  -- | - |
    );

FUNCTION resolved ( s : std_ulogic_vector ) RETURN std_ulogic IS
    VARIABLE result : std_ulogic := 'Z';  -- weakest state default
BEGIN
    -- the test for a single driver is essential otherwise the
    -- loop would return 'X' for a single driver of '-' and that
    -- would conflict with the value of a single driver unresolved
    -- signal.
    IF    (s'LENGTH = 1) THEN    RETURN s(s'LOW);
    ELSE
        FOR i IN s'RANGE LOOP
            result := resolution_table(result, s(i));
        END LOOP;
    END IF;
    RETURN result;
END resolved;
```

Nieuwe types moeten ook aangepaste operatoren hebben

(operator Overloading)

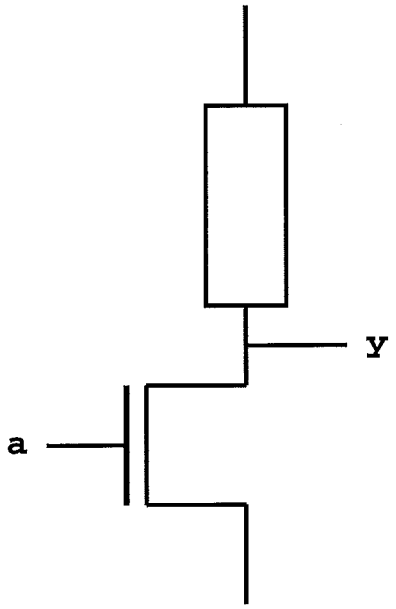
Bijv. de or_operator

```
CONSTANT or_table : stdlogic_table := (
--      ---------------------------------------------------
--      |  U    X    0    1    Z    W    L    H    -      |  |
--      ---------------------------------------------------
        ( 'U', 'U', 'U', '1', 'U', 'U', 'U', '1', 'U' ),  -- | U |
        ( 'U', 'X', 'X', '1', 'X', 'X', 'X', '1', 'X' ),  -- | X |
        ( 'U', 'X', '0', '1', 'X', 'X', '0', '1', 'X' ),  -- | 0 |
        ( '1', '1', '1', '1', '1', '1', '1', '1', '1' ),  -- | 1 |
        ( 'U', 'X', 'X', '1', 'X', 'X', 'X', '1', 'X' ),  -- | Z |
        ( 'U', 'X', 'X', '1', 'X', 'X', 'X', '1', 'X' ),  -- | W |
        ( 'U', 'X', '0', '1', 'X', 'X', '0', '1', 'X' ),  -- | L |
        ( '1', '1', '1', '1', '1', '1', '1', '1', '1' ),  -- | H |
        ( 'U', 'X', 'X', '1', 'X', 'X', 'X', '1', 'X' )   -- | - |
);

FUNCTION "or"   ( l : std_ulogic; r : std_ulogic ) RETURN UX01 IS
BEGIN
    RETURN (or_table(l, r));
END "or";
```

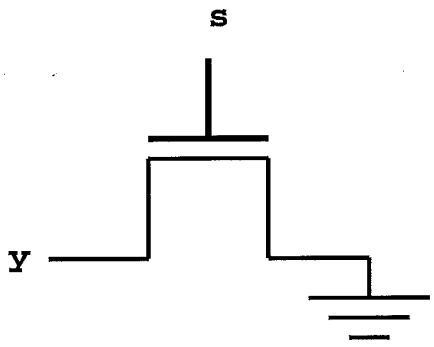
Conversie functies

```
FUNCTION To_bit       ( s : std_ulogic;       xmap : BIT := '0') RETURN BIT IS
BEGIN
        CASE s IS
            WHEN '0' | 'L' => RETURN ('0');
            WHEN '1' | 'H' => RETURN ('1');
            WHEN OTHERS => RETURN xmap;
        END CASE;
END;
```

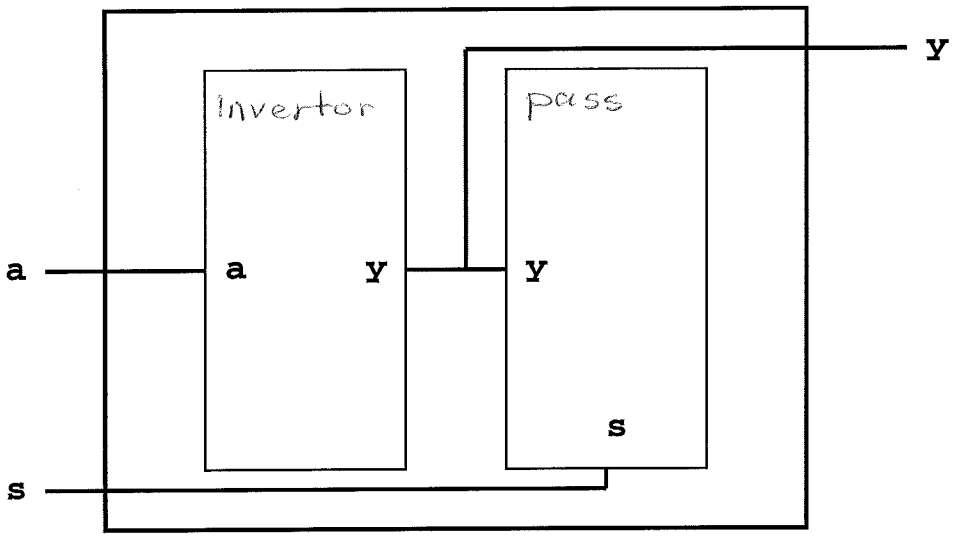
```
FUNCTION To_StdULogic       ( b : BIT              ) RETURN std_ulogic IS
BEGIN
    CASE b IS
        WHEN '0' => RETURN '0';
        WHEN '1' => RETURN '1';
    END CASE;
END;
```



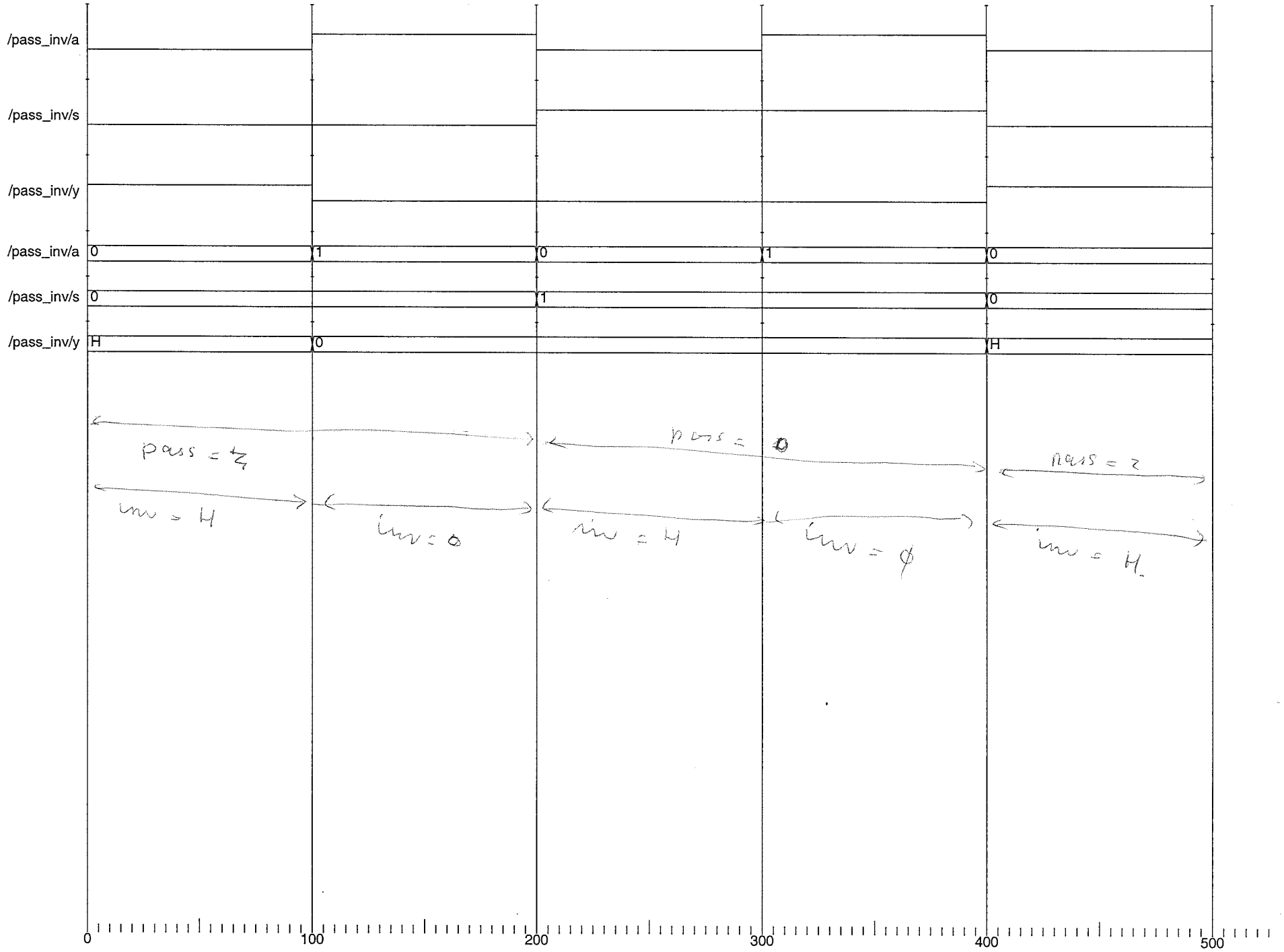

```
library IEEE;
use ieee.std_logic_1164.all;

entity invertor is
   port (a: in  bit;
         y: out std_logic);
end invertor;

architecture behaviour of invertor is
begin
   lbl1:process(a)
   begin
      if (a = '0') then
         y <= 'H';
      else
         y <= '0';
      end if;
   end process;
end behaviour;
```



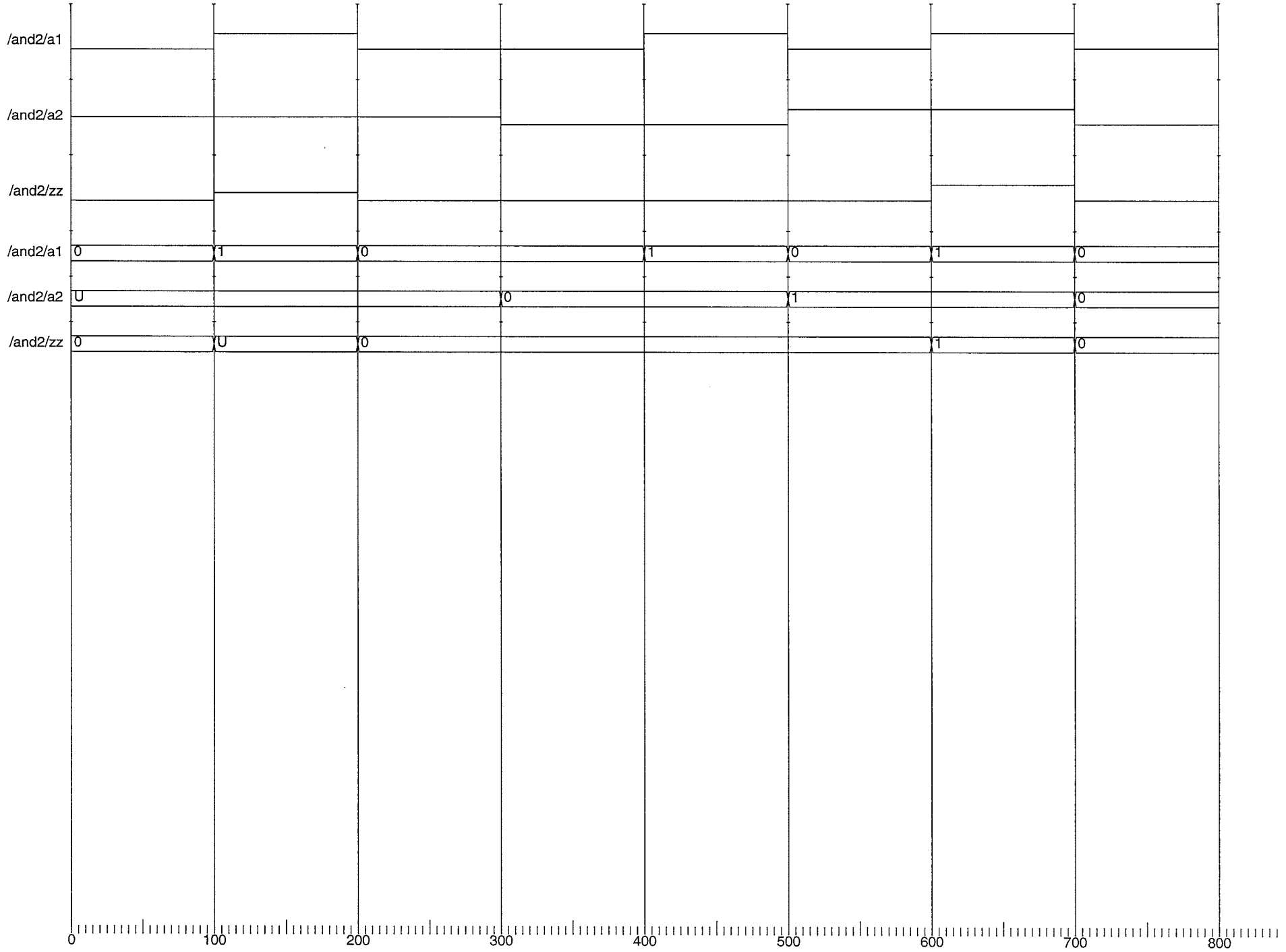

```
library IEEE;
use ieee.std_logic_1164.all;
entity pass is
   port (s : in  bit;
         y : out std_logic);
end pass;


architecture behaviour of pass is
begin
   lbl1:process(s)
   begin
      if (s = '0') then
         y <= 'Z';
      else
         y <= '0';
      end if;
   end process;
end behaviour;
```




```
library IEEE;
use ieee.std_logic_1164.all;
entity pass_inv is
   port (a: in  bit;
         s: in  bit;
         y: out std_logic);
end pass_inv;

architecture struct of pass_inv is
   component invertor
      port (a: in  bit;
            y: out std_logic);
   end component;
   component pass
      port (s : in  bit;
            y : out std_logic);
   end component;
begin
   lbl1: invertor port map (a, y);
   lbl2: pass     port map (s, y);
end struct;
```


/pass_inv/a
/pass_inv/s
/pass_inv/y
/pass_inv/a
/pass_inv/s
/pass_inv/y
0
1
0
1
0
0
1
0
H
0
H
pass = Z
pass = 0
pass = Z
inv = H
inv = 0
inv = H
inv = ϕ
inv = H
0
100
200
300
400
500

```
library IEEE;
use IEEE.std_logic_1164.all;

entity and2 is
   port (a1: in  bit;
         a2: in  std_logic;
         zz: out std_logic);
end and2;

architecture behaviour of and2 is
begin
   zz <= to_stdulogic(a1) and a2;
end behaviour;
```



Entity:and2 Architecture:behaviour Date: Mon Feb 03 09:45:34 W. Europe Daylight Time 2003 Row: 1 Page: 1

Bij (rekenkundige) bewerkingen met
bit of std_logic vectoren hangt het
resultaat af van de interpretatie van de
betreffende vectoren

Vraag is:

Zijn het absolute grootheden (unsigned)
of zijn het grootheden met een teken
(two's complement, signed)

Daarom is het definieren van (rekenkundige)
bewerkingen op bit_vectoren en std_logic_vectoren
weinig zinvol,en worden hiervoor nieuwe types
gedefinieerd:

```
type UNSIGNED is array(NATURAL range <>) of BIT
type SIGNED is array(NATURAL range <>) of BIT

type UNSIGNED is array(NATURAL range <>) of STD_LOGIC
type SIGNED is array(NATURAL range <>) of STD_LOGIC
```

Nieuwe types

SIGNED en UNSIGNED

gedefinieerd in bijv de packages:

numeric_bit en numeric_std

rekenkundige operatoren

+, -, * /, rem, mod, abs

vergelijkings operatoren

<, >, =, /=, <=, >=

verdere operatoren

resize

verdere conversiefuncties

TO_INTEGER

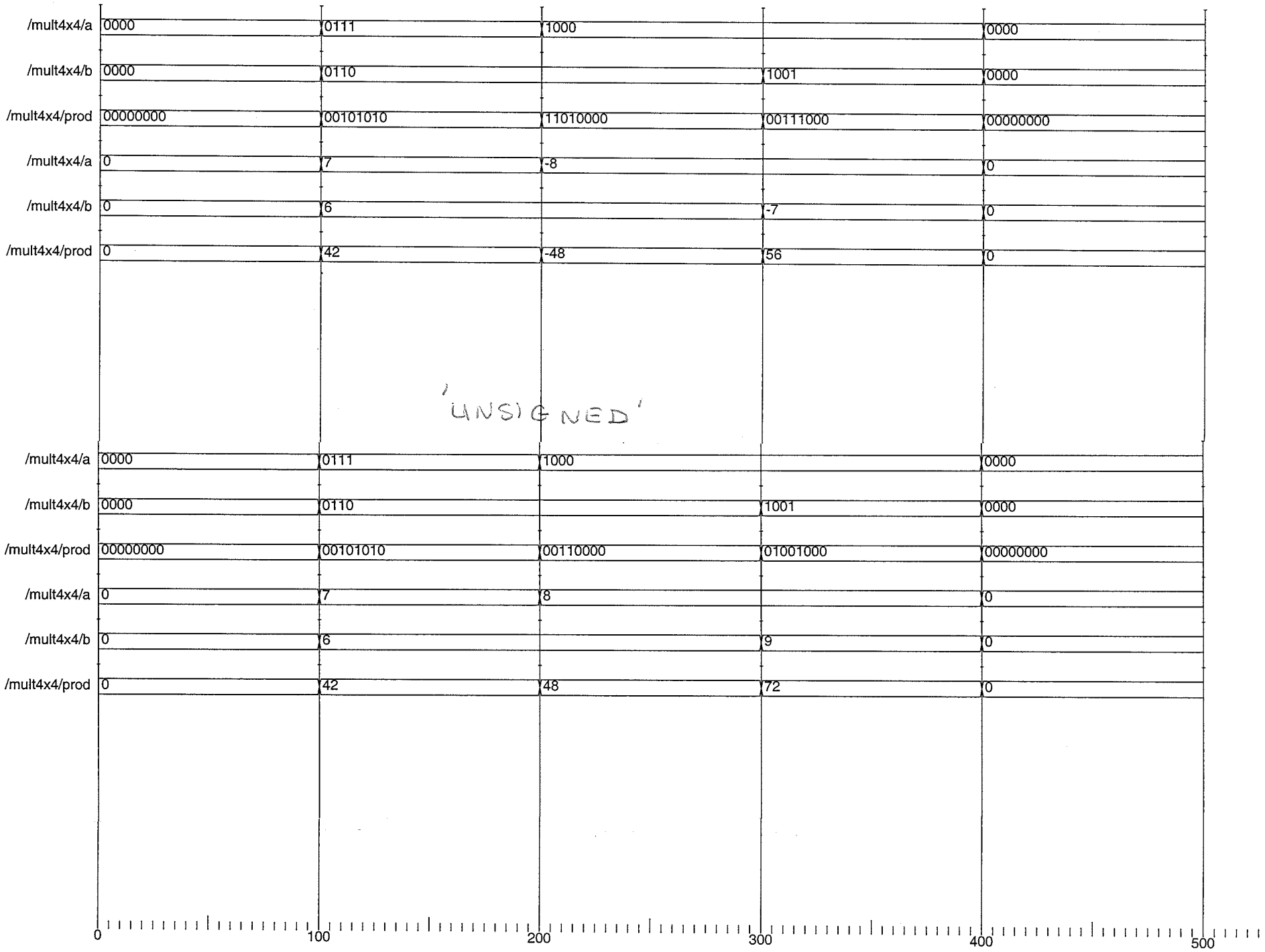
TO_UNSIGNED

TO_SIGNED

```
library ieee;
use ieee.numeric_bit.all;

entity mult4x4 is
   port (a: in  bit_vector(3 downto 0);
         b: in  bit_vector(3 downto 0);
         prod: out bit_vector(7 downto 0));

end mult4x4;

architecture beh of mult4x4 is
begin
   prod <= bit_vector(signed(a) * signed(b));
end beh;
```

```
library ieee;
use ieee.numeric_bit.all;

entity mult4x4 is
   port (a: in  bit_vector(3 downto 0);
         b: in  bit_vector(3 downto 0);
         prod: out bit_vector(7 downto 0));
end mult4x4;

architecture beh of mult4x4 is
begin
   prod <= bit_vector(unsigned(a) * unsigned(b));
end beh;
```

'SIGNED'



'UNSIGNED'

Entity:mult4x4 Architecture:beh Date: Mon Feb 03 10:20:52 W. Europe Daylight Time 2003 Row: 1 Page: 1