Jacob Lundkvist ELA21 – Digital Konstruktion

1. Genomför nedanstående omvandlingar mellan olika talsystem:

a) Omvandla talet 1101 0110 till dess decimala motsvarighet, både på signerad samt osignerad form.

Osignerat: 214 (decimalt)

Signerat: MSB är 1, dvs talet blir ett negativt tal. $1101\ 0110_{\ 2} - 1 = 1101\ 0101_2$. Räkna bort MSB för tillfället och invertera talet, 1->0 - 0->1. = $0010\ 1010_2$ = 32 + 8 + 2 = 42. Med MSB som inverterar talet negativt
SVAR = -42_{10}

b) Omvandla talet 300 till dess 16-bitars signerade motsvarighet.

SVAR: $0000\ 0001\ 0010\ 1100_2$ (Skulle kunna förkorta bort alla nollor)

c) Omvandla det decimala talet -123 till dess 8-bitars 2-komplement.

SVAR: -123 till positivt tal är (0111 1011), inverteras till 1000 0100 +1 = $1000\ 0101_2$

d) Omvandla det binära talet 0110 1101 0111 1010 till dess hexadecimala motsvarighet.

$SVAR: 0110\ 1101\ 0111\ 1010 = 6\ D\ 7\ A = 6D7A_{16}$

e) Omvandla det hexadecimala talet $3F754B04_{16}$ till dess binära motsvarighet.

$$SVAR: 3F754B04_{16} = \underset{3}{0011}\ \underset{F}{1111}\ \underset{7}{0111}\ \underset{5}{0101}\ \underset{4}{0100}\ \underset{B}{1011}\ \underset{0}{0000}\ \underset{4}{0100}_2$$

2. ErrorCode

```
----------------------------------------------------------------------------------
-- Modulen ErrorCode används för att tända lysdioden led vid udda antal höga
-- slideswitchar
-- switch[3:0], förutsatt att tryckknappen key_n är nedtryckt.
-- Annars skall lysdioden vara släckt.
-- Därmed gäller att om en eller tre slide-switchar är höga samtidigt som
-- tryckknappen key_n trycks ned, så tänds lysdioden, annars hålls den släckt.
----------------------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all; -- lagt till .all

entity ErrorCode is
   port
   (
      switch : in std_logic_vector(3 downto 0);
      key_n  : in std_logic;
      led    : out std_logic     -- // TAGIT BORT SEMICOLON
   );
end entity;

architecture Behaviour of ErrorCode is
signal led_s        : std_logic; -- lagt till std_logic signal
signal odd_switches : std_logic; -- Ändrat långt namn och gjort om variabel till signal,
-- bool till std_logic.
begin
----------------------------------------------------------------------------------
-- Vid udda antal höga slide switchar samtidigt som tryckknappen key_n trycks ned så
-- tänds lysdioden, annars hålls den släckt.
--
-- ändrat sensitivity list så odd_switches ingår istället för input från switch
--
-- tagit bort överflödig begin från process.
-- ändrat från !key_n till key = '0'. Lagt till jämförelse om odd_switches = '1'
-- lagt till ordet "then" efter jämförelsen.
-- lagt till tilldelning av värde "<= och '' "
--
-- Använder mig av led_s för att kunna tilldela output utanför processen
--
----------------------------------------------------------------------------------
process (odd_switches, key_n) is
begin
   if (odd_switches = '1' and key_n = '0') then
      led_s <= '1';  -- LAGT TILL TILLDELNING AV VÄRDE "<= och '' "
   else
      led_s <= '0';  -- LAGT TILL TILLDELNING AV VÄRDE "<= och '' "
   end if;
end process;

   odd_switches <= switch(3) xor switch(2) xor switch(1) xor switch(0);
   led          <= led_s;

end architecture;
```

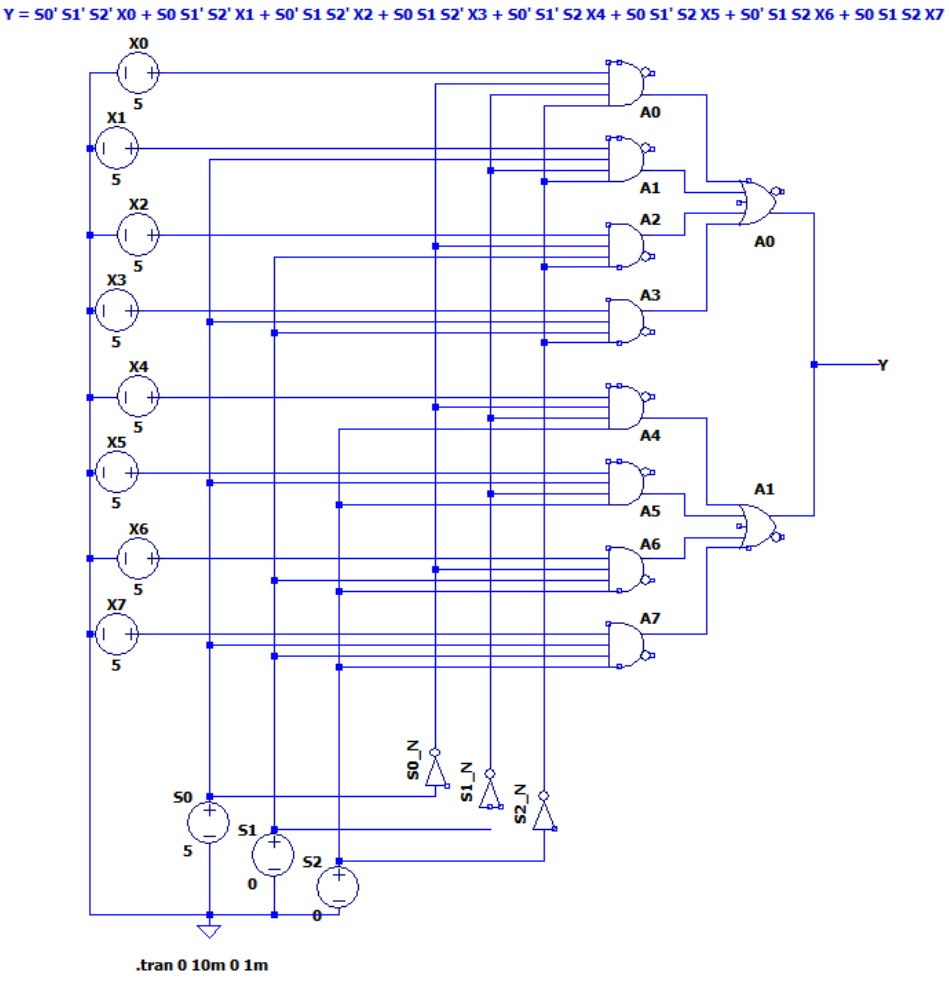
Jacob Lundkvist ELA21 – Digital Konstruktion

3. 8-1 MUX

Dont care införs i sanningstabellen då Multiplexerns bitar S2-S0

$$Y = S0'S1'S2'C0 + S0\,S1'S2'C1 + S0'S1\,S2'C2 + S0\,S1\,S2'\,C3 + S0'\,S1'\,S2\,C4 + S0\,S1'\,S2\,C5 + S0'\,S1\,S2\,C6 + S0\,S1\,S2\,C7$$

S2	S1	S0	C0	C1	C2	C3	C4	C5	C6	C7	Y
0	0	0	1	X	X	X	X	X	X	X	1
0	0	1	X	1	X	X	X	X	X	X	1
0	1	0	X	X	1	X	X	X	X	X	1
0	1	1	X	X	X	1	X	X	X	X	1
1	0	0	X	X	X	X	1	X	X	X	1
1	0	1	X	X	X	X	X	1	X	X	1
1	1	0	X	X	X	X	X	X	1	X	1
1	1	1	X	X	X	X	X	X	X	1	1



Jacob Lundkvist ELA21 – Digital Konstruktion

4. 3 bitars AD-omvandlare

Signalen X sanningstabell och Karnaugh

A	B	C	D	E	F	G	H	X
0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	1	X	0
0	0	0	0	0	1	X	X	0
0	0	0	0	1	X	X	X	0
0	0	0	1	X	X	X	X	1
0	0	1	X	X	X	X	X	1
0	1	X	X	X	X	X	X	1
1	X	X	X	X	X	X	X	1

X	EFGH	0000	0001	0011	0010	0100	0101	0111	0110	1000	1001	1011	1010	1100	1101	1111	1110
ABCD	0000																
	0001	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	0011	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	0010	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	0100	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	0101	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	0111	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	0110	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	1000	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	1001	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	1011	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	1010	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	1100	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	1101	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	1111	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	1110	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Signalen Y sanningstabell och Karnaugh

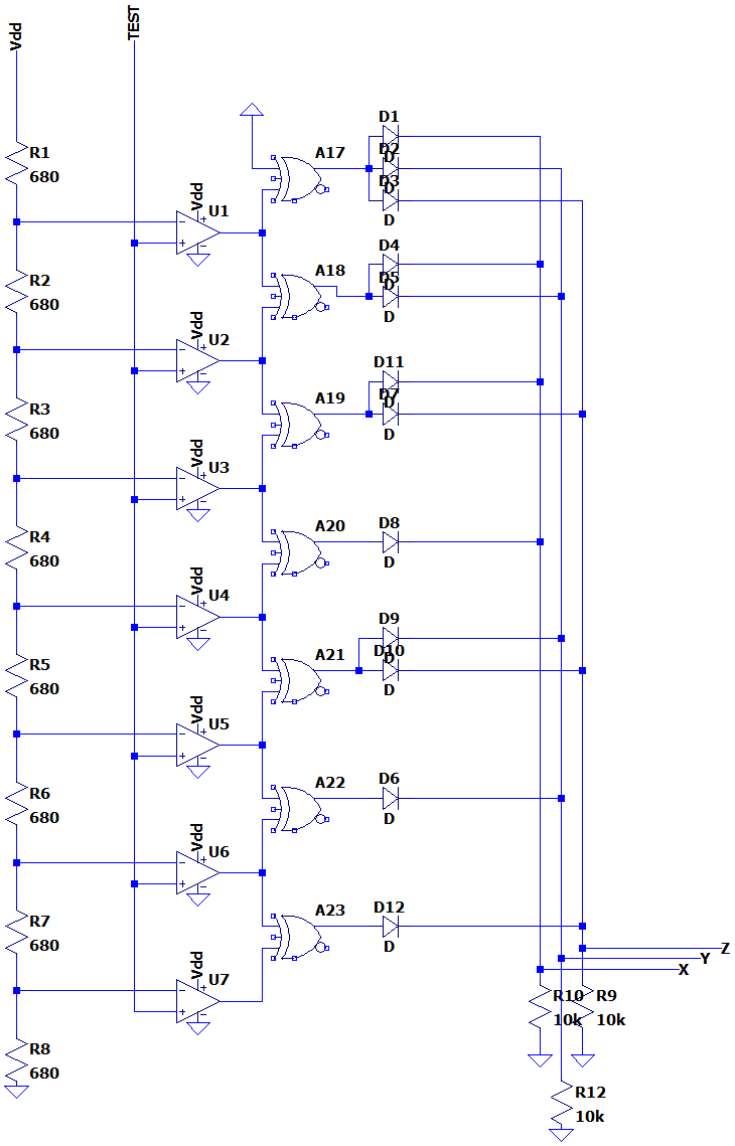
A	B	C	D	E	F	G	H	Y
0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	1	X	0
0	0	0	0	0	1	X	X	1
0	0	0	0	1	X	X	X	1
0	0	0	1	X	X	X	X	0
0	0	1	X	X	X	X	X	0
0	1	X	X	X	X	X	X	1
1	X	X	X	X	X	X	X	1

Y	EFGH	0000	0001	0011	0010	0100	0101	0111	0110	1000	1001	1011	1010	1100	1101	1111	1110
ABCD	0000					1	1	1	1	1	1	1	1	1	1	1	1
	0001																
	0011																
	0010																
	0100	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	0101	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	0111	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	0110	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	1000	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	1001	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	1011	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	1010	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	1100	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	1101	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	1111	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	1110	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Signalen Z sanningstabell och Karnaugh

A	B	C	D	E	F	G	H	Z
0	0	0	0	0	0	0	1	
0	0	0	0	0	0	1	X	1
0	0	0	0	0	1	X	X	
0	0	0	0	1	X	X	X	1
0	0	0	1	X	X	X	X	
0	0	1	X	X	X	X	X	1
0	1	X	X	X	X	X	X	
1	X	X	X	X	X	X	X	1

Z	EFGH	0000	0001	0011	0010	0100	0101	0111	0110	1000	1001	1011	1010	1100	1101	1111	1110
ABCD	0000			1	1					1	1	1	1	1	1	1	1
	0001																
	0011	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	0010	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	0100																
	0101																
	0111																
	0110																
	1000	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	1001	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	1011	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	1010	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	1100	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	1101	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	1111	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	1110	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1



Vid 5V så ger XOR (A) grinden 3 höga utsignaler, X Y Z

Vid 4.35V ger XOR (B) 2 höga utsignaler, X Y

Vid 3.7V ger XOR (C) 2 höga utsignaler, X Z

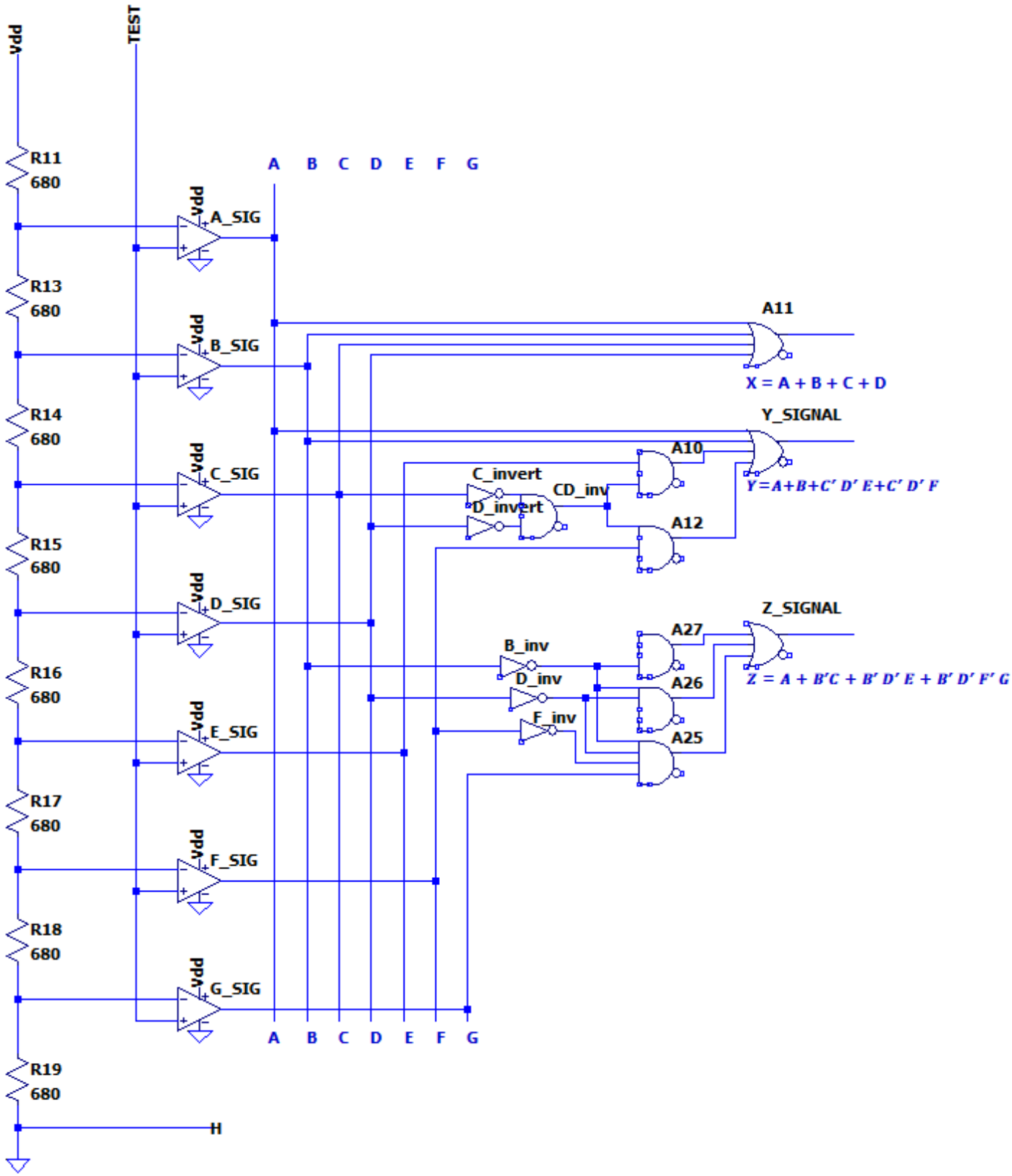
Vid 3.1V ger XOR (D) 1 hög utsignal, X

Vid 2.45V ger XOR (E) 2 höga utsignaler, Z Y

Vid 1.85V ger XOR (F) 1 hög utsignal, Y

Vid 1.2V ger XOR (G) 1 hög utsignal, Z

Signalen H ger ingen hög utsignal och kan bortses.


Vdd
TEST
R11
680
R13
680
R14
680
R15
680
R16
680
R17
680
R18
680
R19
680
H
A B C D E F G
A_SIG
B_SIG
C_SIG
D_SIG
E_SIG
F_SIG
G_SIG
A11
X = A + B + C + D
Y_SIGNAL
A10
C_invert
D_invert
CD_inv
A12
Y=A+B+C' D' E+C' D' F
Z_SIGNAL
A27
B_inv
D_inv
F_inv
A26
A25
Z = A + B'C + B' D' E + B' D' F' G

5.

A[1:0]		B[1:0]		X[3:0]	4	2	1	
A1	**A0**	**B1**	**B0**	**X3**	**X2**	**X1**	**X0**	
0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0
0	0	1	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0
0	1	0	0	0	0	0	0	0
0	1	0	1	0	0	0	1	1
0	1	1	0	0	0	1	0	2
0	1	1	1	0	0	1	1	3
1	0	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0	2
1	0	1	0	0	1	0	0	4
1	0	1	1	0	1	1	0	6
1	1	0	0	0	0	0	0	0
1	1	0	1	0	0	1	1	3
1	1	1	0	0	1	1	0	6
1	1	1	1	1	0	0	1	9

A[1:0]	B[1:0]			
X0	**00**	**01**	**11**	**10**
00	0	0	0	0
01	0	1	1	0
11	0	1	1	0
10	0	0	0	0

A[1:0]	B[1:0]			
X2	**00**	**01**	**11**	**10**
00	0	0	0	0
01	0			
11	0			1
10	0		1	1

A[1:0]	B[1:0]			
X1	**00**	**01**	**11**	**10**
00	0	0	0	0
01	0	0	1	1
11	0	1	0	1
10	0	1	1	0

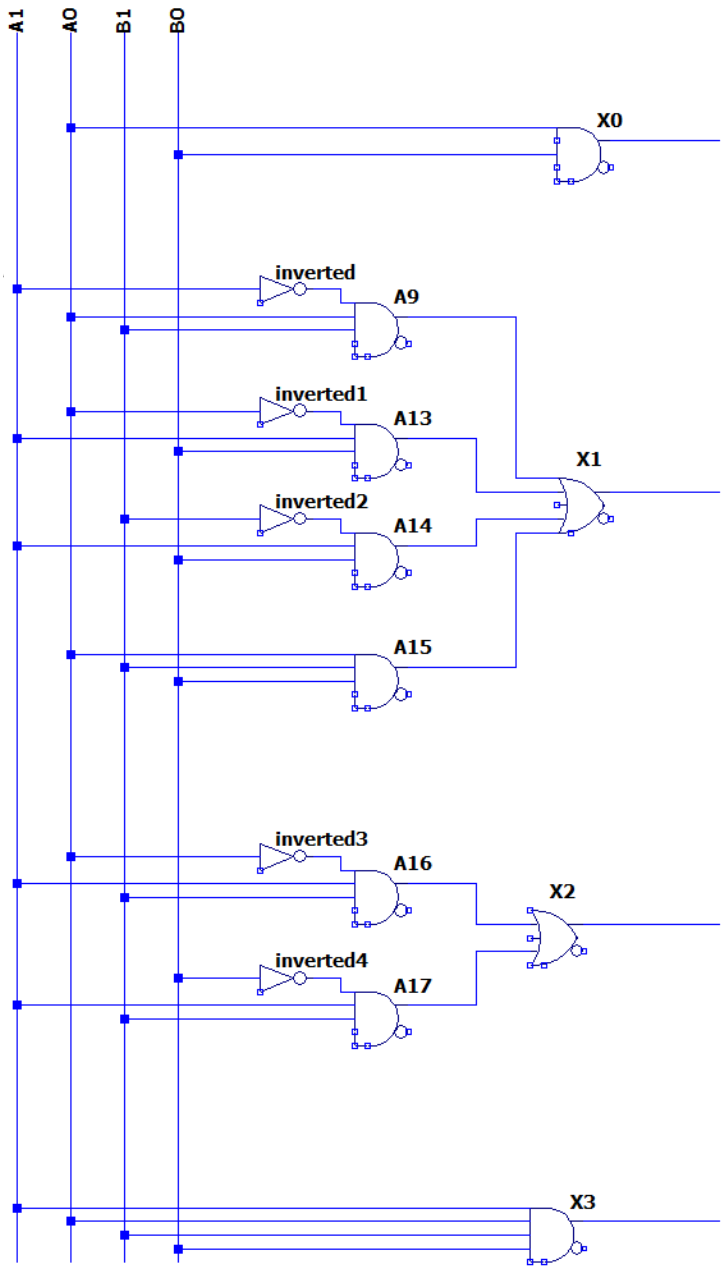
A[1:0]	B[1:0]			
X3	**00**	**01**	**11**	**10**
00	0	0	0	0
01	0			
11	0		1	
10	0			

$X0 = A0\ B0$

$X1 = A1'A0\ B1 + A1\ A0'\ B0 + A1\ B1'\ B0 + A0\ B1\ B0$

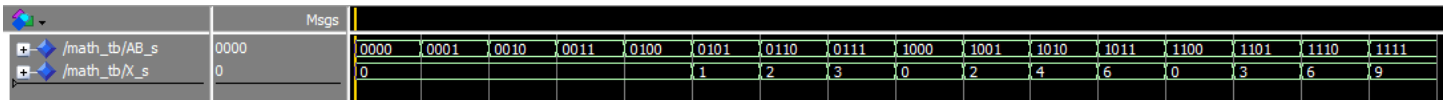
$X2 = A1\ A0'\ B1 + A1B1\ B0'$

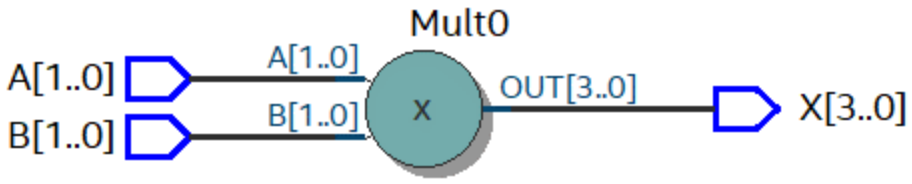
$X3 = A1\ A0\ B1\ B0$



Jacob Lundkvist ELA21 – Digital Konstruktion

Version 1. Typomvandlar ”unsigned vektor” till integer och sedan typomvandlas det tillbaka till unsigned -> std_logic_vector.






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

entity math is
   port
   (
      A,B : in  unsigned(1 downto 0);
      X   : out std_logic_vector(3 downto 0)
   );
end entity;

-- Typomvandlar min unsigned vektor till heltal via to_integer
-- multiplicerar insignalerna med varandra, sedan typomvandlas
-- resultatet först till unsigned vektor med längden av vektorn X
-- för att till slut omvandlas till std_logic_vector och tilldelas
-- till variabeln X.
architecture behaviour of math is
begin
   X <= std_logic_vector(to_unsigned(to_integer(A) * to_integer(B),X'length));
end architecture;
```

Testad på FPGA med godkänt resultat.

Jacob Lundkvist ELA21 – Digital Konstruktion

Version 2. Med logiska uttryck framtaget från Karnaughdiagram.

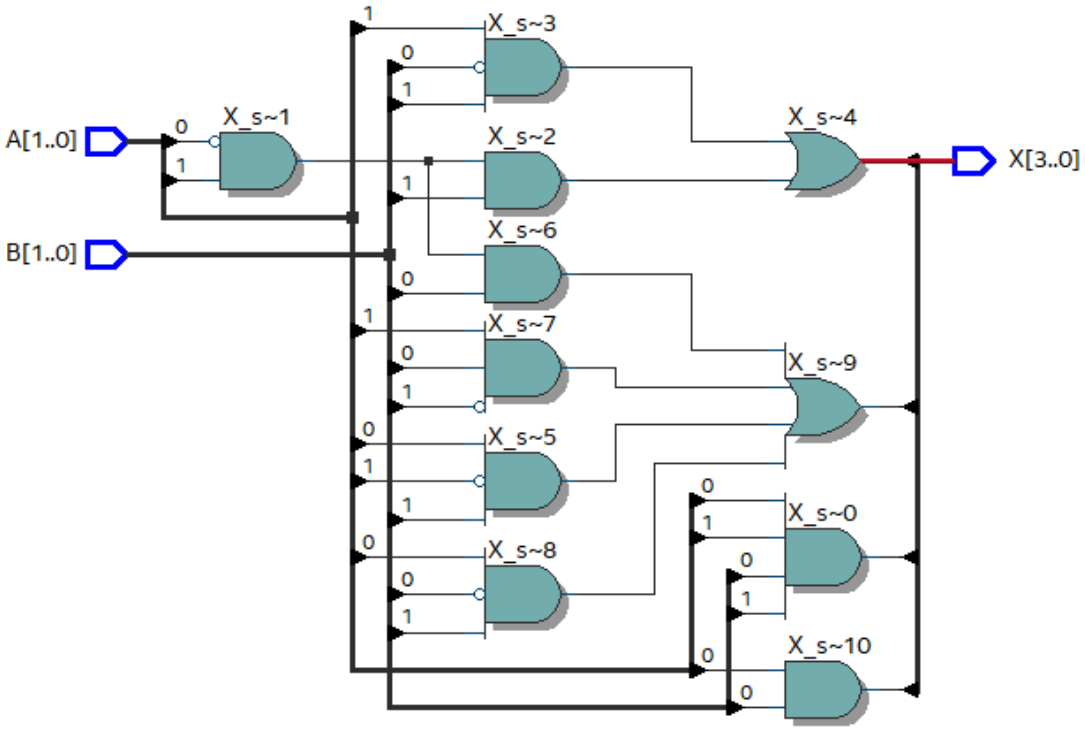
```
begin
   process(A, B)
   begin
      X_s(3) <= (A(1) and A(0) and B(1) and B(0));

      X_s(2) <= (A(1) and not A(0) and B(1)) or
                (A(1) and B(1) and not B(0));

      X_s(1) <= (not A(1) and A(0) and B(1)) or
                (A(1) and not A(0) and B(0)) or
                (A(1) and not B(1) and B(0)) or
                (A(0) and B(1) and not B(0));

      X_s(0) <= (A(0) and B(0));

   end process;
   X <= X_s;
end architecture;
```



Testad på FPGA med godkänt resultat.