Task 1 Morse Beacon 2018

Logic Systems And Processors (České Vysoké Učení Technické v Praze)

Task 1: KOM circuit for Morse Beacon - 15 points

Version of document 2018.V1 from October of 8th, 2018

Assignment

Create KOM combinational circuit with two outputs **Y** and STOP that has the following function: The binary sequence assigned to you appears on **Y** output if the 8-bit counter drives X inputs. Each student has a special sequence that corresponds to the Morse code derived from his first name and surname name as a unique identifier in all exercises☺. For its creation, we also took into account the similar logical complexity for all students.

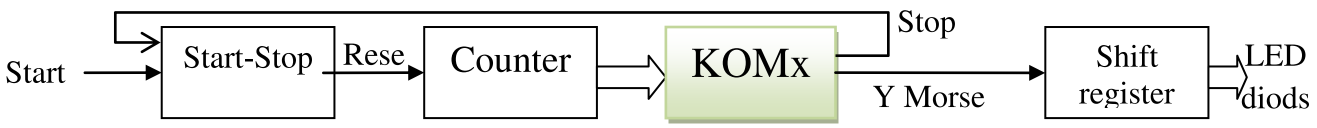


Figure 1 - Block iagram

Requirements:

The task has a training character for getting to know FPGAs. Thus, many limitations are specified:

- Create COM in two versions, such as KOM1 in VHDL (i.e. * .vhd file), where you directly write the found logical equations from Karnaugh maps. Nothing but logical equations and signal definitions are allowed in the VHDL architecture block of KOM1.
- After you fully debug KOM1, then create KOM2 as a separate Quartus II Block / Schematic File, where you rewrite the equation in the logic schema. In this task, you clarify the relationship between the different logic circuit descriptions and find out why VHDL is used for circuits.
- In the schematic KOM2 circuit, you can utilize any number of NOT, AND, NAND, OR, NOR, and XOR gates, input and output ports, but **at the most one 4input multiplexer**. The benefits from multiplexers are shown below in the solved example, and they will also be described in the third lecture. It is forbidden to use in KOM2 other types of circuits than mentioned above.
- First, test KOM1 and KOM2 with the aid of Quartus simulator demonstrated on the practical exercises.
- When your KOM1 and KOM2 are debugged, use them in schematic together with a shift register to see Morse code as running light snake and a beeper to here sound. Then, you can demonstrate the result to tutor to obtain credits.

Note. You can utilize any circuit to test or demonstrate KOM behavior.

Example: - Morse Beacon EA from 1st lecture

The Beacon EA, shown at the first lecture, has Beacon output M = 010001011100 sequence. The counter begins counting on START='1'. It counts up in the sequence 0, 1, 2, 3, 4, 5 ... etc. If STOP='1', the counter is reset to zero and is stopped.

STOP and Beacon outputs looks like this:

Counter8bitsStartStop Q7..Q0	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
KOM-output: **STOP**	0	0	0	0	0	0	0	0	0	0	0	**1**
KOM-output **M- Morse Code** EA	0	1	0	0	0	1	0	1	1	1	0	0

KOM output-STOP is in logical '1' in the last bit of the sequence.

Logic equations were designed in 1st lecture:

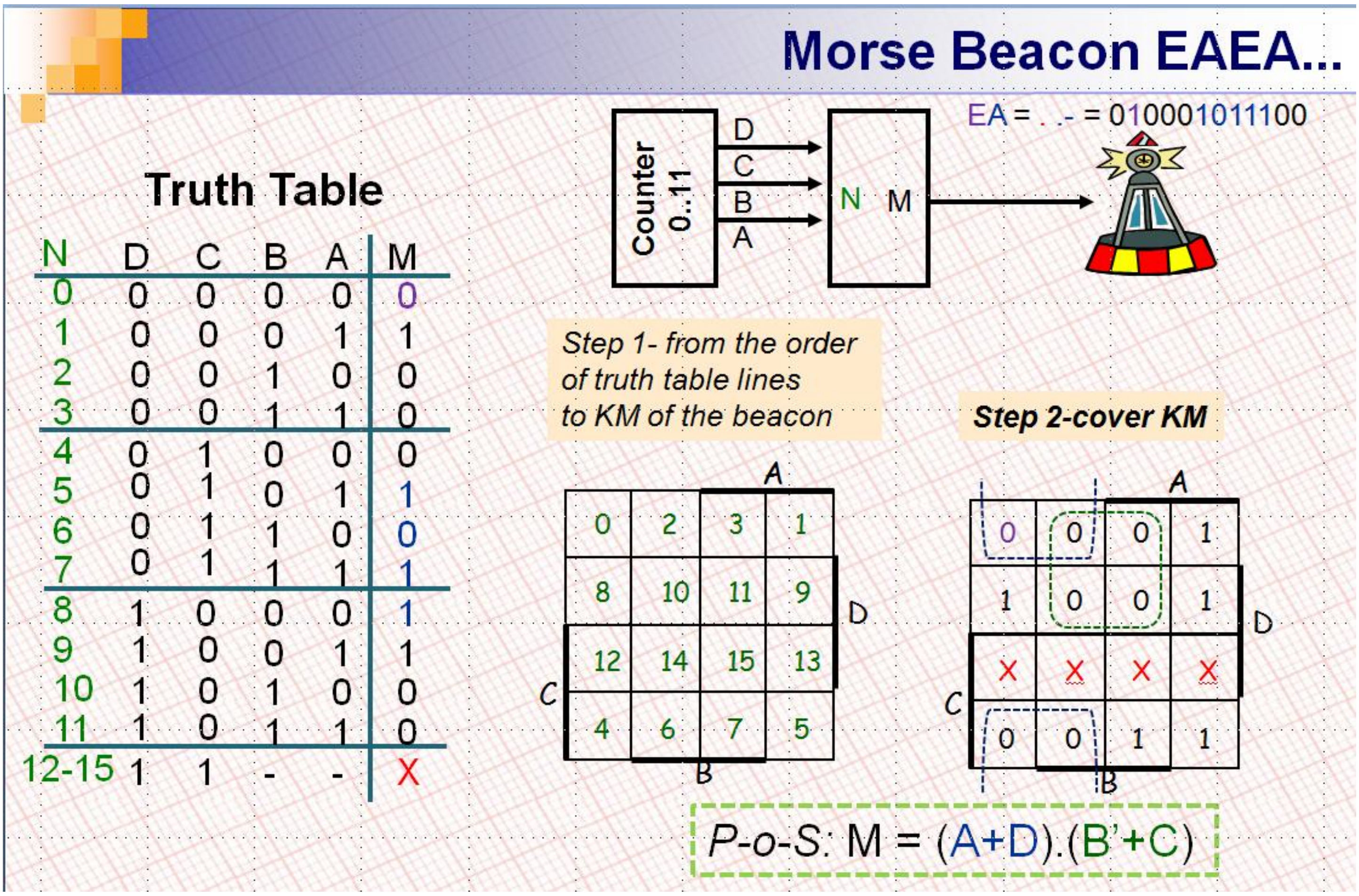


Figure 2 - Beacon EA from lectures

In equations, we replace D, C, B, A by X3, X2, X1, X0 to emphasize that X0 is the lowest scale bit.

Y = (X0+X3).(X1'+X2);

We add STOP when count is :>= 11

STOP = X3 . X1 . X0;

In VHDL language, we write directly equations. Remember that brackets are in VHDL absolutely necessary.

```vhdl
library ieee; use ieee.std_logic_1164.all;
entity KOM1ea is
        port
        (       X0,X1,X2,X3    : in std_logic;
                STOP, Y : out std_logic );
end entity;

architecture behavioral of KOM1ea is
begin
        Y <= (X0 or X3) and (not X1 or X2);
        STOP <= X3 and X1 and X0;
end architecture;
```

In the home, you can simulate by the way discussed at seminars. Translate KOM1ea as top-level entity and subsequently simulate it using the file type University program VWF (* .wvf - Vector Waveform File).

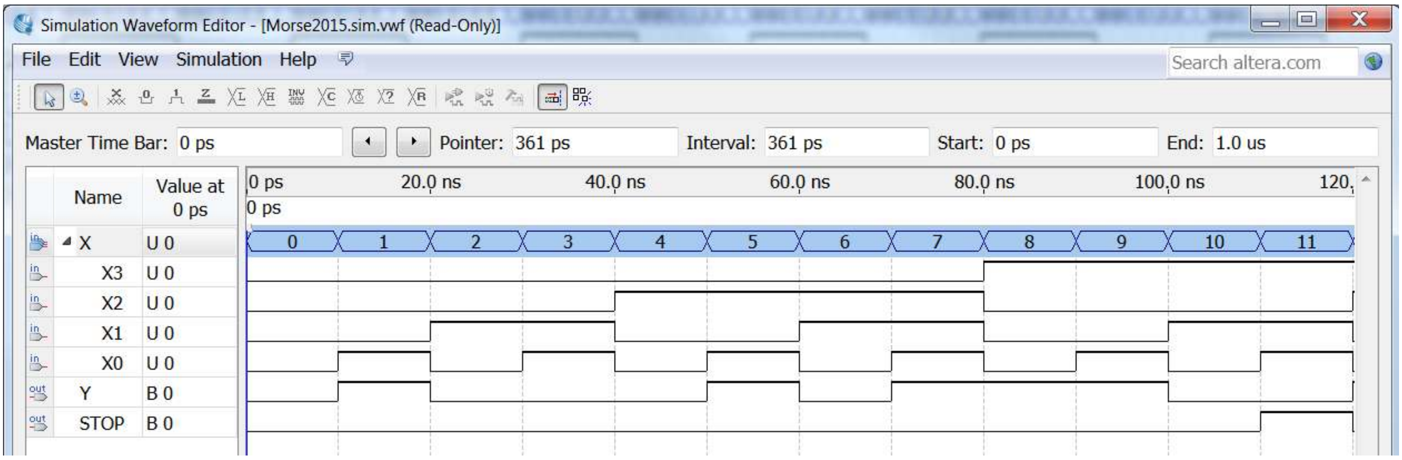


Figure 3- Output of simulator

In the laboratory, you can test in the following circuit by the manual clock KEY[2].

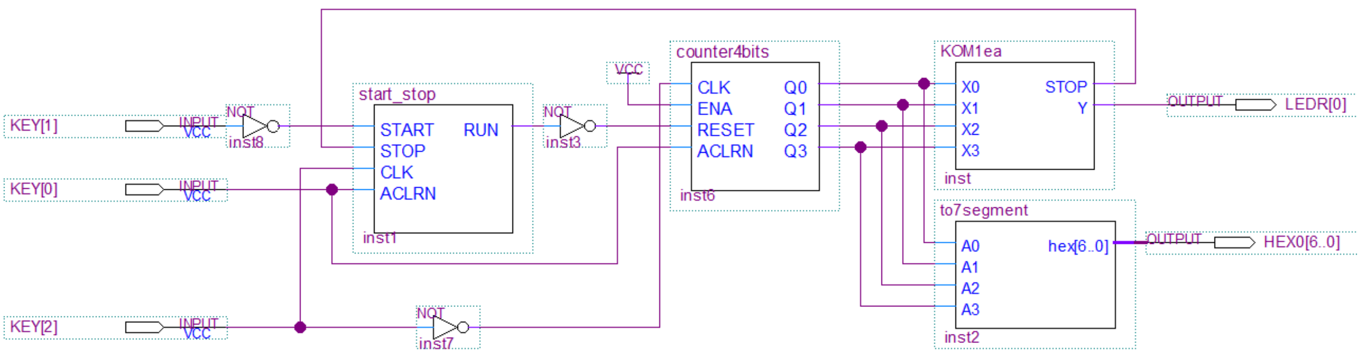


Figure 4 - Test circuit by manual clock KEY[2]

Beware, if you want to use KEY[3] instead of KEY[2], you can, but you must also insert input KEY[2] and leave it unconnected, because members of vector pins of type YYY[x] cant be skipped in Quartus. You must always insert into drawing all pins between 0 to some upper used index.

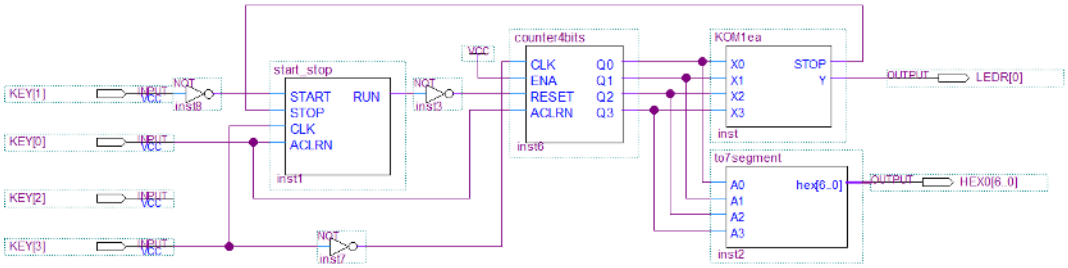


Figure 5 - Test circuit by manual clock KEY[3]

The designed KOM2 circuit from the 1st lecture:

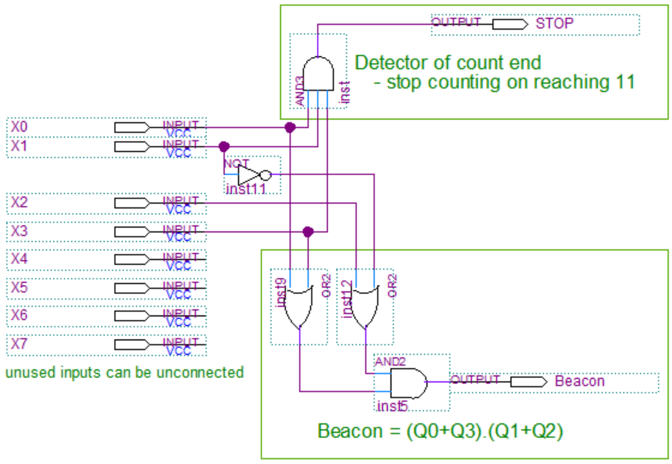


Figure 6 - BDF schema

Its function can be tested by this circuit:

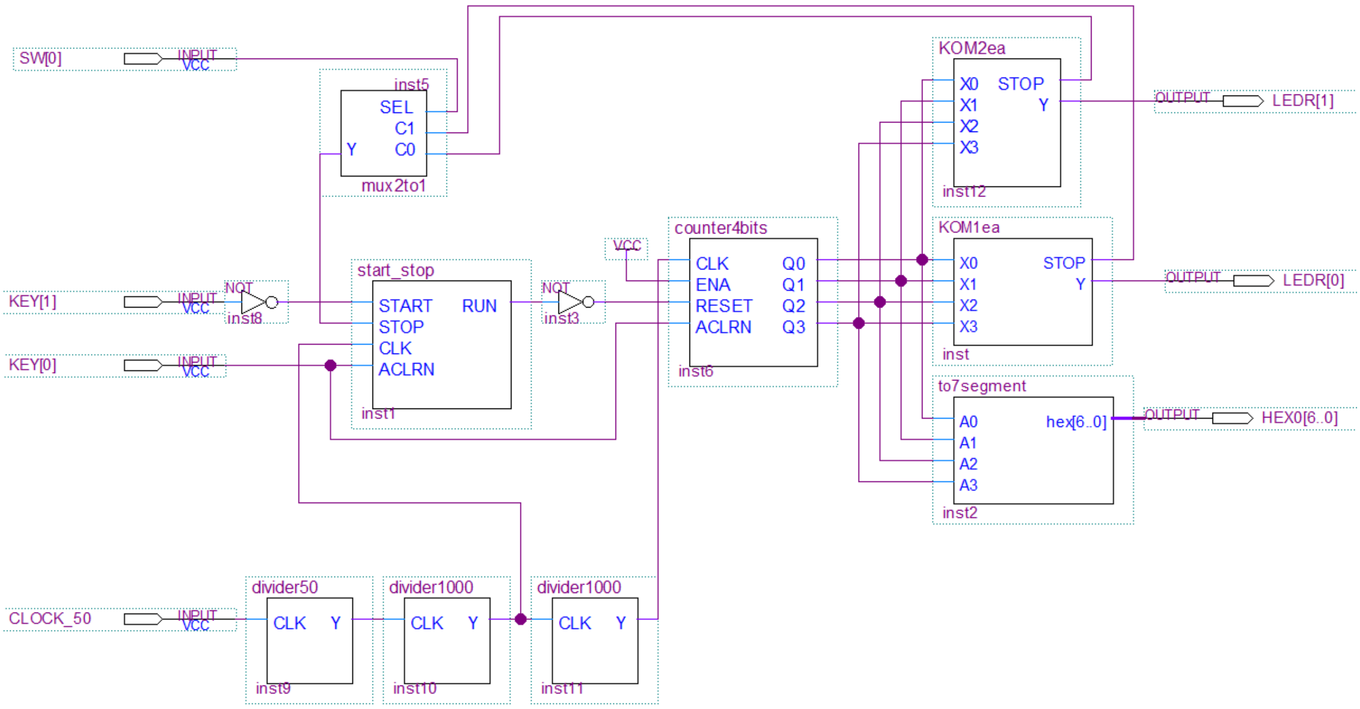


Figure 7 - 2 KOM in schematic with slow clock 1 Hz

CLOCK_50, 50 MHz clock input, is divided here to 1 Hz. The Start-Stop Circuit, is driven by 2kHz clock to achieve a faster response to pushing a button. The 2-input multiplexer 'mux2to1' selects STOP signal from KOM1ea or KOM2ea.

In the final connection, **we accelerate the frequency to 5 Hz**, because the slow Morse code is difficultly readable, we add the multiplexer to the output and the scroll register and the buzzer. In this form, you can present your solution to the teacher.

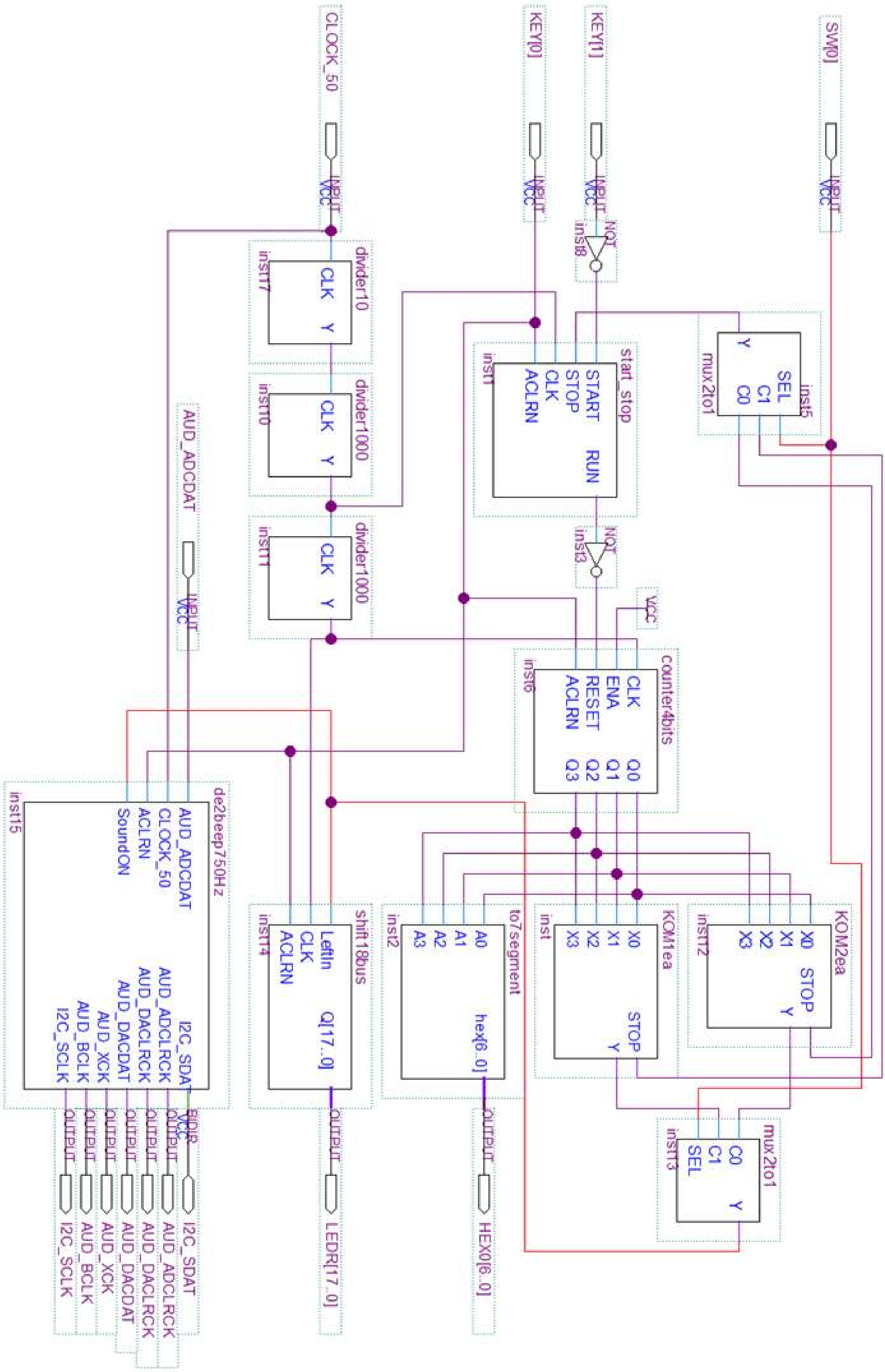


Figure 8 - Final schematic for demonstration

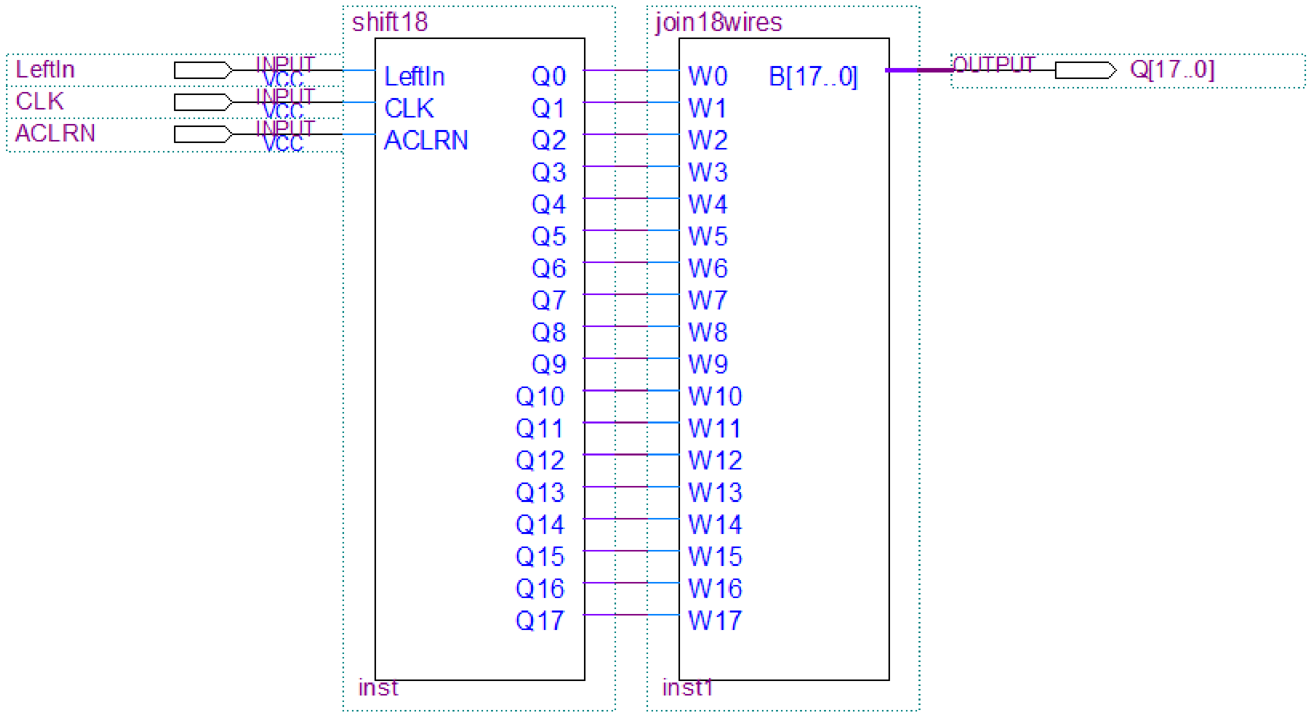


Figure 9 - Created circuit saved as shift18bus

The scheme contains shift18bus circuit that does not exist in the DCE library . It is composed of 2 circuits from the library and saved it as shift18bus.bdf. After generating its symbol file (Menu: File -> Create / Update - > Create Symbols Files for Current File...), you can insert shift18bus into the schematic.

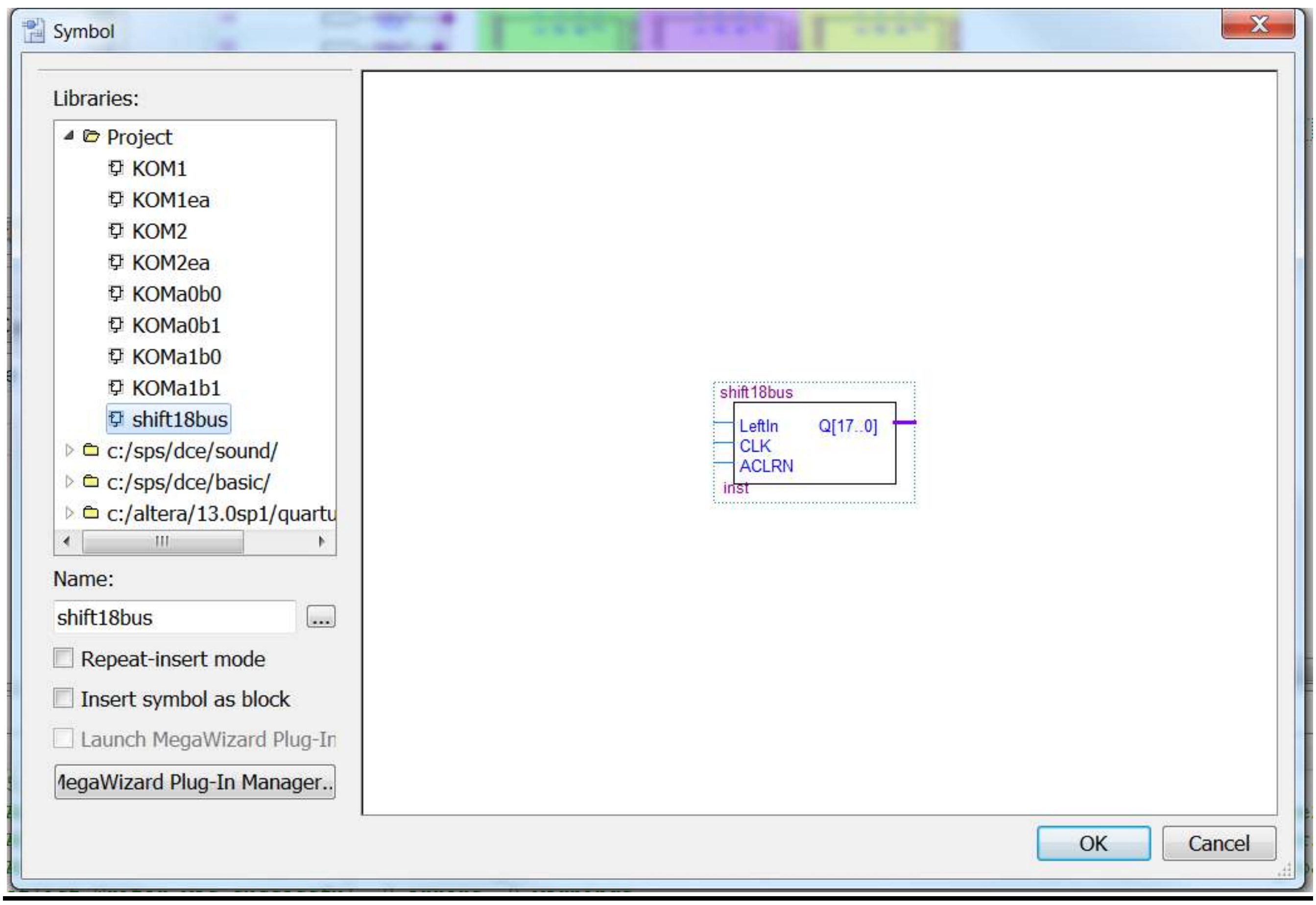


Figure 10 - Inserting your new circuit

MHL2
01110111000101010100010111010100010101110111011100

<u>*Timing complies with the International Maritime Standard:*</u>

- *short mark, dot or "dit": "dot duration" is one time unit long;*
- *longer mark, dash or "dah": three time units long;*
- *inter-element gap between the dots and dashes within a character: one dot duration or one unit long;*
- *short gap (between letters): three time units long.*

<u>*Why does the sequence begin with 0 and end with a pair of 00?*</u>

In the beginning, in the STOP state, nothing happens. However, when the beacon is running continuously (we keep the START button permanently pressed), the correct 3-time pause appears because your binary sequence starts at 0 and ends 00 so that during continuous transmission, the last character is separated from the first by the correct gap between characters three times units long, i.e. 000.

<u>*How to convert another sequence?*</u>

Typically, various mnemonic tools are known: M as 'MA-MA', H as 'hitting the hip', L as 'a light is lit'. There is nothing for numbers, but their codes have clear structures. It should be noted here that ***mnemotechnical aids were never used*** *in real radiotelegraphy because they would only slow down operators, so their teaching was very strictly forbidden. Telegraphists listened to Morse like to music and perceived characters as individual tones of melody.*

There are some sites to convert, for example, http://www.onlineconversion.com/morse_code.htm

Designing equations

To reduce Boolean equations, we substitute names of input from X0 to X5 for avoiding mistypes - frequent sources of errors. We write:

KOM- input name:	X0	X1	X2	X3	X4	X5
Our substituted variable:	f	e	d	c	b	a

First, we rewrite our binary sequence into 6-input truth table, see the next page:

<u>**Note:** This year, you have obtained 5 character codes, but nobody has his whole Morse sequence longer than 64-bits.</u>

Stav	a	b	c	d	e	f	X (KOM output Beacon)	Y (KOM – output STOP)
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	1	1	0
2	0	0	0	0	1	0	1	0
3	0	0	0	0	1	1	1	0
4	0	0	0	1	0	0	0	0
5	0	0	0	1	0	1	1	0
6	0	0	0	1	1	0	1	0
7	0	0	0	1	1	1	1	0
8	0	0	1	0	0	0	0	0
9	0	0	1	0	0	1	0	0
10	0	0	1	0	1	0	0	0
11	0	0	1	0	1	1	1	0
12	0	0	1	1	0	0	0	0
13	0	0	1	1	0	1	1	0
14	0	0	1	1	1	0	0	0
15	0	0	1	1	1	1	1	0
16	0	1	0	0	0	0	0	0
17	0	1	0	0	0	1	1	0
18	0	1	0	0	1	0	0	0
19	0	1	0	0	1	1	0	0
20	0	1	0	1	0	0	0	0
21	0	1	0	1	0	1	1	0
22	0	1	0	1	1	0	0	0
23	0	1	0	1	1	1	1	0
24	0	1	1	0	0	0	1	0
25	0	1	1	0	0	1	1	0
26	0	1	1	0	1	0	0	0
27	0	1	1	0	1	1	1	0
28	0	1	1	1	0	0	0	0
29	0	1	1	1	0	1	1	0
30	0	1	1	1	1	0	0	0
31	0	1	1	1	1	1	0	0
32	1	0	0	0	0	0	0	0
33	1	0	0	0	0	1	1	0
34	1	0	0	0	1	0	0	0
35	1	0	0	0	1	1	1	0
36	1	0	0	1	0	0	0	0
37	1	0	0	1	0	1	1	0
38	1	0	0	1	1	0	1	0
39	1	0	0	1	1	1	1	0
40	1	0	1	0	0	0	0	0
41	1	0	1	0	0	1	1	0
42	1	0	1	0	1	0	1	0
43	1	0	1	0	1	1	1	0
44	1	0	1	1	0	0	0	0
45	1	0	1	1	0	1	1	0
46	1	0	1	1	1	0	1	0
47	1	0	1	1	1	1	1	0
48	1	1	0	0	0	0	0	0
49	1	1	0	0	0	1	0	1
50	1	1	0	0	1	0	X	X
51	1	1	0	0	1	1	X	X
52	1	1	0	1	0	0	X	X
53	1	1	0	1	0	1	X	X
54	1	1	0	1	1	0	X	X
55	1	1	0	1	1	1	X	X
56	1	1	1	0	0	0	X	X
57	1	1	1	0	0	1	X	X
58	1	1	1	0	1	0	X	X
59	1	1	1	0	1	1	X	X
60	1	1	1	1	0	0	X	X
61	1	1	1	1	0	1	X	X
62	1	1	1	1	1	0	X	X
63	1	1	1	1	1	1	X	X

The minimization of logic functions by Karnaugh map from the truth table above would be very complicated. Therefore, we use different techniques. We simplify rhe **Beacon** table by Shannon expansion and solve STOP signal equation by knowledge of the circuit behavior.

First, we begin by Shannon's expansion of Beacon output. We split the big truth table into 4 smaller truth tables (Shannon cofactors) using **a** and **b** as splitting variables, see 2^{nd} lecture:

a = 0, b = 0

Stav	c	d	e	f	Y (KOM output Beacon)
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	0
11	1	0	1	1	1
12	1	1	0	0	0
13	1	1	0	1	1
14	1	1	1	0	0
15	1	1	1	1	1

---

a = 0, b = 1

Stav	c	d	e	f	Y (KOM output Beacon)
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	1
10	1	0	1	0	0
11	1	0	1	1	1
12	1	1	0	0	0
13	1	1	0	1	1
14	1	1	1	0	0
15	1	1	1	1	0

---

a = 1, b = 0

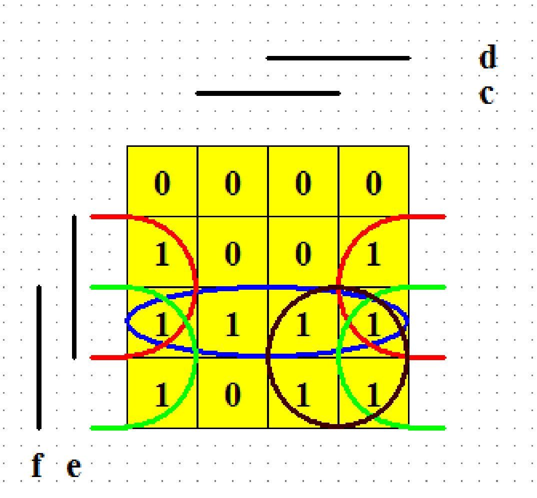
Stav	c	d	e	f	Y (KOM output Beacon)
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	0
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	1
10	1	0	1	0	1
11	1	0	1	1	1
12	1	1	0	0	0
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	1

---

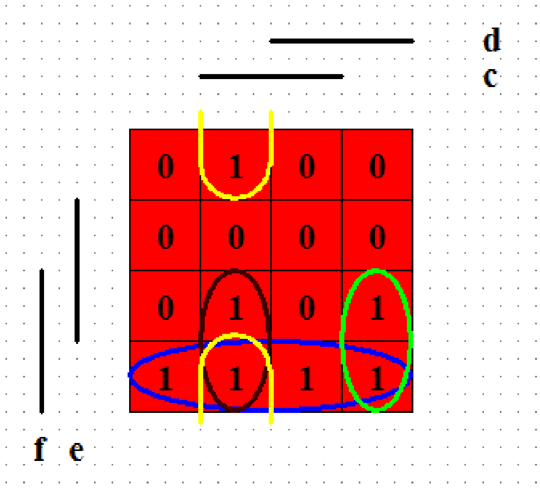
a = 1, b = 1

Stav	c	d	e	f	Y (KOM output Beacon)
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	X
3	0	0	1	1	X
4	0	1	0	0	X
5	0	1	0	1	X
6	0	1	1	0	X
7	0	1	1	1	X
8	1	0	0	0	X
9	1	0	0	1	X
10	1	0	1	0	X
11	1	0	1	1	X
12	1	1	0	0	X
13	1	1	0	1	X
14	1	1	1	0	X
15	1	1	1	1	X

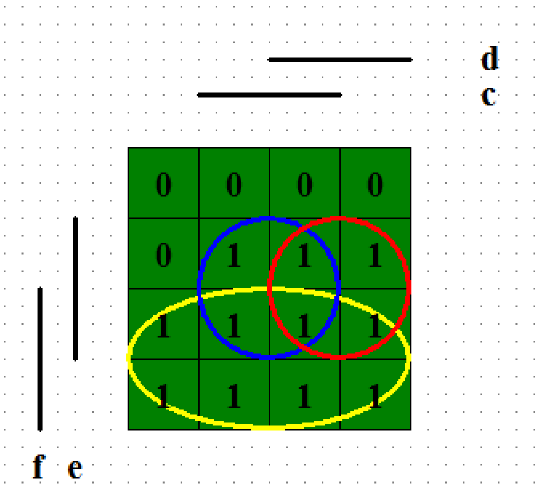
Now we deal with minimization of much simple logic functions - Karnaugh maps have only 4 variables. We create 4 Y functions denoted as Yab = **f(c,d,e,f)**, where **'a' and 'b'** are values of logic inputs **'a', 'b'** for a given map.



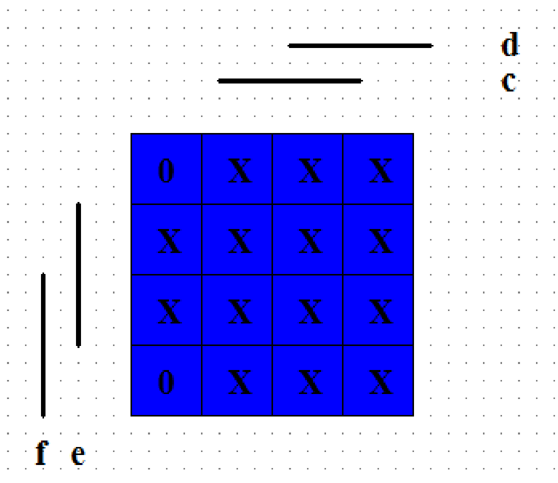
$Y00 = ef + \bar{c}e + \bar{c}f + df$



$Y01 = \bar{e}f + c\bar{d}f + \bar{c}df + c\bar{d}\bar{e}$



$Y10 = ce + de + f$



Is equals to logical contradiction Y11 = 0

For output STOP, we utilize knowledge of the circuit behavior.

We know that the counter counts from zero up to STOP state, then it resets. So its sequence always ends in number 49, which has the binary code:

$$49_{10} = 110001_2 = 2^5 + 2^4 + 1$$

Number 49 must be the first number in the sequence from 0 to 49. It has '1' in the 5th, 4th and 0th bit, i.e., a='1' and b='1' and f='1'. No other such number can occur earlier because it would be greater than 49, which is contrary to the nature of arithmetical series. Thus, we can write a simple equation for STOP output:

$$STOP = abf$$

Test VHDL KOM1 on DE2 board

--Note: Use only ASCII characters in VHDL comments.

```
library ieee; use ieee.std_logic_1164.all;
entity KOM1 is
        port
        (
                X0, X1, X2, X3, X4, X5 : in std_logic;
                STOP, Y : out std_logic
        );
end entity;
architecture behavioral of KOM1 is
signal a, b, c, d, e, f : std_logic;
signal Y00, Y01, Y10, Y11 : std_logic;
begin
-- renaming signals
   a <= X5; b <= X4; c <= X3; d <= X2; e <= X1; f <= X0;
   Y00 <= (e and f) or (not c and e) or (not c and f) or (d and f);
   Y01 <=  (not e and f) or (c and not d and f) or (not c and d and f)
             or (c and not d and not e);
   Y10 <=  (c and e) or (d and e) or f;
   Y11 <= '0';
-- Multiplexer --
      Y <=       (not a and not b and Y00)
                 or (not a and b and Y01)
                 or (a and not b and Y10)
                 or (a and b and Y11);
       STOP <=        a and b and f;
end architecture;
```

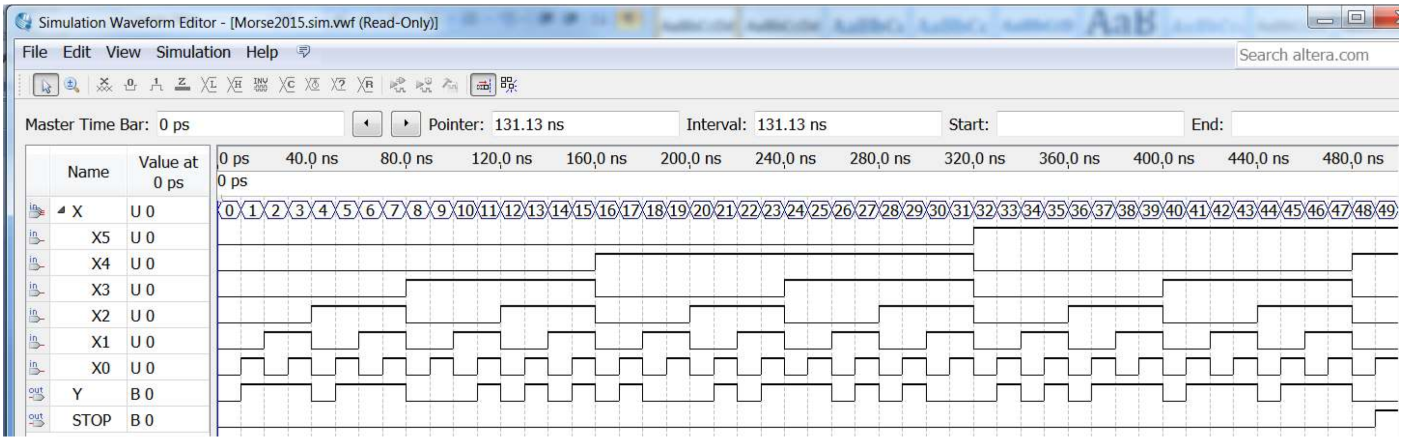


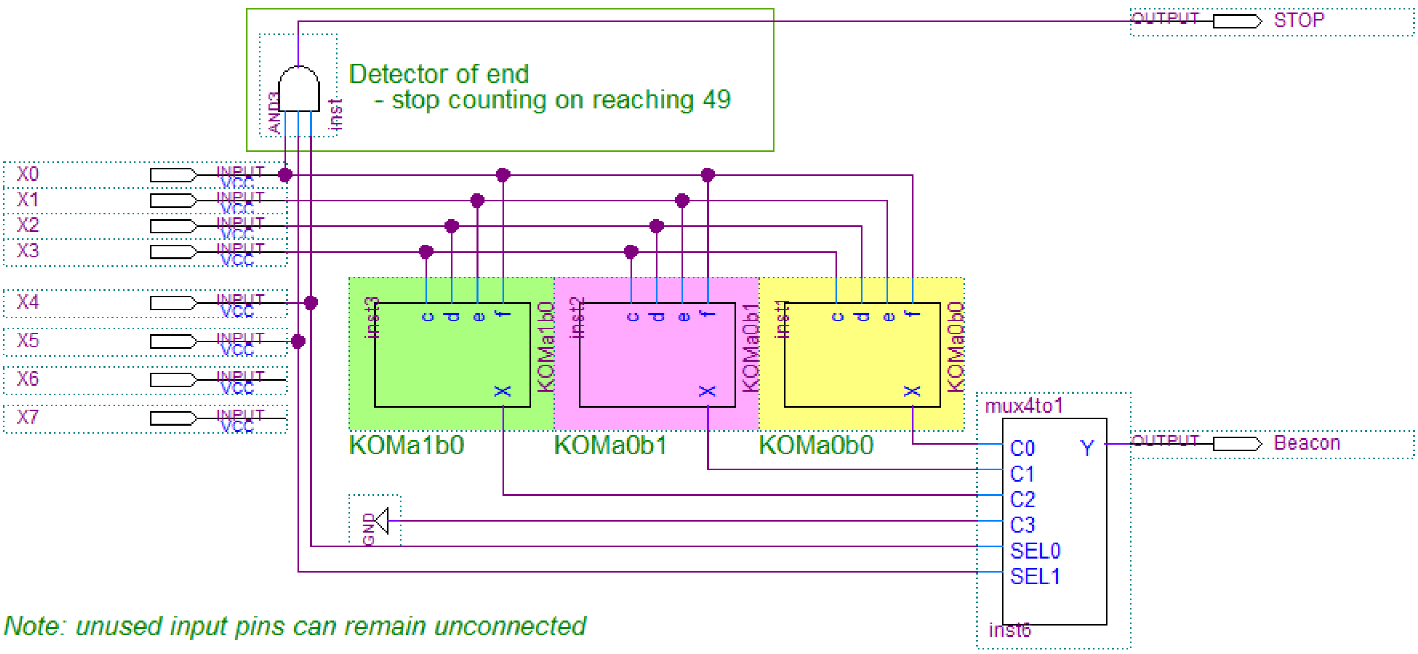
Figure 11 - Test of VHDL code

Schematic KOM2

The best way how to join of cofactor tables into final Beacon output is the usage of a multiplexer, whose address inputs are splitting variables of Shannon's expansion, i.e., **a** and **b**.

First we create each Karnaugh map as 4input simple logic function X = f (c, d, e, f). We name them according to the values of **a** and **b** inputs as KOMa0b0, KOMa0b1, and KOMa1b0. The function KOMa1b1 was not created; it was reduced to 0, thus the ground. Each function could be optionally independently tested by simulations, which is other advantages of this method.

Finally, we create symbolic files for our KOMa0b0, KOMa0b1 and KOMa1b0. We insert them as circuits into the new scheme and join their output by the multiplexer from the DCE/basic library



At the beginning of this example, we renamed inputs: X0 corresponds to **f,** X1 to **e,** X2 to **d,** X3 to **c,** X4 to **b,** X5 to **a. Where do we rename them back to original names?**
We easily change names of pins in KOM above ☺

The test circuit own is the same as for the EA morse beacon with the only difference, and we utilize the 8-bit counter after Start-Stop circuit.

Appendix:

We could write all KOMx in one equation:

$$Y = \bar{a}\bar{b}\,(ef + \bar{c}e + \bar{c}f + df) + \bar{a}b\,\left(\bar{e}f + c\bar{d}f + \bar{c}df + c\bar{d}\bar{e}\right) + a\bar{b}\,(ce + de + f)$$

After simplifications:

$$Y = f\,(\bar{a}\bar{b}(e + \bar{c} + d) + \bar{a}b(\bar{e} + c\bar{d} + \bar{c}d) + a\bar{b}) + \bar{a}(\bar{b}\bar{c}e + bc\bar{d}\bar{e}) + a\bar{b}e(c + d)$$

The equation results in a huge schematic, thus, we did not do it by this way. **We recommend you also not to do.** Students who have tried create all in one schematic in recent years have spent much more time than those who have chosen to split up their solution to 4 smaller Karnaugh maps, and then join them by the multiplexor.

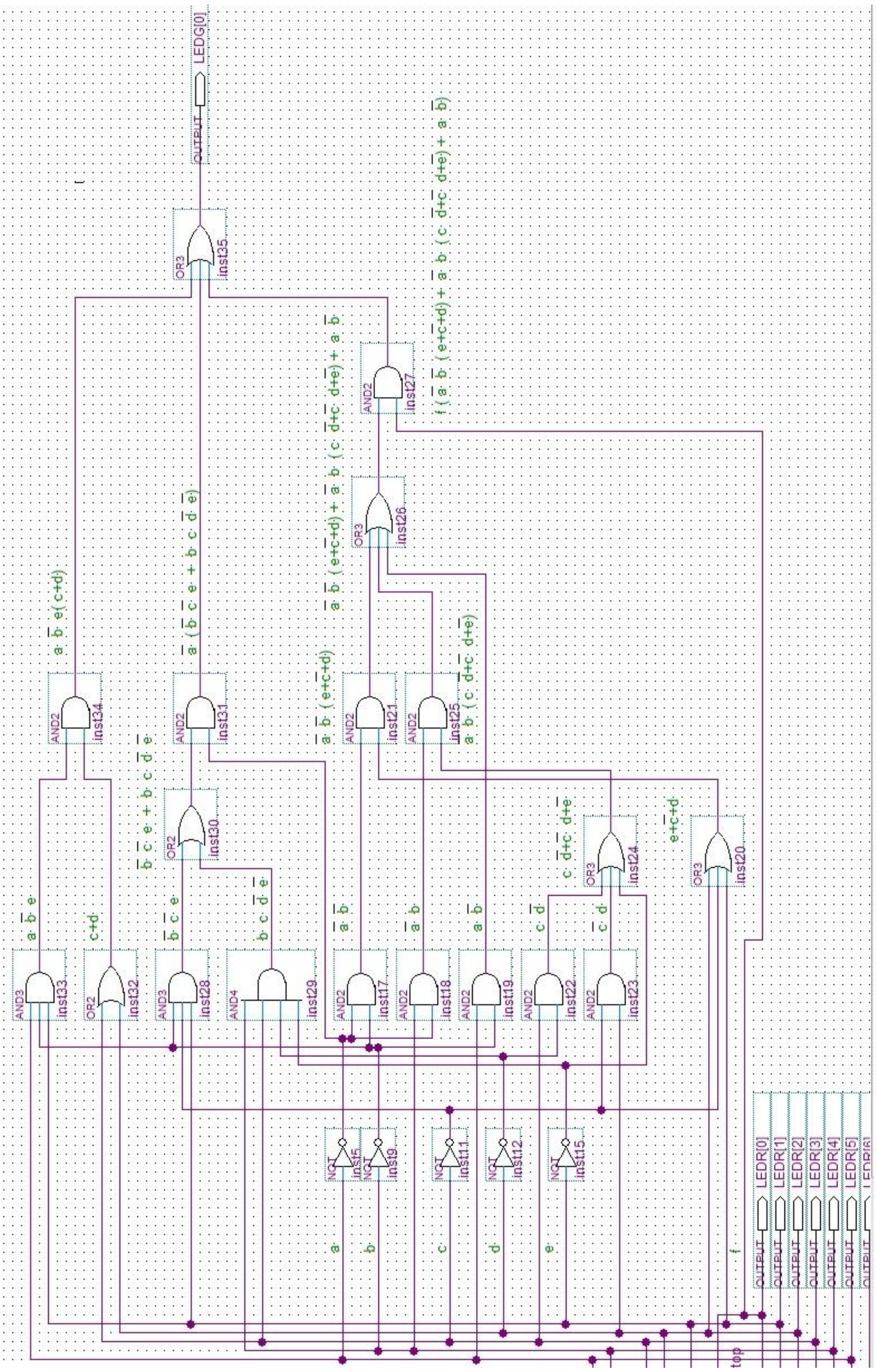


Figure 12 - Unrecomended 'All in one'