

**Variable frequency signal generation**

Embedded Systems

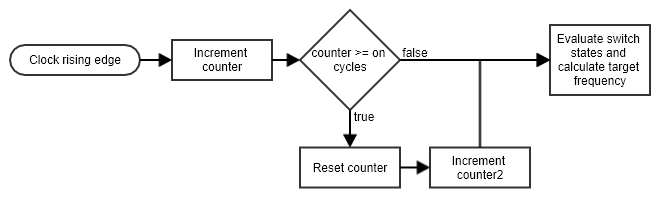
Intermediate Exam 1

Done by: EKSfmu-16 gr. st. Arūnas Butkus

Checked by: doc. dr. Andrius Ušinskas

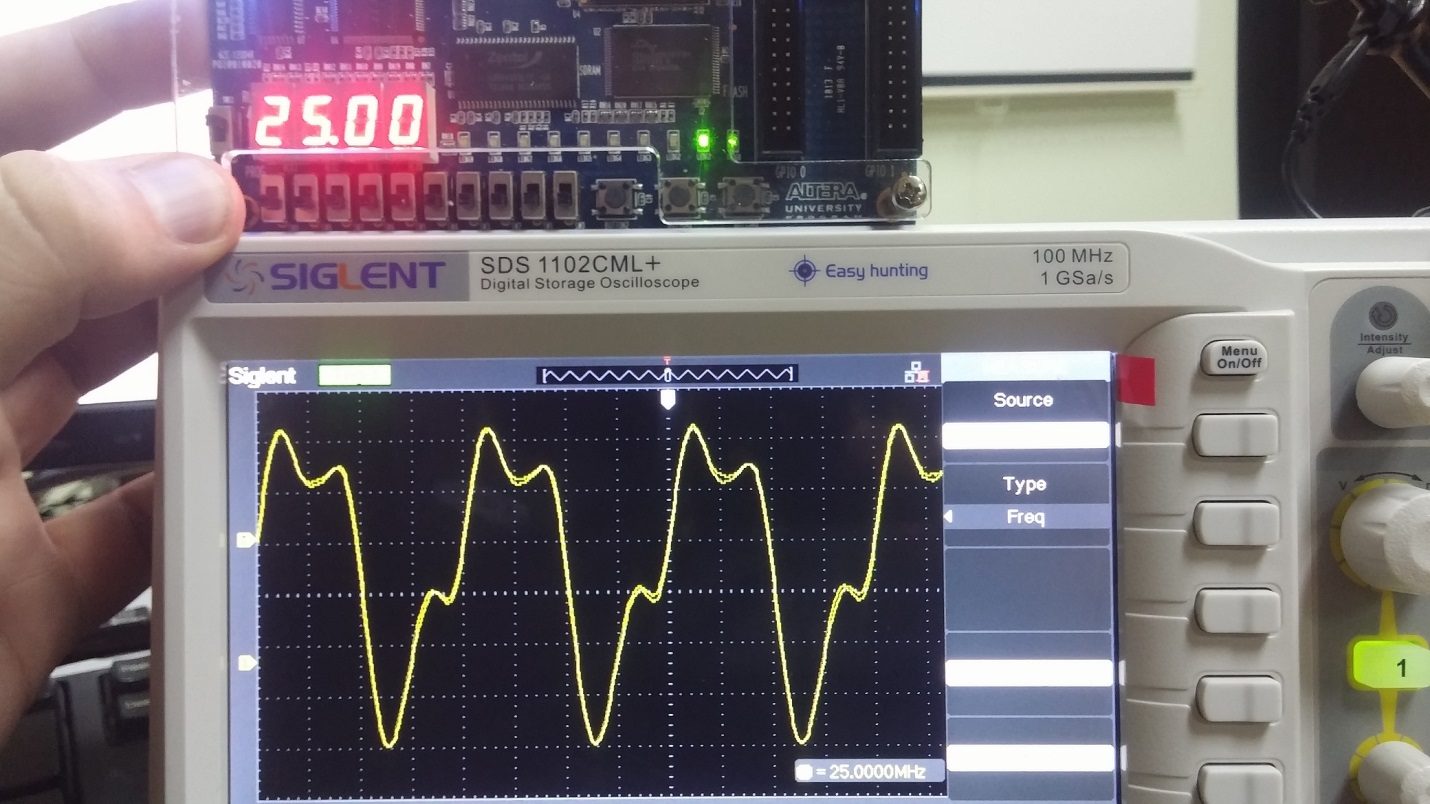
**Work goal:** using DE0 development board generate periodical signal showing frequency using HEX displays and allowing to change said frequency with switches.

Algorithms used in tasks.



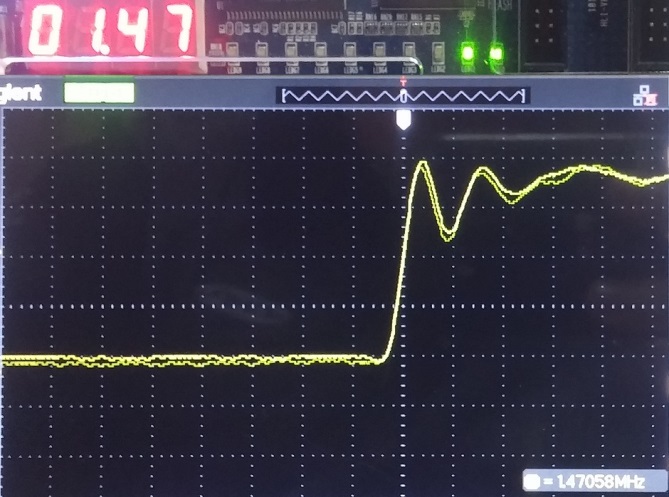
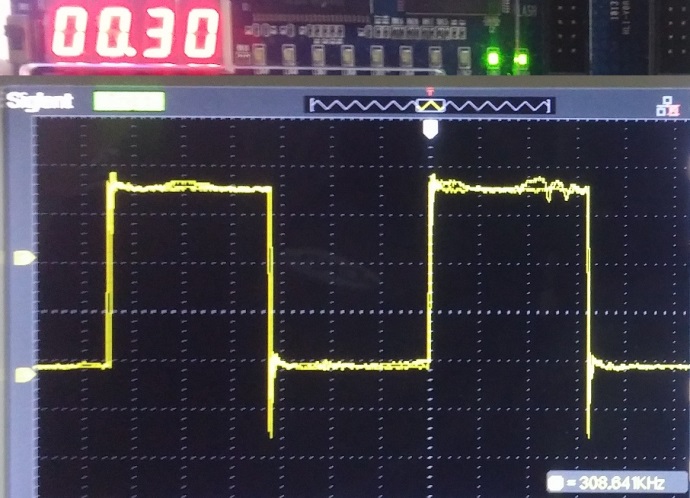
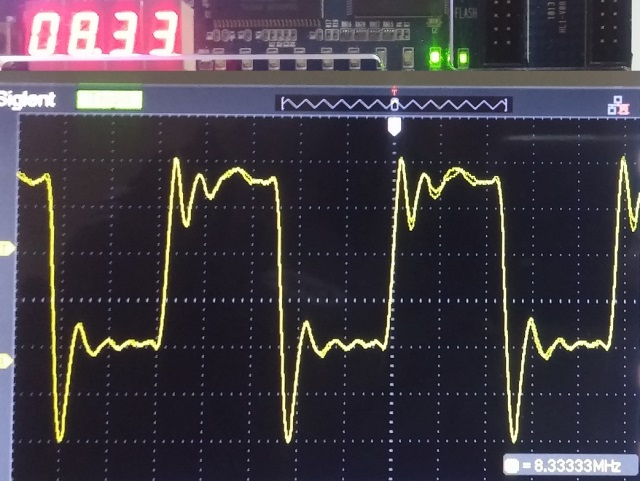
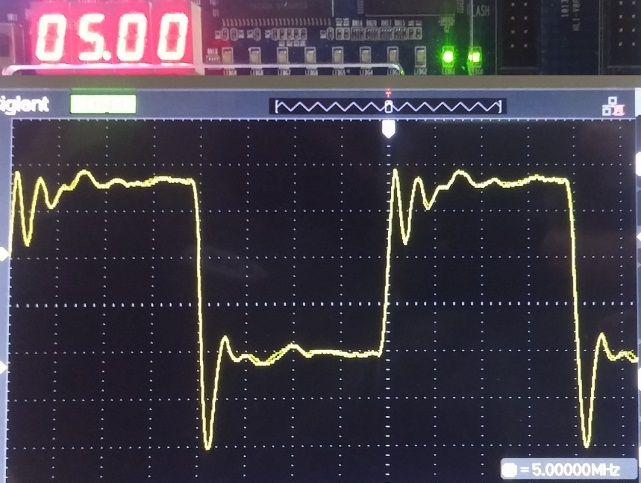
**Fig. 1.** “Algorithm” for signal generation.

Results analysis.



**Fig. 2.** Signal generation without any frequency division.

All in all it is simple operation. First on each clock rising edge a counter is increased and that value is tested against a threshold value *x* and, if counter is greater than *x*, then it is reset and another counter, *y*, is increased. Threshold value *x* is calculated from the switches – switches representing a binary number. 10 switches give *x* value ranging from 0 to 1023. In practice, *x* is on or off cycle count, since the second counter is used to determine the output state – output high when counter even, output low when counter odd. This means that it is effectively a frequency divider. In same cycle frequency for display is also determined, by a simple equation. HEX displays are directly tied to that frequency value. Examples of operation are shown in Figure 3 and can also be viewed on YouTube at https://youtu.be/Wulkias8waE



**Fig. 3.** Few examples of signal generation using the divider.

Conclusions.

Well, it works and is quite precise. The 4 HEX displays can’t provide much precision when indicating target frequency in some cases, but does point to the right direction. By the way the signal is a square wave with maximum frequency of 25 MHz. And in this particular setup minimum frequency is 48 kHz.

Source code.

001 library ieee;

002 use ieee.std\_logic\_1164.all;

003 use ieee.numeric\_std.all;

004

005 entity IESigGen is

006 Port (

007 CLOCK\_50 : IN STD\_LOGIC;

008 SW : IN STD\_LOGIC\_VECTOR(9 DOWNTO 0);

009 KEY : IN STD\_LOGIC\_VECTOR(2 DOWNTO 0);

010 HEX0 : OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0);

011 HEX1 : OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0);

012 HEX2 : OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0);

013 HEX3 : OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0);

014 LEDG : OUT STD\_LOGIC\_VECTOR(9 downto 0);

015 GPIO\_0 : OUT STD\_LOGIC\_VECTOR(2 DOWNTO 0)

016 );

017 end IESigGen;

018

019 architecture Whatever of IESigGen is

020

021 signal timer\_count : INTEGER := 0;

022 signal counter : INTEGER := 0;

023 SIGNAL FREQUENCY : INTEGER := 0;

024 signal INTERVAL : INTEGER := 0;

025

026 begin

027

028 WITH FREQUENCY mod 10 SELECT HEX0 <=

029 "11000000" WHEN 0,

030 "11111001" WHEN 1,

031 "10100100" WHEN 2,

032 "10110000" WHEN 3,

033 "10011001" WHEN 4,

034 "10010010" WHEN 5,

035 "10000010" WHEN 6,

036 "11111000" WHEN 7,

037 "10000000" WHEN 8,

038 "10010000" WHEN 9,

039 "01111111" WHEN OTHERS;

040

041 WITH (FREQUENCY / 10) mod 10 SELECT HEX1 <=

042 "11000000" WHEN 0,

043 "11111001" WHEN 1,

044 "10100100" WHEN 2,

045 "10110000" WHEN 3,

046 "10011001" WHEN 4,

047 "10010010" WHEN 5,

048 "10000010" WHEN 6,

049 "11111000" WHEN 7,

050 "10000000" WHEN 8,

051 "10010000" WHEN 9,

052 "01111111" WHEN OTHERS;

053

054 WITH (FREQUENCY / 100) mod 10 SELECT HEX2 <=

055 "01000000" WHEN 0,

056 "01111001" WHEN 1,

057 "00100100" WHEN 2,

058 "00110000" WHEN 3,

059 "00011001" WHEN 4,

060 "00010010" WHEN 5,

061 "00000010" WHEN 6,

062 "01111000" WHEN 7,

063 "00000000" WHEN 8,

064 "00010000" WHEN 9,

065 "01111111" WHEN OTHERS;

066

067 WITH (FREQUENCY / 1000) mod 10 SELECT HEX3 <=

068 "11000000" WHEN 0,

069 "11111001" WHEN 1,

070 "10100100" WHEN 2,

071 "10110000" WHEN 3,

072 "10011001" WHEN 4,

073 "10010010" WHEN 5,

074 "10000010" WHEN 6,

075 "11111000" WHEN 7,

076 "10000000" WHEN 8,

077 "10010000" WHEN 9,

078 "01111111" WHEN OTHERS;

079

080 WITH COUNTER MOD 2 SELECT GPIO\_0 <=

081 "001" WHEN 0,

082 "000" WHEN OTHERS;

083

084 process(CLOCK\_50)

085 begin

086 if rising\_edge(CLOCK\_50) then

087 timer\_count <= timer\_count + 1;

088 if (timer\_count >= INTERVAL) then

089 counter <= counter + 1;

090 timer\_count <= 0;

091 end if;

092

093 INTERVAL <= TO\_INTEGER(UNSIGNED(SW));

094 FREQUENCY <= 5000/((INTERVAL+1)\*2);

095

096 end if;

097 end process;

098

099 end Whatever;

Electrical schemes:

