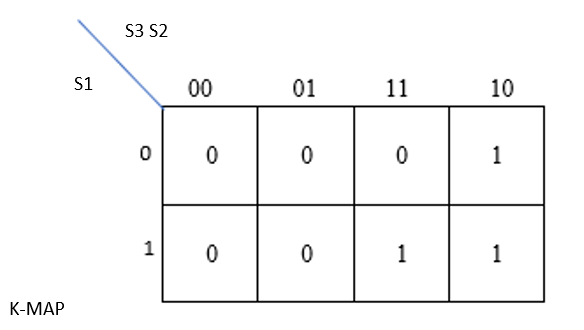
* **Design Methodology**

I establish this labs purpose with the scenario that I used in Lab02. In the scenario there is a Supervisor (S1), Rookie(S2), Cashier(S3) and the goal is getting inside the bank vault.

Only the supervisor has an access the bank vault so if someone else wants to get in the vault they must be together with the supervisor. Also, rookie always must be with the cashier in order to be monitored. If rookie is in the vault cashier must be in the vault as well as the supervisor. At the seven-segment display S1 S2 S3 L4 will be displayed with ‘1’ s and ‘0’ s according to their values which means if the value is 0 then

LED\_on ="0000001", if it is 1 then LED\_on = "1001111”.

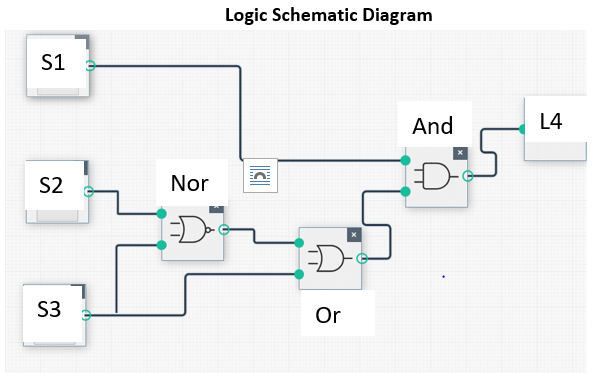
* **Differences from Preliminary:**
* I added clock module and I am using it to display 4 different seven-segments at the same time on Basys3
* About Activator and LED\_bin I changed the locations and the way I used them in my code.
* I added 4 LEDs to visualize inputs and outputs values.
* **Results**



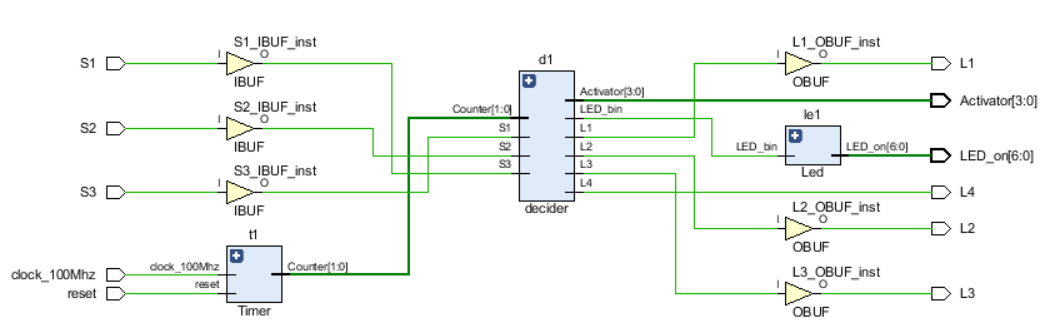
Truth Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S | C | R | H(S,C,R) | Minterms |
| 0 | 0 | 0 | 0 |  |
| 0 | 0 | 1 | 0 |  |
| 0 | 1 | 0 | 0 |  |
| 0 | 1 | 1 | 0 |  |
| 1 | 0 | 0 | 1 |  |
| 1 | 0 | 1 | 0 |  |
| 1 | 1 | 0 | 1 |  |
| 1 | 1 | 1 | 1 |  |

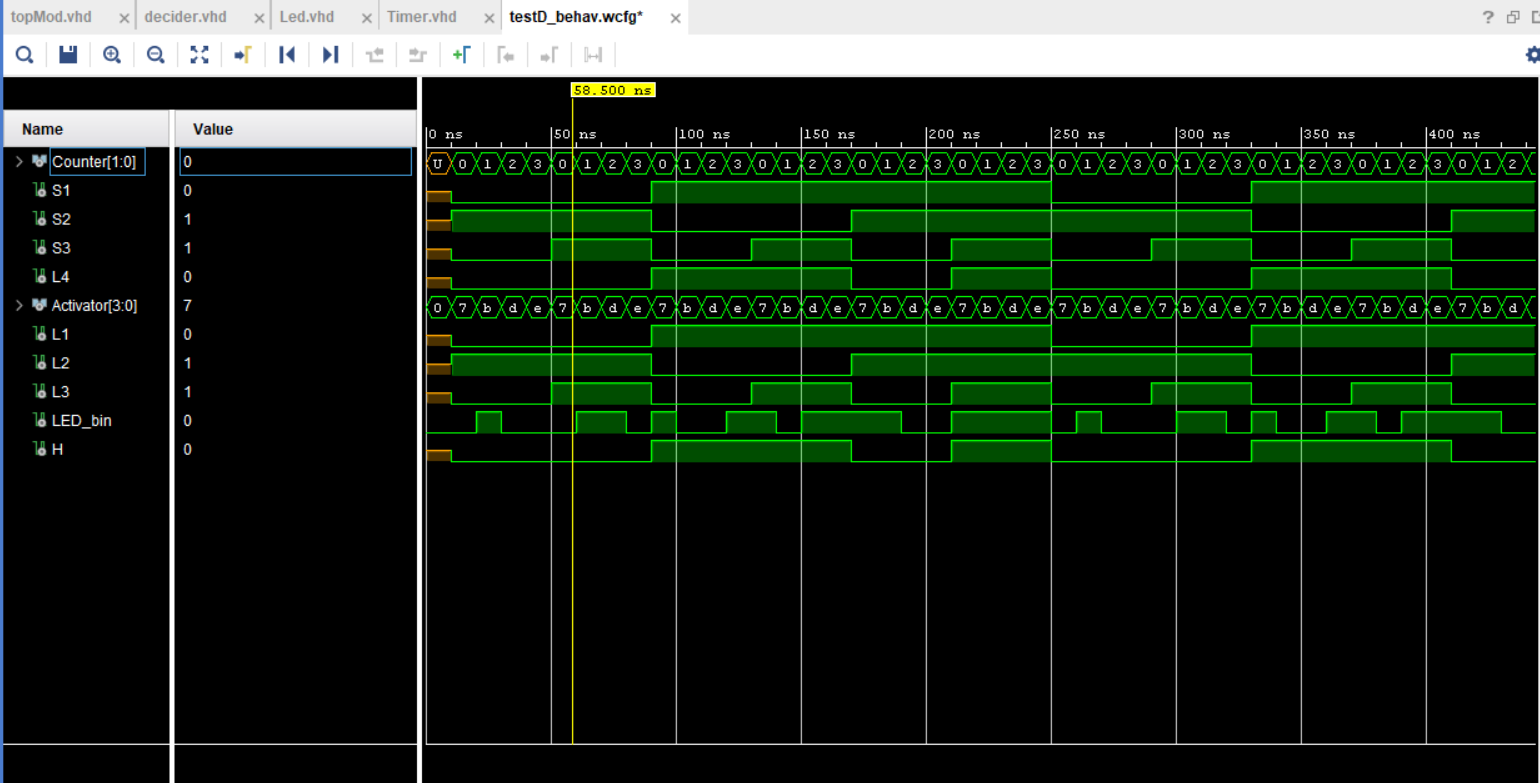
Sum of products( minterms)

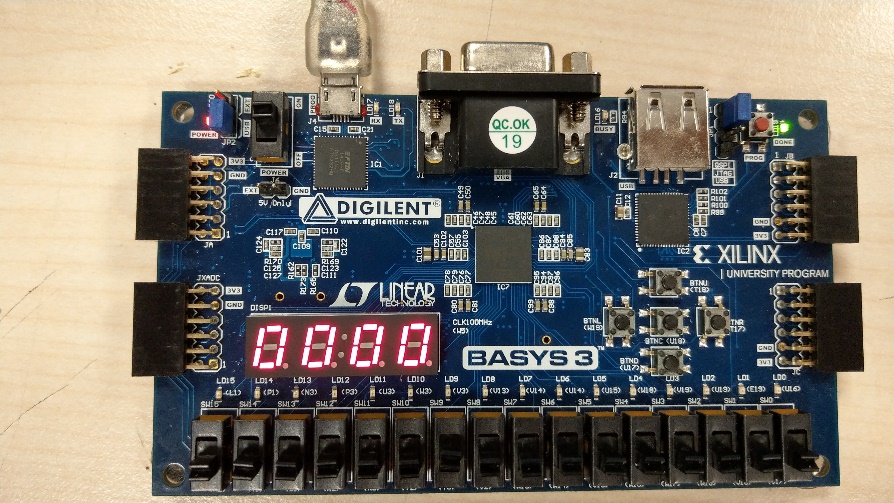


**Schematic**

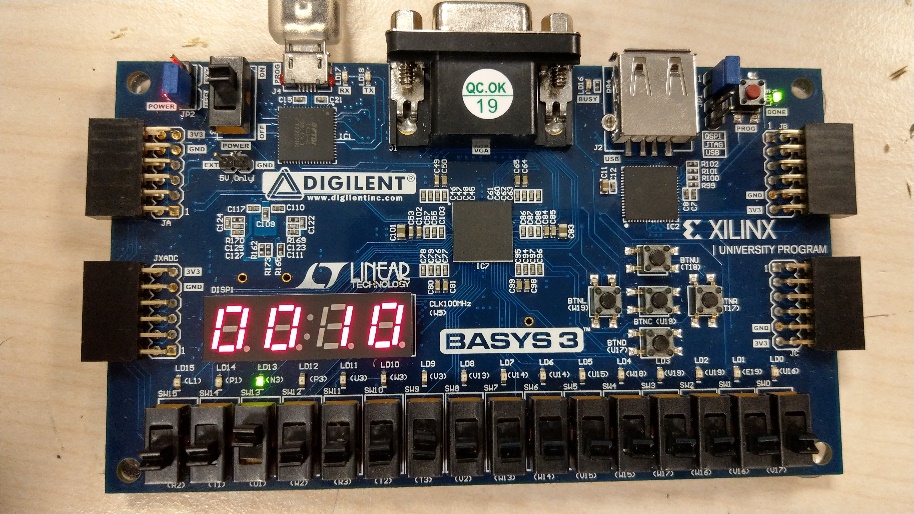


**Test bench of decider**

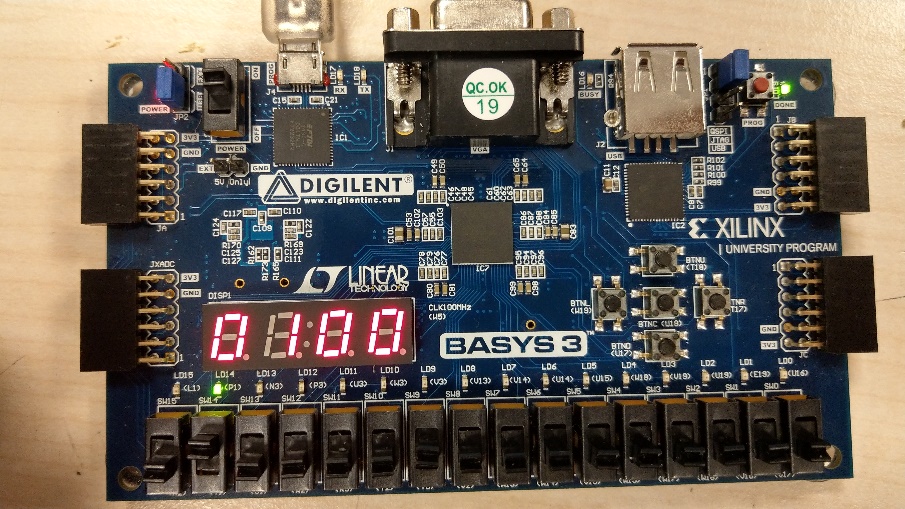




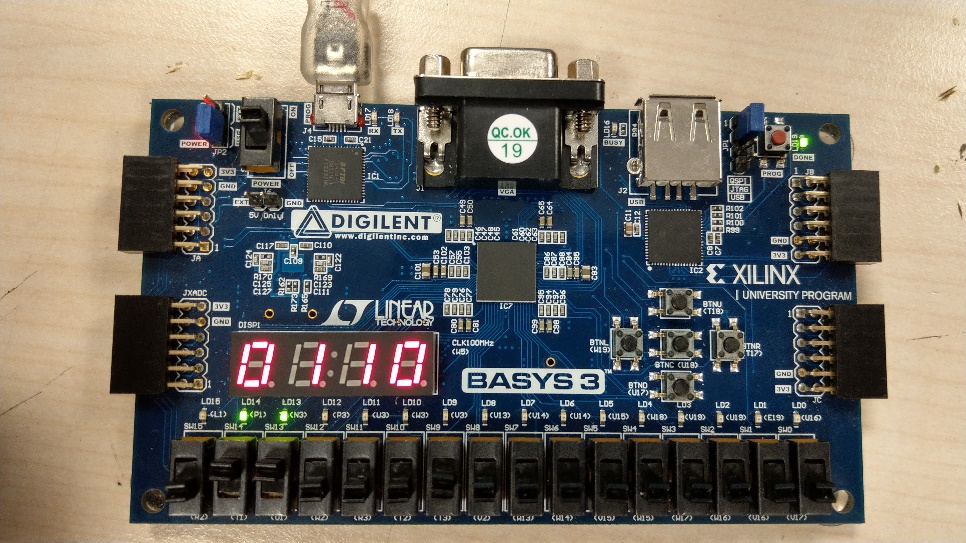
S1:0 S2:0 S3:0 L4:0



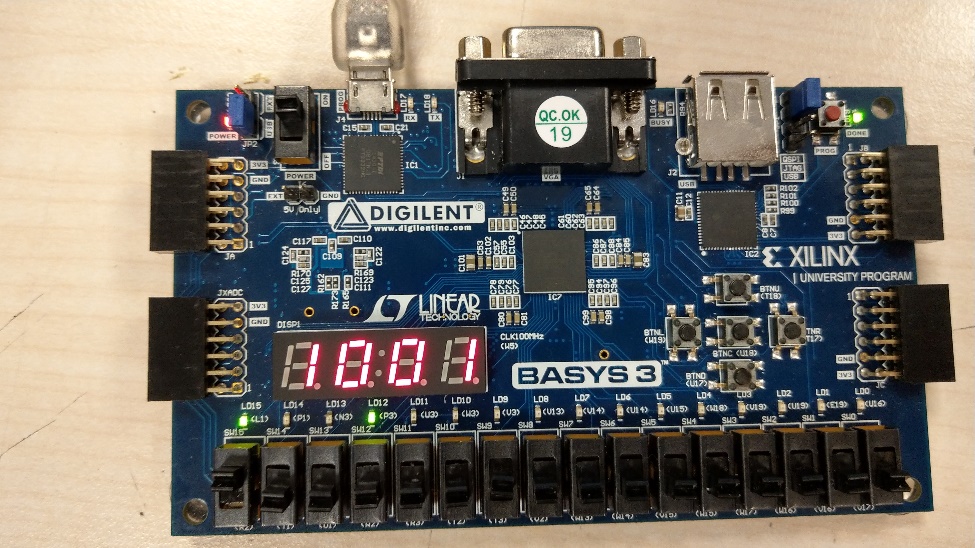
S1:0 S2:0 S3:1 L4:0



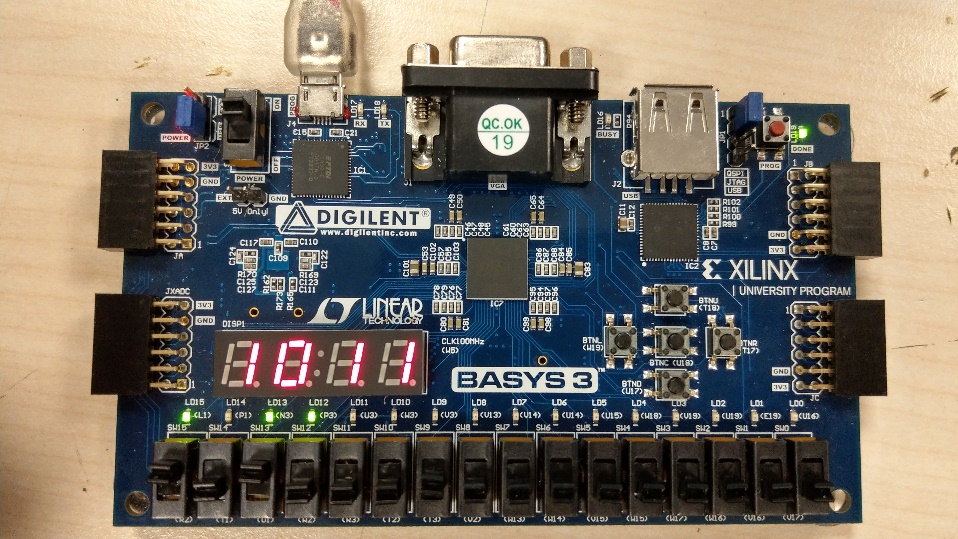
S1:0 S2:1 S3:0 L4:0



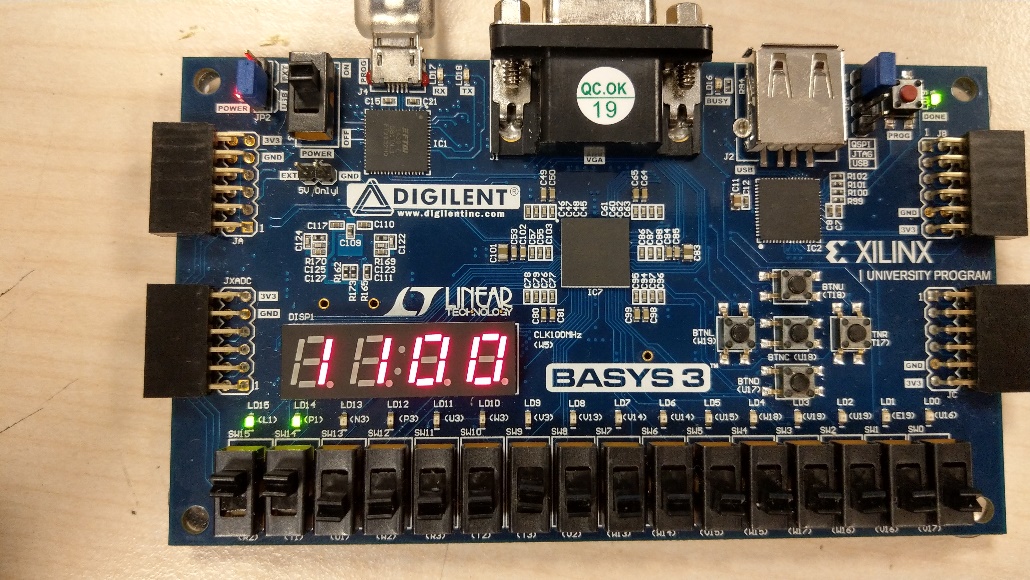
S1:0 S2:1 S3:1 L4:0



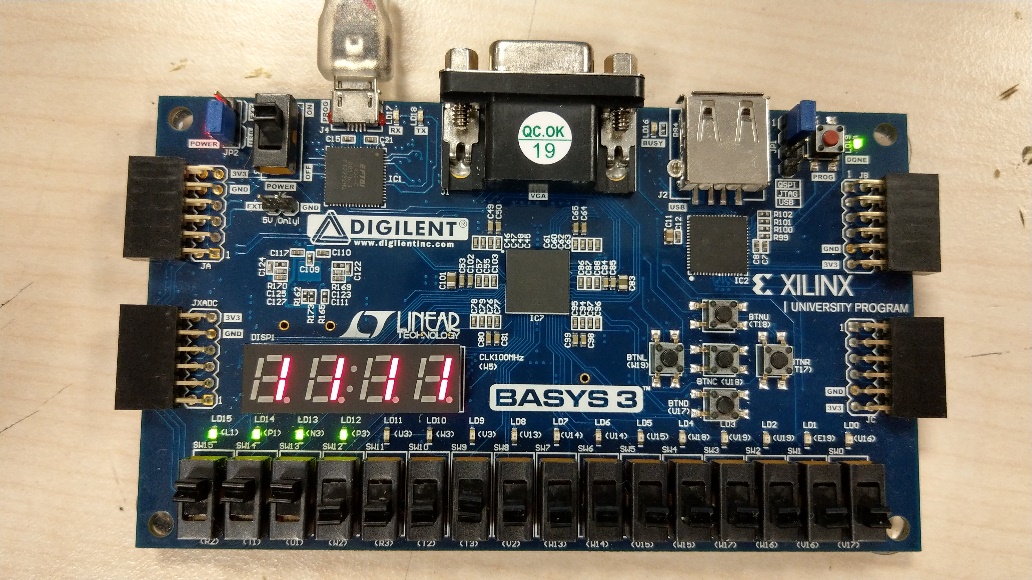
S1:1 S2:0 S3:0 L4:1



S1:1 S2:0 S3:1 L4:1



S1:1 S2:1 S3:0 L4:0



S1:1 S2:1 S3:1 L4:1

* **Conclusion**

I have learnt the differences between variable types like in, out, inout, signals… and places where can we use them, or which one must use. I have faced with some errors like “usf-xsim-62 'elaborate' step failed with error(s)” and I learnt to track the log files and find the errors. I have learnt whether I included all possible values I should also add “others” case in cases. I have learnt how to use top and sub modules and how to write test bench in better way. I also have learnt how to use seven-segment display in which how to display numbers and letters, how to order them and showing four seven segments at the same time using clock on the Basys3.

* **Appendices**
* VHDL:

TopModule:

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity topMod is

Port ( clock\_100Mhz : in STD\_LOGIC;

reset : in STD\_LOGIC;

S3,S2,S1 : in STD\_LOGIC;

L4,L3,L2,L1 : out STD\_LOGIC;

Activator : out STD\_LOGIC\_VECTOR (3 downto 0);

LED\_on : out STD\_LOGIC\_VECTOR (6 downto 0));

end topMod;

architecture Behavioral of topMod is

signal Counter\_temp: std\_logic\_vector(1 downto 0);

signal BIN\_LED: std\_logic;

component decider is

PORT

(

Counter : in std\_logic\_vector(1 downto 0);

S3,S2,S1 : in STD\_LOGIC;

Activator : out std\_logic\_vector(3 downto 0);

L4,L3,L2,L1 : out STD\_LOGIC;

LED\_bin : out STD\_LOGIC);

end component;

component Led is

PORT

(LED\_bin : in STD\_LOGIC;

LED\_on : out STD\_LOGIC\_VECTOR (6 downto 0));

end component;

component Timer is

PORT

(clock\_100Mhz : in STD\_LOGIC;

reset : in STD\_LOGIC;

Counter: out std\_logic\_vector(1 downto 0));

end component;

begin

d1 : decider PORT MAP (Counter\_temp,S1,S2,S3,Activator,L4,L3,L2,L1,BIN\_LED);

le1: Led PORT MAP (BIN\_LED,LED\_on);

t1 : Timer PORT MAP ( clock\_100Mhz,reset,Counter\_temp);

end Behavioral;

Decider module:

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity decider is

Port ( Counter : in std\_logic\_vector(1 downto 0);

S1 : in STD\_LOGIC;

S2 : in STD\_LOGIC;

S3 : in STD\_LOGIC;

Activator : out std\_logic\_vector(3 downto 0);

L1 : out STD\_LOGIC;

L2 : out STD\_LOGIC;

L3 : out STD\_LOGIC;

L4 : out STD\_LOGIC;

LED\_bin : out STD\_LOGIC);

end decider;

architecture Behavioral of decider is

signal H: STD\_LOGIC;

begin

H <= S1 and (( not S2 and not S3) or S3);

L1 <= S1;

L2 <= S2;

L3 <= S3;

L4 <= H;

process(Counter,S1)

begin

case Counter is

when "00" =>

Activator <= "0111";

LED\_bin <= S1;

when "01" =>

Activator <= "1011";

LED\_bin <= S2;

when "10" =>

Activator <= "1101";

LED\_bin <= S3;

when "11" =>

Activator <= "1110";

LED\_bin <= H;

when others =>

Activator <= "0000";

LED\_bin <= '0';

end case;

end process;

end Behavioral;

Led Module:

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Led is

Port ( LED\_bin : in STD\_LOGIC;

LED\_on : out STD\_LOGIC\_VECTOR (6 downto 0));

end Led;

architecture Behavioral of Led is

begin

process(LED\_bin)

begin

case LED\_bin is

when '0' => LED\_on <= "0000001";

when '1' => LED\_on <= "1001111";

when others => LED\_on <= "0000001";

end case;

end process;

end Behavioral;

Timer Module:

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.std\_logic\_unsigned.all;

entity Timer is

Port ( clock\_100Mhz : in STD\_LOGIC;

reset : in STD\_LOGIC;

Counter: out std\_logic\_vector(1 downto 0));

end Timer;

architecture Behavioral of Timer is

signal refresh\_counter: STD\_LOGIC\_VECTOR (19 downto 0);

begin

process(clock\_100Mhz,reset)

begin

if(reset='1') then

refresh\_counter <= (others => '0');

elsif(rising\_edge(clock\_100Mhz)) then

refresh\_counter <= refresh\_counter + 1;

end if;

end process;

Counter <= refresh\_counter(19 downto 18);

end Behavioral;

TESTBENCHES:

TESTD:

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity testD is

end testD;

architecture Behavioral of testD is

component decider is

Port ( Counter : in std\_logic\_vector(1 downto 0);

S1 : in STD\_LOGIC;

S2 : in STD\_LOGIC;

S3 : in STD\_LOGIC;

Activator : out std\_logic\_vector(3 downto 0);

L1 : out STD\_LOGIC;

L2 : out STD\_LOGIC;

L3 : out STD\_LOGIC;

L4:out STD\_LOGIC;

LED\_bin : out STD\_LOGIC

);

end component;

signal Counter : std\_logic\_vector(1 downto 0);

signal S1 : STD\_LOGIC;

signal S2 : STD\_LOGIC;

signal S3 : STD\_LOGIC;

signal L4 : STD\_LOGIC;

signal Activator : std\_logic\_vector(3 downto 0) := "0000";

signal L1 : STD\_LOGIC;

signal L2 : STD\_LOGIC;

signal L3 : STD\_LOGIC;

signal LED\_bin : STD\_LOGIC := '0';

begin

UUT: decider PORT MAP(

Counter => Counter,

S1 => S1,

S2 => S2,

S3 => S3,

L4 => L4,

Activator => Activator,

L1 => L1,

L2 => L2,

L3 => L3,

LED\_bin => LED\_bin

--Activator => Activator

);

Bench1: PROCESS

begin

--wait for 100 ns;

--Counter <= "00";

--S1 <= '0';

--S2 <= '0';

--S3 <= '0';

--wait for 100 ns;

--Counter <= "01";

--S1 <= '0';

--S2 <= '0';

--S3 <= '0';

--wait for 100 ns;

--Counter <= "10";

--S1 <= '0';

--S2 <= '0';

--S3 <= '0';wait for 100 ns;

--Counter <= "10";

--S1 <= '0';

--S2 <= '0';

--S3 <= '0';

----------------

--wait for 100 ns;

--Counter <= "00";

--S1 <= '0';

--S2 <= '0';

--S3 <= '1';

--wait for 100 ns;

--Counter <= "01";

--S1 <= '0';

--S2 <= '0';

--S3 <= '1';

--wait for 100 ns;

--Counter <= "10";

--S1 <= '0';

--S2 <= '0';

--S3 <= '1';

--wait for 100 ns;

--Counter <= "11";

--S1 <= '0';

--S2 <= '0';

--S3 <= '1';

-------------

wait for 10 ns;

Counter <= "00";

S1 <= '0';

S2 <= '1';

S3 <= '0';

wait for 10 ns;

Counter <= "01";

S1 <= '0';

S2 <= '1';

S3 <= '0';

wait for 10 ns;

Counter <= "10";

S1 <= '0';

S2 <= '1';

S3 <= '0';

wait for 10 ns;

Counter <= "11";

S1 <= '0';

S2 <= '1';

S3 <= '0';

-------------

wait for 10 ns;

Counter <= "00";

S1 <= '0';

S2 <= '1';

S3 <= '1';

wait for 10 ns;

Counter <= "01";

S1 <= '0';

S2 <= '1';

S3 <= '1';

wait for 10 ns;

Counter <= "10";

S1 <= '0';

S2 <= '1';

S3 <= '1';

wait for 10 ns;

Counter <= "11";

S1 <= '0';

S2 <= '1';

S3 <= '1';

------------

wait for 10 ns;

Counter <= "00";

S1 <= '1';

S2 <= '0';

S3 <= '0';

wait for 10 ns;

Counter <= "01";

S1 <= '1';

S2 <= '0';

S3 <= '0';

wait for 10 ns;

Counter <= "10";

S1 <= '1';

S2 <= '0';

S3 <= '0';

wait for 10 ns;

Counter <= "11";

S1 <= '1';

S2 <= '0';

S3 <= '0';

--------------

wait for 10 ns;

Counter <= "00";

S1 <= '1';

S2 <= '0';

S3 <= '1';

wait for 10 ns;

Counter <= "01";

S1 <= '1';

S2 <= '0';

S3 <= '1';

wait for 10 ns;

Counter <= "10";

S1 <= '1';

S2 <= '0';

S3 <= '1';

wait for 10 ns;

Counter <= "11";

S1 <= '1';

S2 <= '0';

S3 <= '1';

--------------------

wait for 10 ns;

Counter <= "00";

S1 <= '1';

S2 <= '1';

S3 <= '0';

wait for 10 ns;

Counter <= "01";

S1 <= '1';

S2 <= '1';

S3 <= '0';

wait for 10 ns;

Counter <= "10";

S1 <= '1';

S2 <= '1';

S3 <= '0';

wait for 10 ns;

Counter <= "11";

S1 <= '1';

S2 <= '1';

S3 <= '0';

-----------------

wait for 10 ns;

Counter <= "00";

S1 <= '1';

S2 <= '1';

S3 <= '1';

wait for 10 ns;

Counter <= "01";

S1 <= '1';

S2 <= '1';

S3 <= '1';

wait for 10 ns;

Counter <= "10";

S1 <= '1';

S2 <= '1';

S3 <= '1';

wait for 10 ns;

Counter <= "11";

S1 <= '1';

S2 <= '1';

S3 <= '1';

End process;

end Behavioral;

TESTLED:

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity testLed is

end testLed;

architecture Behavioral of testLed is

component Led is

Port (

LED\_bin : in STD\_LOGIC;

LED\_on : out STD\_LOGIC\_VECTOR (6 downto 0));

end component;

signal LED\_bin : STD\_LOGIC;

signal LED\_on : STD\_LOGIC\_VECTOR (6 downto 0):= b"0000000" ;

begin

UUT: Led PORT MAP(

LED\_bin => LED\_bin,

LED\_on=> LED\_on);

Bench1: PROCESS

begin

wait for 10 ns;

LED\_bin<='0';

wait for 10 ns;

LED\_bin<='1';

end PROCESS;

end Behavioral;

TEST\_TIME:

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity testTime is

end testTime;

architecture Behavioral of testTime is

component Timer is

Port ( clock\_100Mhz : in STD\_LOGIC;

reset : in STD\_LOGIC;

Counter: out std\_logic\_vector(1 downto 0));

end component;

signal clock\_100Mhz : STD\_LOGIC;

signal reset : STD\_LOGIC;

signal Counter: std\_logic\_vector(1 downto 0);

begin

UUT: Timer PORT MAP(

clock\_100Mhz => clock\_100Mhz,

reset => reset,

Counter => Counter

);

Bench2: PROCESS

begin

wait for 50 ns;

clock\_100Mhz <= '0';

reset <= '0';

wait for 50 ns;

clock\_100Mhz <= '0';

reset <= '1';

wait for 50 ns;

clock\_100Mhz <= '1';

reset <= '0';

wait for 50 ns;

clock\_100Mhz <= '1';

reset <= '1';

wait for 50 ns;

clock\_100Mhz <= '0';

reset <= '0';

end PROCESS;

end Behavioral;