Mohammadreza Taraz and Tien Mai

Dr. Mihail Cutitaru

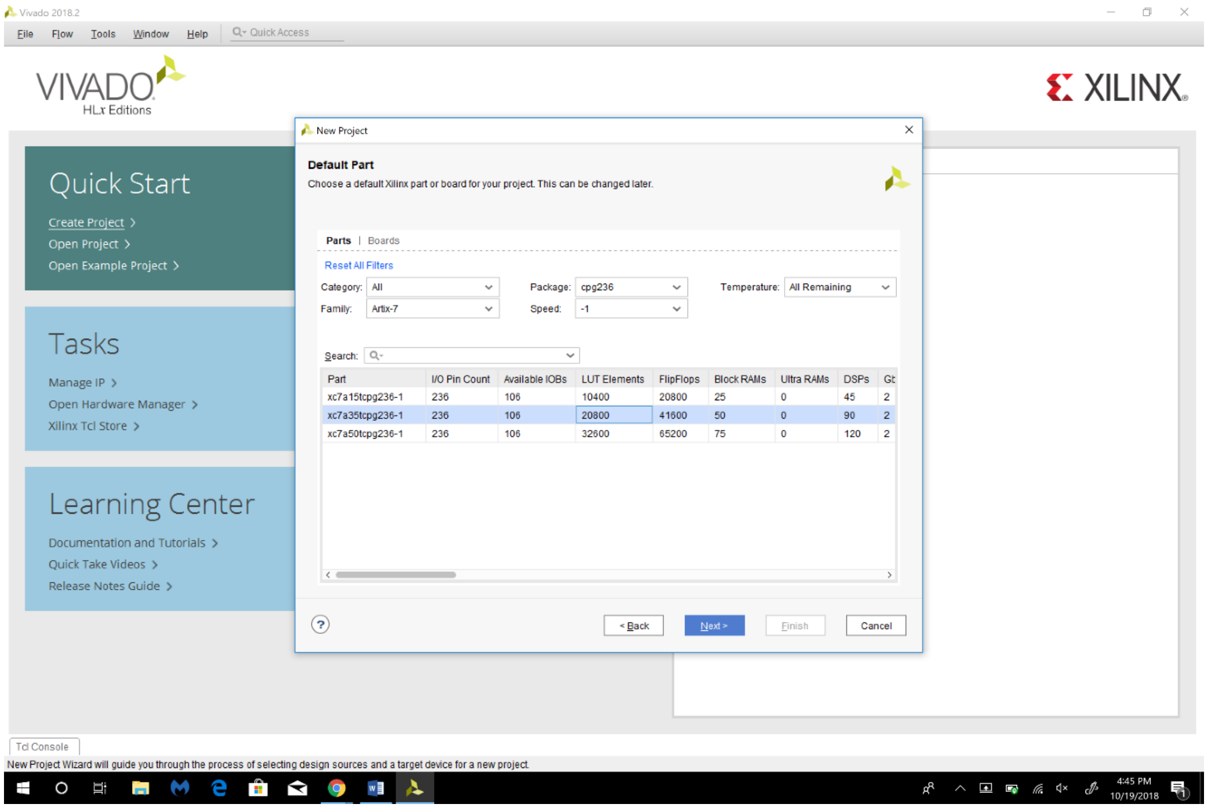
ECGR 2181-002

October 19, 2018

VHDL Report

We start by creating a directory, and me made it as Register Transfer Level (RTL) project. After we had to change the simulation language to VHDL.

here we changed Family: Artix-7, Package: CPG236, Speed: -1, and we choose the one with 41600 FlipFlops.



Default Part

Choose a default Xilinx part or board for your project. This can be changed later.

Category: All
Package: cpg236
Temperature: All Remaining
Family: Artix-7
Speed: -1

Part	I/O Pin Count	Available IOBs	LUT Elements	FlipFlops	Block RAMs	Ultra RAMs	DSPs	Gt
xc7a15tcpg236-1	236	106	10400	20800	25	0	45	2
xc7a35tcpg236-1	236	106	20800	41600	50	0	90	2
xc7a50tcpg236-1	236	106	32600	65200	75	0	120	2

New Project Wizard will guide you through the process of selecting design sources and a target device for a new project.

Create A, B, Cin as input for out project and outputs are Sum and Cout.

TASK 1:

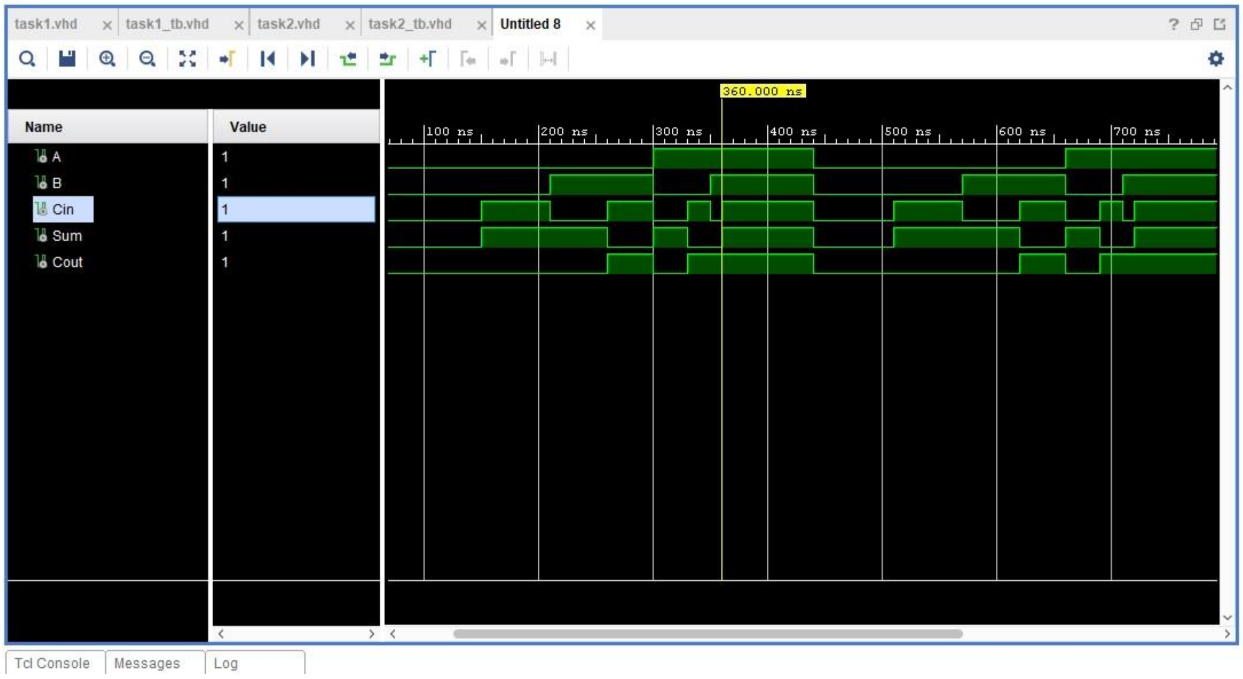
We Followed all the steps that was listed and that is our result for the first task.

Our code:

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.NUMERIC_STD.ALL;
```

```
entity task1 is
    Port ( A : in STD_LOGIC;
           B : in STD_LOGIC;
           Cin : in STD_LOGIC;
           Sum : out STD_LOGIC;
           Cout : out STD_LOGIC);
end task1;
architecture Behavioral of task1 is
begin
Sum <= A XOR B XOR Cin;
Cout <= (A AND B) OR (A AND Cin) OR (B AND Cin);

end Behavioral;
```

Result:



Name	Value
A	1
B	1
Cin	1
Sum	1
Cout	1

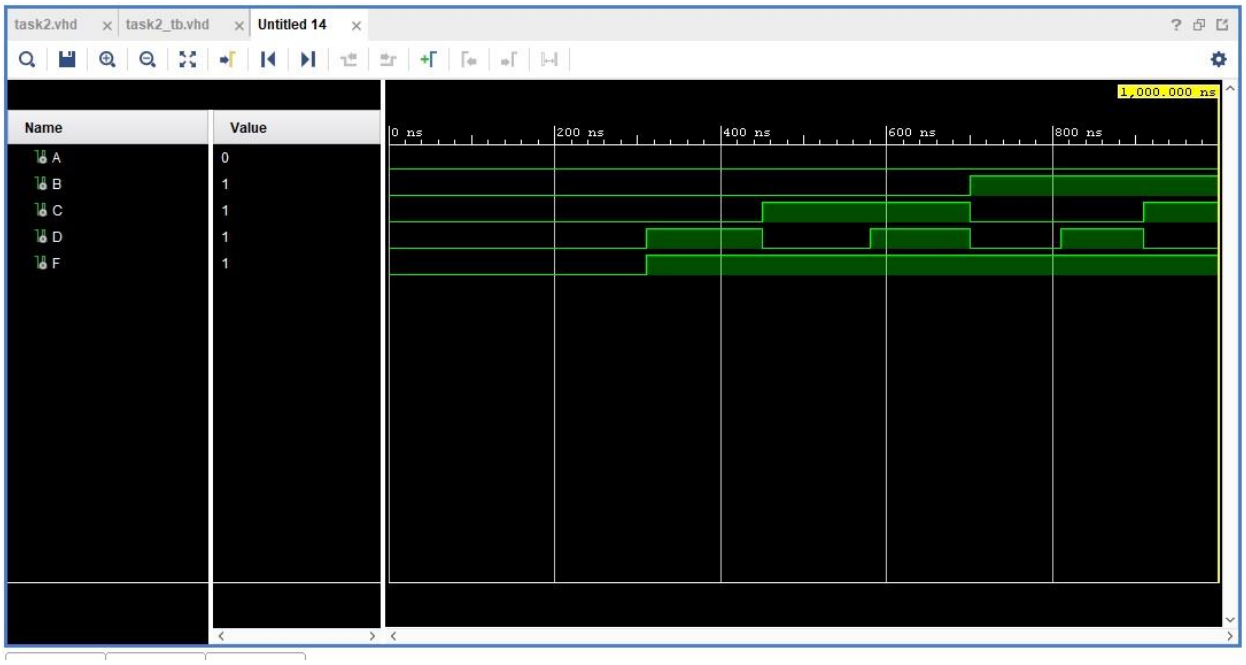
One problem we had was that we forgot to add code into table-bench and we had a hard time writing a table-bench.

PART 1 Task2: The problem we face was simplifying the implement function to the canonical SOP for the second part of the project.

Our code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity task2 is
    Port ( A : in STD_LOGIC;
           B : in STD_LOGIC;
           C : in STD_LOGIC;
           D : in STD_LOGIC;
           F : out STD_LOGIC);
end task2;
architecture Behavioral of task2 is
begin
F <= A OR B OR C OR D OR (A AND B AND C AND D) OR (A AND (NOT B) AND (NOT C) AND (NOT D));
end Behavioral;
```

Result:



Name	Value
A	0
B	1
C	1
D	1
F	1

PART 2:

In the second part of the project we write and simulate a hex to seven segment display converter. We filled up the truth table for it on the figure bellow wherever we have a segment of the value will be 1 and when it's on it will be 0.

in(3)	in(2)	in(1)	in(0)	[redacted]	A	B	C	D	E	F	G
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
0	0	0	0	[redacted]	1	1	1	1	1	1	0
0	0	0	1	[redacted]	0	1	1	0	0	0	0
0	0	1	0	[redacted]	1	1	0	1	1	0	1
0	0	1	1	[redacted]	1	1	1	1	0	0	1
0	1	0	0	[redacted]	0	1	1	0	0	1	1
0	1	0	1	[redacted]	1	0	1	1	0	1	1
0	1	1	0	[redacted]	1	0	1	1	1	1	1
0	1	1	1	[redacted]	1	1	1	0	0	0	0
1	0	0	0	[redacted]	1	1	1	1	1	1	1
1	0	0	1	[redacted]	1	1	1	1	0	1	1
1	0	1	0	[redacted]	1	1	1	0	1	1	1
1	0	1	1	[redacted]	0	0	1	1	1	1	1
1	1	0	0	[redacted]	1	0	0	1	1	1	0
1	1	0	1	[redacted]	0	1	1	1	1	0	1
1	1	1	0	[redacted]	1	0	0	1	1	1	1
1	1	1	1	[redacted]	1	0	0	0	1	1	1

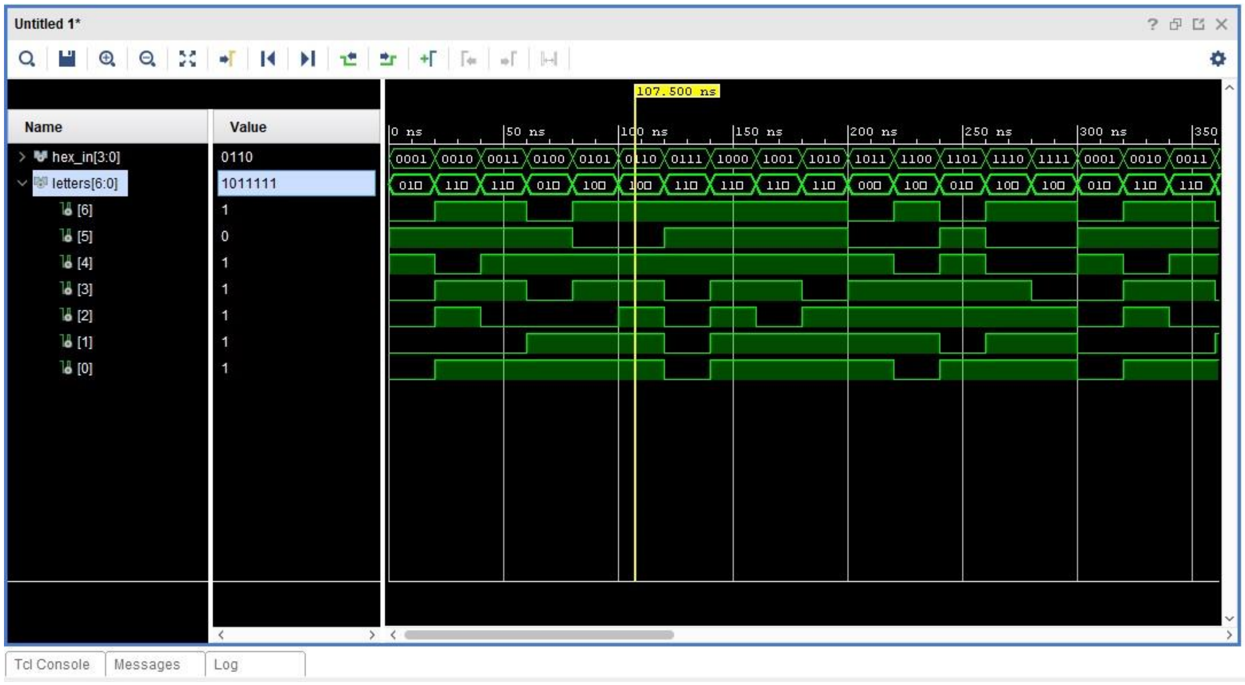
Table 1: Truth Table of the Encoder

Our code for PART 2:

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.NUMERIC_STD.ALL;
```

```
entity task3 is
   Port (
          hex_in : in STD_LOGIC_VECTOR (3 downto 0);
          letters : out STD_LOGIC_vector (6 downto 0)
          );
end task3;

architecture Behavioral of task3 is

signal temphex_in: integer;

begin

process (hex_in)

    begin

    temphex_in <= to_integer(unsigned(hex_in));

        case hex_in is
                   when "0000" => letters <= "0000001";
                   when "0001" => letters <= "1001111";
                   when "0010" => letters <= "0010010";
                   when "0011" => letters <= "0000110";
                   when "0100" => letters <= "1001100";
                   when "0101" => letters <= "0100100";
                   when "0110" => letters <= "0100000";
                   when "0111" => letters <= "0001111";
                   when "1000" => letters <= "0000000";
                   when "1001" => letters <= "0000100";
                   when "1010" => letters <= "0001000";
                   when "1011" => letters <= "1100000";
                   when "1100" => letters <= "0110001";
                   when "1101" => letters <= "1000010";
                   when "1110" => letters <= "0110000";
                   when others => letters <= "0111000";
         end case;
         end process;

end Behavioral;
```

Result:



Untitled 1*
107.500 ns
Name
Value
hex_in[3:0]
0110
letters[6:0]
1011111
[6] 1
[5] 0
[4] 1
[3] 1
[2] 1
[1] 1
[0] 1
0 ns
50 ns
100 ns
150 ns
200 ns
250 ns
300 ns
350
0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111 0001 0010 0011
Tcl Console
Messages
Log