FPGA Implementation of Digital Systems == Practical Activities ==

Lecturer PhD Eng Ionel BOSTAN
University of Pitesti
Romania

What is FPGA ?

– Actual trend in the implementation of digital systems is to use reconfigurable circuits like FPGA

“You can configure these chips to implement custom hardware functionality without ever having to pick up a breadboard or soldering iron”

– Advantage of FPGA (**Field-Programmable Gate Array**)

– **Large number of digital circuits** which can be interconnected by the end user by software means (using a configuration bitstream);

Bitstream contains information on how the components should be wired together .

– **Completely reconfigurable** - instantly take on a brand new “personality” when you recompile a different configuration of circuitry (a new bitstream);

– **Easy to use** – mature high-level design tools are available;

– **Low cost** do to the mass production;

Why we insist on teaching FPGA ?

Student Perspective:

- students have the feeling that they working in software;
- easy to develop and test new application (even without learn a hardware description language like Verilog or VHDL);
- increased chances of finding a god job;

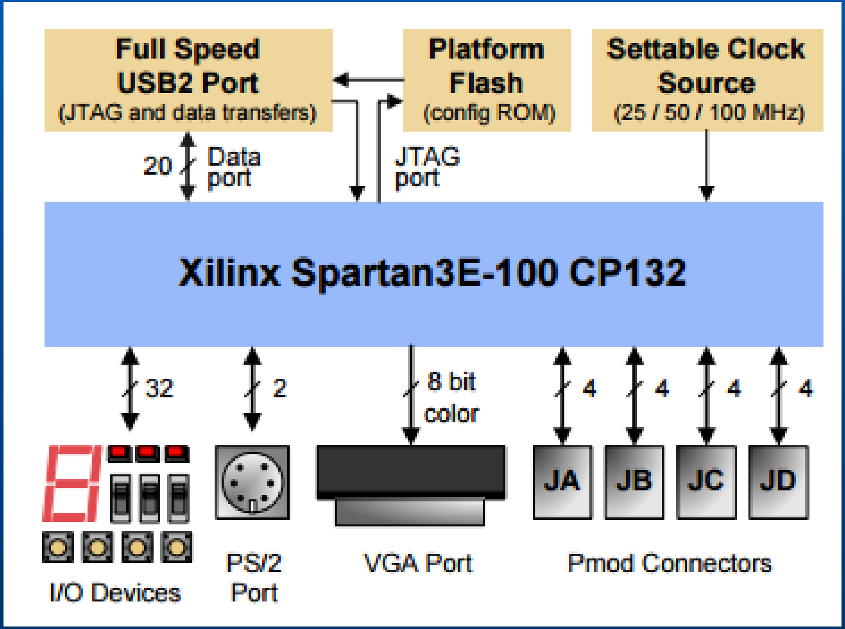
Technical advantage:

- Easy to use: "You can configure these chips to implement custom hardware functionality without ever having to pick up a breadboard or soldering iron" ;
- Large number of digital circuits which can be interconnected by the end user by software means
- Completely reconfigurable: in case of mistake in design, just fix your "logic function", re-compile and re-download it.
- **Low cost** do to the mass production;

1. Hardware support ; 2. Software support; 3. Applications

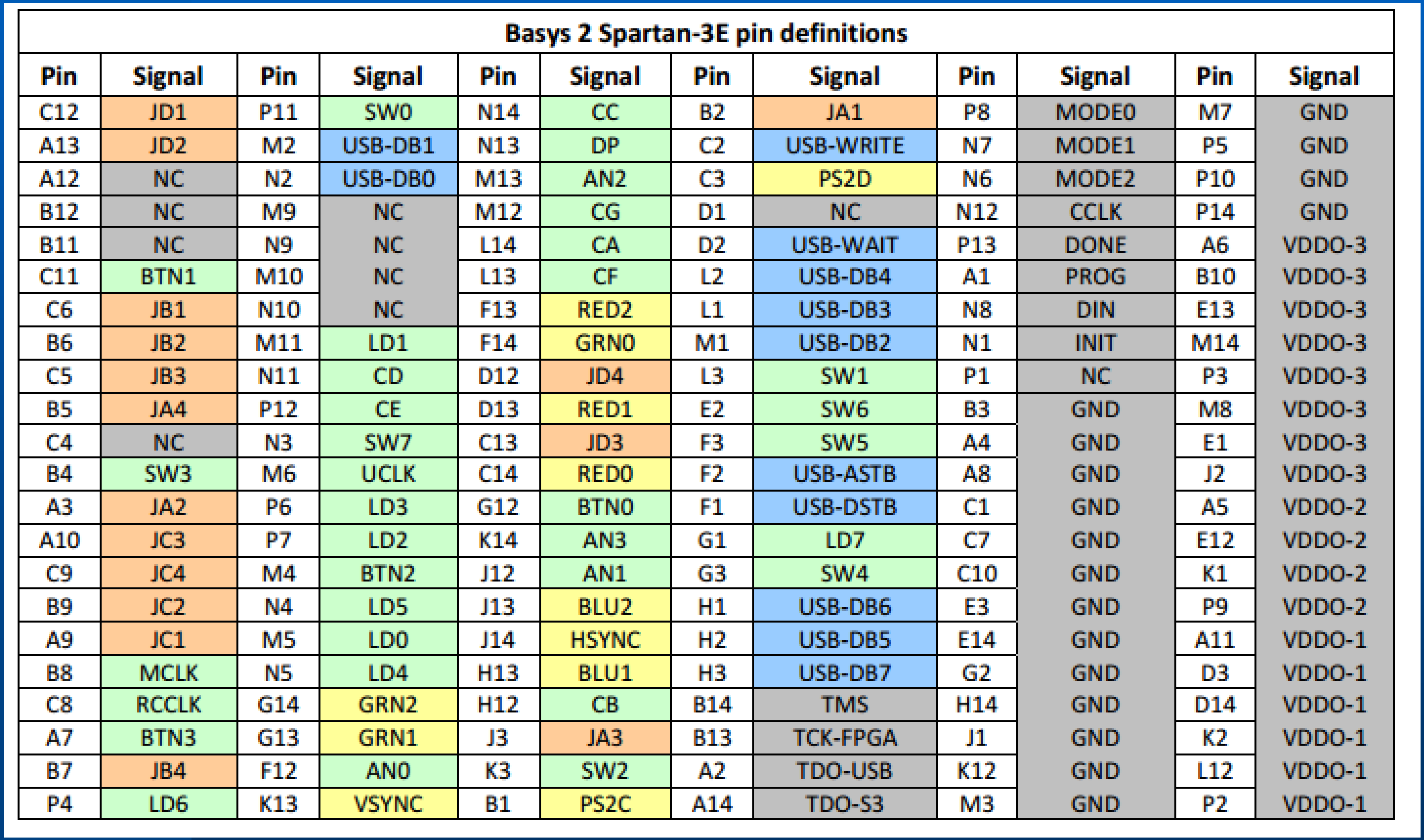
All practical activities will be tested using Basys 2 Board :

- **FPGA: Spartan 3E from Xilinx**
- **Free Software: ISEWebPack**
- **Low cost (69$ in academic program);**


Full Speed USB2 Port (JTAG and data transfers)
Platform Flash (config ROM)
Settable Clock Source (25 / 50 / 100 MHz)
20
Data port
JTAG port
Xilinx Spartan3E-100 CP132
32
2
8 bit color
4
4
4
4
JA
JB
JC
JD
I/O Devices
PS/2 Port
VGA Port
Pmod Connectors

1. Hardware support ; 2. Software support; 3. Applications

Basys 2 Spartan-3E Pin Definitions ;

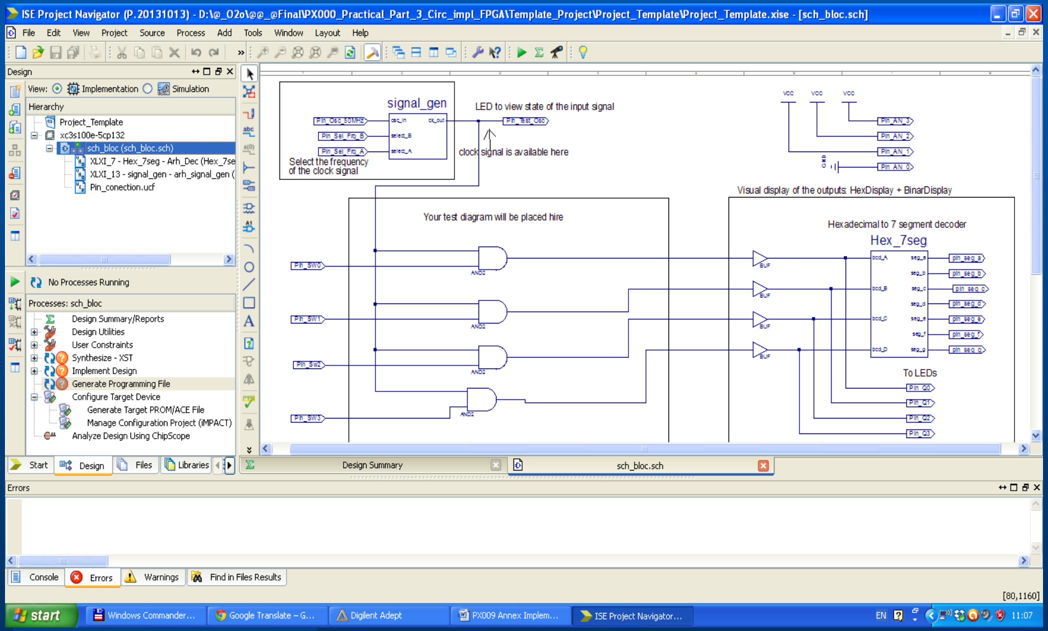


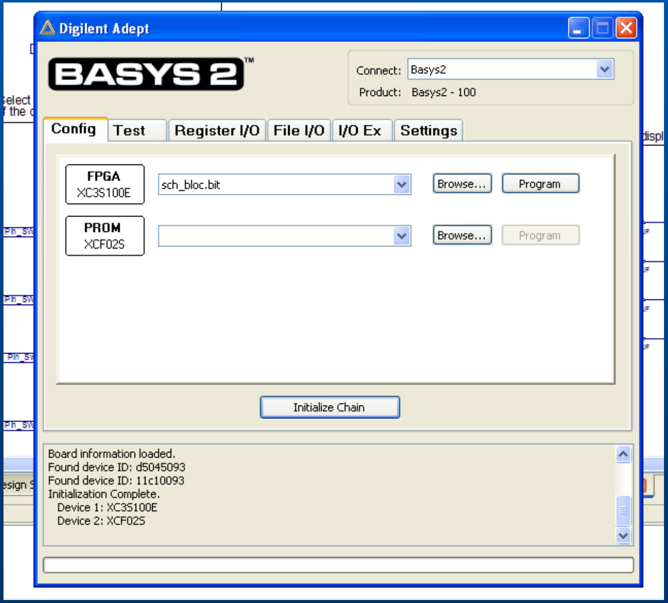
Basys 2 Spartan-3E pin definitions

Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
C12	JD1	P11	SW0	N14	CC	B2	JA1	P8	MODE0	M7	GND
A13	JD2	M2	USB-DB1	N13	DP	C2	USB-WRITE	N7	MODE1	P5	GND
A12	NC	N2	USB-DB0	M13	AN2	C3	PS2D	N6	MODE2	P10	GND
B12	NC	M9	NC	M12	CG	D1	NC	N12	CCLK	P14	GND
B11	NC	N9	NC	L14	CA	D2	USB-WAIT	P13	DONE	A6	VDDO-3
C11	BTN1	M10	NC	L13	CF	L2	USB-DB4	A1	PROG	B10	VDDO-3
C6	JB1	N10	NC	F13	RED2	L1	USB-DB3	N8	DIN	E13	VDDO-3
B6	JB2	M11	LD1	F14	GRN0	M1	USB-DB2	N1	INIT	M14	VDDO-3
C5	JB3	N11	CD	D12	JD4	L3	SW1	P1	NC	P3	VDDO-3
B5	JA4	P12	CE	D13	RED1	E2	SW6	B3	GND	M8	VDDO-3
C4	NC	N3	SW7	C13	JD3	F3	SW5	A4	GND	E1	VDDO-3
B4	SW3	M6	UCLK	C14	RED0	F2	USB-ASTB	A8	GND	J2	VDDO-3
A3	JA2	P6	LD3	G12	BTN0	F1	USB-DSTB	C1	GND	A5	VDDO-2
A10	JC3	P7	LD2	K14	AN3	G1	LD7	C7	GND	E12	VDDO-2
C9	JC4	M4	BTN2	J12	AN1	G3	SW4	C10	GND	K1	VDDO-2
B9	JC2	N4	LD5	J13	BLU2	H1	USB-DB6	E3	GND	P9	VDDO-2
A9	JC1	M5	LD0	J14	HSYNC	H2	USB-DB5	E14	GND	A11	VDDO-1
B8	MCLK	N5	LD4	H13	BLU1	H3	USB-DB7	G2	GND	D3	VDDO-1
C8	RCCLK	G14	GRN2	H12	CB	B14	TMS	H14	GND	D14	VDDO-1
A7	BTN3	G13	GRN1	J3	JA3	B13	TCK-FPGA	J1	GND	K2	VDDO-1
B7	JB4	F12	AN0	K3	SW2	A2	TDO-USB	K12	GND	L12	VDDO-1
P4	LD6	K13	VSYNC	B1	PS2C	A14	TDO-S3	M3	GND	P2	VDDO-1

Free software tools:

- ♦ **ISE WebPACK 14.7 ;** *(www.xilinx.com)*
- ♦ **Digilent Adept 2** *(www.digilentinc.com)*


ISE Project Navigator (P.20131013) - D:\@_O2o\@@_@Final\PX000_Practical_Part_3_Circ_impl_FPGA\Template_Project\Project_Template\Project_Template.xise - [sch_bloc.sch]
File Edit View Project Source Process Add Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
Project_Template
xc3s100e-5cp132
sch_bloc (sch_bloc.sch)
XLXI_7 - Hex_7seg - Arh_Dec (Hex_7se
XLXI_13 - signal_gen - arh_signal_gen (
Pin_conection.ucf
No Processes Running
Processes: sch_bloc
Design Summary/Reports
Design Utilities
User Constraints
Synthesize - XST
Implement Design
Generate Programming File
Configure Target Device
Generate Target PROM/ACE File
Manage Configuration Project (iMPACT)
Analyze Design Using ChipScope
signal_gen
LED to view state of the input signal
Select the frequency of the clock signal
clock signal is available here
Visual display of the outputs: HexDisplay + BinarDisplay
Your test diagram will be placed hire
Hexadecimal to 7 segment decoder
Hex_7seg
To LEDs
Start Design Files Libraries
Design Summary
sch_bloc.sch
Errors
Console Errors Warnings Find in Files Results
start Windows Commander... Google Translate – G... Digilent Adept PX009 Annex Implem... ISE Project Navigator...



Connect: Basys2
Product: Basys2 - 100

FPGA XC3S100E: sch_bloc.bit
PROM XCF02S

```
Board information loaded.
Found device ID: d5045093
Found device ID: 11c10093
Initialization Complete.
  Device 1: XC3S100E
  Device 2: XCF02S
```

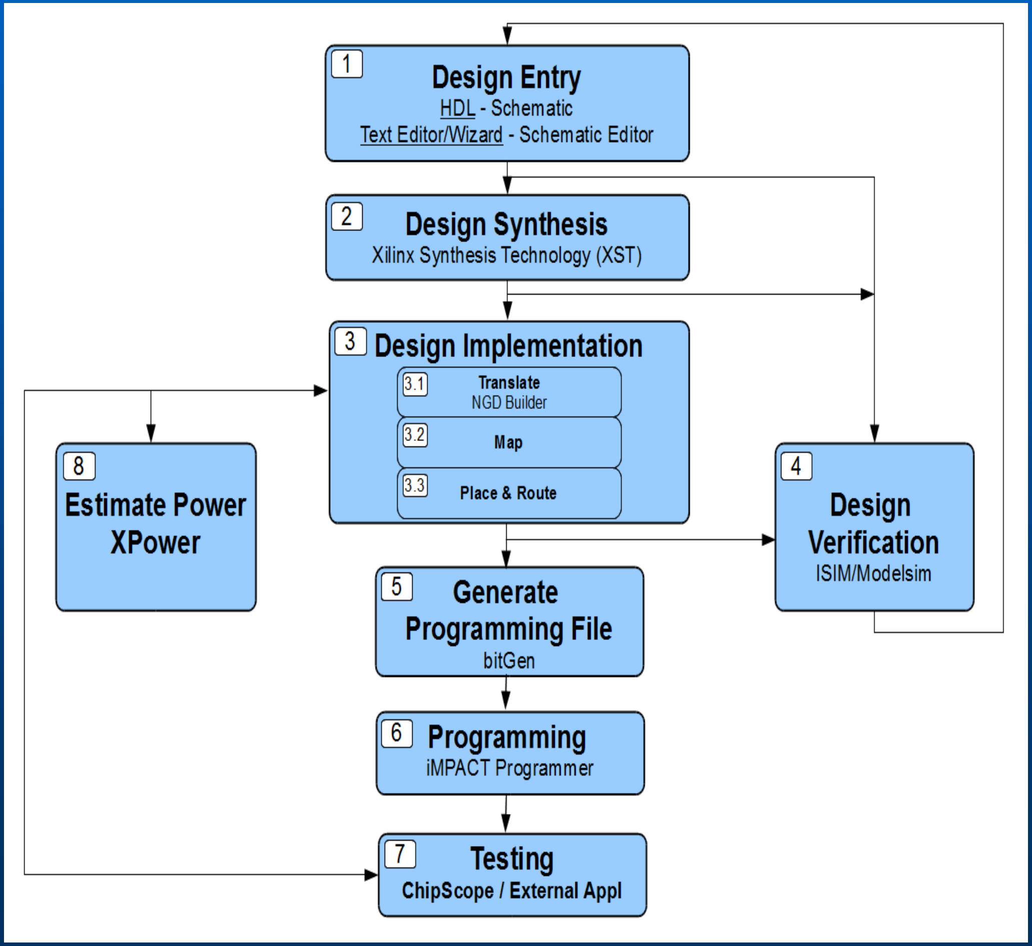
1. Hardware support ; 2. Software support; 3. Applications

Step	Activities
Step 1	Install two software ("ISE Design Tools 14.7" and "Adept 2.3.") following the steps presented in this tutorial; http://www.nhn.ou.edu/~bumm/ELAB/DigiLab_software.html
Step 2	Create a new Project following steps presented in this tutorial; https://classes.soe.ucsc.edu/cmpe100/Winter15/lab/new_project/new_project.html
Step 3	Enter the logic diagram following steps similar to those presented in this tutorial; https://classes.soe.ucsc.edu/cmpe100/Winter15/lab/schematic/schematic.html - ***draw a schematic*** or - ***create a text file describing the logic function*** (using hardware description language like VHDL or Verilog).
Step4	Enter the Implementation Constrain File;

Step	Activities
Step 5	Generate the bitfile Select your top level schematic in the Project Navigator and double click on the process ***Generate Programming File***.
Step 6	Downloading bitfile into FPGA, following steps similar to those presented in this tutorial; https://classes.soe.ucsc.edu/cmpe100/Winter15/lab/configure/configure.html
Step 7	Test the functionality of the implemented circuit according to your specifications ;
Step 8	In case of errors GOTO Step 3, fix your "logic function", re-compile and re-download it. ***We can make downloads in FPGAs as many time as we need (almost no limit), with different functionalities every time we want.***

FPGA Design Flow

Fortunately many steps are made by the design environment without our intervention



Practical aspects:

- **Working with FPGA is not an easy task for beginners;**
- **In each project, for any provider, it is mandatory to have a minimal specification and source file:**
    - specify the target FPGA – in our case we work with Spartan 3E-100 CP132;
    - at least one source file to describe the circuit (schematic or text);
    - at least one constrain file to specify how external devices are connected to the FPGA pins;
- To make things easier, in each practical application, you will use one ***Template Project*** in which there are already implemented some circuits that are necessary for testing different circuits;

Project template architecture (1/2):

In *Project_Template* there are already implemented some circuits that are necessary for testing different circuits with counters or frequency dividers:

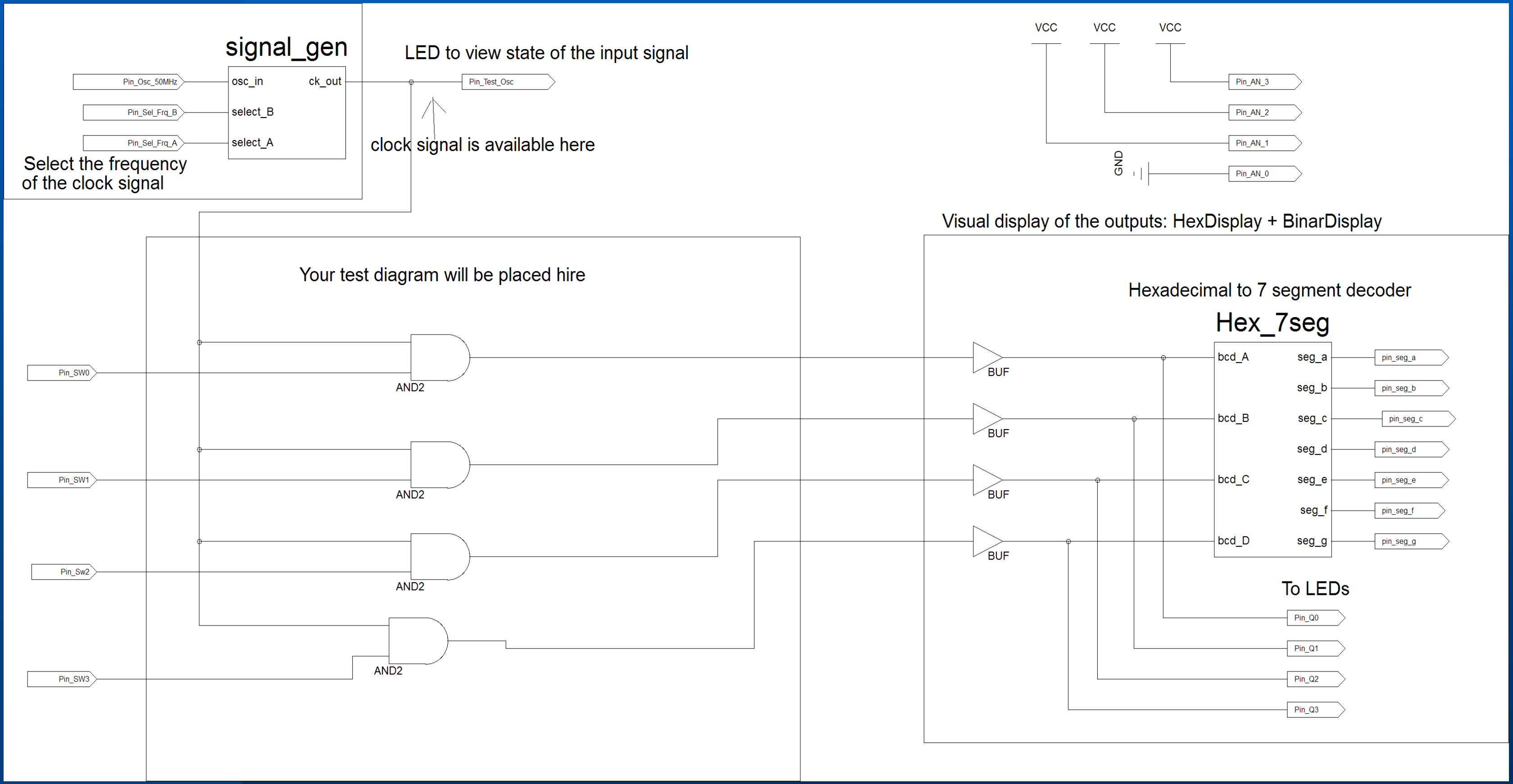
- **Signal generator** – to provide the clock signal for counting under test. This generator has two inputs which are used to change the frequency of the clock signal. These control inputs are connected board switches SW7 and SW6.

- **HexDisplay** – to display the state of the counter, in numeric format, on one of the four digits of the Basys2 board.

- **BinDisplay** – to display the state of the counter, in binary format, using the last four LEDs of the Basys2 board, LD3 to LD0.

- **One Working Area** - where you will put your project;

Template Project Architecture (2/2):



Example 1: Implement and test *Project Template*

Steps to follow:

Step 1: Start *Project_Template*

- Copy *Project_Template* folder on your computer;
- Find *Project_Template.xise* and double-click on this file;
- In *Hierarchy Window* double-click on *Sch_bloc.sch*;

Step 2: Generate configuration file:

- click one time on *Design Tab;*
- click one time on *Sch_bloc.sch* in *Hierarchy Window*;
- double-click on *Generate Programming File* in *Processes Window*;
- wait until obtain green light on *Generate Programming File.*

Example 1: Implement and test *Project Template*

Steps to follow:

Step 3: Configure the FPGA board (download configuration file):

- Connect USB cable between PC and Basys2 board and then turn on the power (put switch SW8 in ON position);
- Start *Digilent Adept* and browse in project folder and find *sch_bloc.bit*;
- Then click on *Program Button* and ignore (click Yes) some warning;

Step 4: Test the circuit functionality:

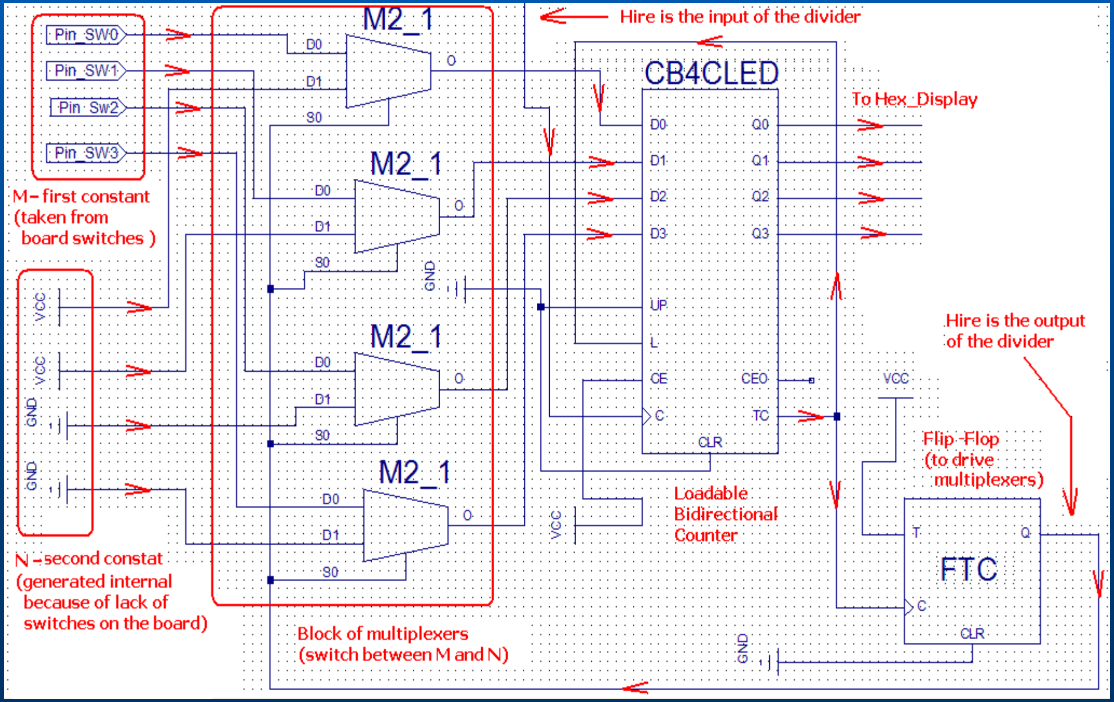
- Put SW7 and SW6 in lower position to select the lower frequency of clock signal;
- Try different combination on last 4 switches (SW3 to SW0) and see what happens on the display and what happens with the last 4 LEDs (LD3 to LD0);
- Change position on SW6 and see what happens with frequency of clock signal. Try different combination on SW7 and SW6 to see the possibilities of the signal generator.

Example 2: Programmable Frequency divider implementation in FPGA

Steps to follow:

Step 1: Load *Project_Template* and remove all logic gate from working area;

Step 2: Draw the diagram in the next figure in working area



Pin_SW0
Pin_SW1
Pin_Sw2
Pin_SW3
M – first constant (taken from board switches)
VCC
VCC
GND
GND
N – second constat (generated internal because of lack of switches on the board)
M2_1
M2_1
M2_1
M2_1
D0
D1
S0
O
Block of multiplexers (switch between M and N)
Hire is the input of the divider
CB4CLED
D0
D1
D2
D3
UP
L
CE
C
CLR
Q0
Q1
Q2
Q3
CEO
TC
To Hex_Display
Loadable Bidirectional Counter
Hire is the output of the divider
Flip -Flop (to drive multiplexers)
FTC
T
Q
C
CLR

Example 2: Programmable Frequency divider implementation in FPGA

Steps to follow:

Step 3: Generating program file and then configure the FPGA;

Step 4: Test the circuit functionality:

- From switches SW7 and SW6 select the lowest frequency for the signal clock;
- From switches SW3 and SW0 select M = 7 = 0111 and carefully analyze the operating mode of the divider;

Example 3: Modify the previous project

Steps to follow:

Step 1: Load the previous project

Step 2: Add a supplementary output in schematic diagram

in order to see the logic state of the output signal.

For example you can use LD5 which is connected at pin N4.

Step 3: Add a supplementary line in constrain file

Step 4: Save the modification

Step 5: Remake the programming file and download in FPGA

Step 6: Verify the functionality of the implemented project

see how time signal stay in *High* state (LED lighting) and how time stay in Low state;

Thanks for your attention!