

UNIVERSITÀ DEGLI STUDI DELL' AQUILA

Embedded Systems (Dott. Luigi Pomante)

A.A. 2019/2020

HomeLab ARTY FPGA

(VHDL's Exercises and Introduction to Vivado Design Suite)

Author

Giacomo valente (giacomo.valente@univaq.it)

Version 5.0: 10/10/2019

Index

Before start: preparation.....	4
Components	Errore. Il segnalibro non è definito.
Required software	Errore. Il segnalibro non è definito.
Time and feedbacks.....	Errore. Il segnalibro non è definito.
Virtual Machine and Vivado Design Suite	4
ARTY.....	4
Vivado Installation	Errore. Il segnalibro non è definito.
Driver Installation	Errore. Il segnalibro non è definito.
Communication test between Board and PC	4
Further investigations.....	5
Section 1: Simulation and Synthesis of simple combinatorial circuits	6
1.1 – VHDL Description of Half Adder	6
1.2 - Simulation.....	9
1.3 -Synthesis.....	10
1.3 – To Do.....	10
Section 2: Sequential circuits, I/O and constraints.....	12
2.1 – Counter	12
2.2 - Sequence detector (Finite State Machines - FSM)	15
2.3 – To Do.....	15
Section 3: PicoBlaze.....	16
3.1 – Input/Output and base of programming.....	Errore. Il segnalibro non è definito.
3.2 – Further investigations.....	Errore. Il segnalibro non è definito.
Section 4: Serial Communication.....	Errore. Il segnalibro non è definito.
4.1 – PicoBlaze & UART	Errore. Il segnalibro non è definito.
4.2 – Further Investigations.....	Errore. Il segnalibro non è definito.

Before start: preparation

Board reservation

In order to perform the homelab, one week is allowed. Please reserve the board using the following link:

https://docs.google.com/forms/d/e/1FAIpQLSfW05Px7HxUUZk2IgnHq3Pn5pPEfkskCJ_04O8GAmtzo1fEn/viewform?usp=sf_link

and you will be scheduled.

Virtual Machine and Vivado Design Suite

Install Virtual Box on the host PC (download the latest version).

Import the Virtual Machine image on Virtual Box and launch the booting. The password is: *embeddedsystem*

ARTY

Take a deep look to Reference Manual (RM) [\[link\]](#) and verify, before the beginning of the homelab, that the board is properly configured. This means that the *jumpers* configuration must be checked: a jumper serves as connection between two points. In general, it can allow certain voltages to be applied in some parts of the circuit, so take care when selecting jumpers configurations: they can damage the board if not correctly set. The rightness of the jumpers position can be checked on RM: select them in order to use the board with JTAG connection.

Communication test between Board and PC

Connect the board to the PC through USB cable, and it will be automatically powered-on.

Connect the ARTY board to the Virtual Machine by selecting it among the devices:

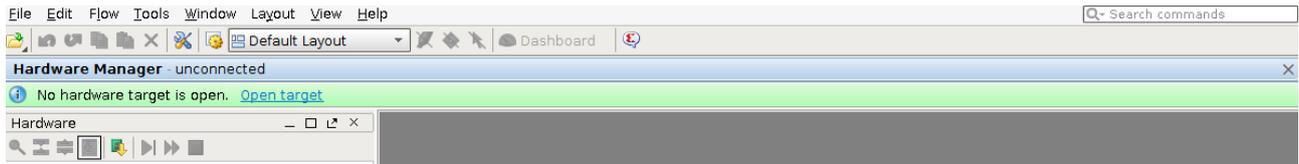


Check the connection of the board to Vivado:

- Set environment variables with the command:

```
source /opt/Xilinx/Vivado/2017.4/settings64.sh
```
- run Vivado with the command:

```
vivado
```
- Select *Open Hardware Manager* from the icons.



Select the link *Open Target* and click to *Auto-Connect*. The ARTY board has to be recognized by the environment without errors.

Further investigations

- Standard IEEE 1149.1: Standard Test Access Port and Boundary-Scan Architecture (JTAG)
- ARTY board, documentation [\[link\]](#)

Section 1: Simulation and Synthesis of simple combinatorial circuits

This section refers on how to use Vivado Design Suite for simulation and synthesis of a simple digital circuit, the half adder.

1.1 – VHDL Description of Half Adder

The creation of a new project in Vivado and description of the half-adder come through the following steps:

- Exit from Vivado and re-launch it from linux shell
- Select *Create New Project* from icons
- Click *Next* and gives project name and project location (pay attention to select destination folders without spaces in the folder name)
- Click *Next* and select RTL Project
- Click *Next* and Select VHDL Language and Mixed simulator. Do not add sources and click *Next*. Do not add Existing IP and click *Next*. Do not add constraints and click *next*.

Select ARTY Board from the list and click next:

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



Select: Parts Boards

Filter

Vendor:

Display Name:

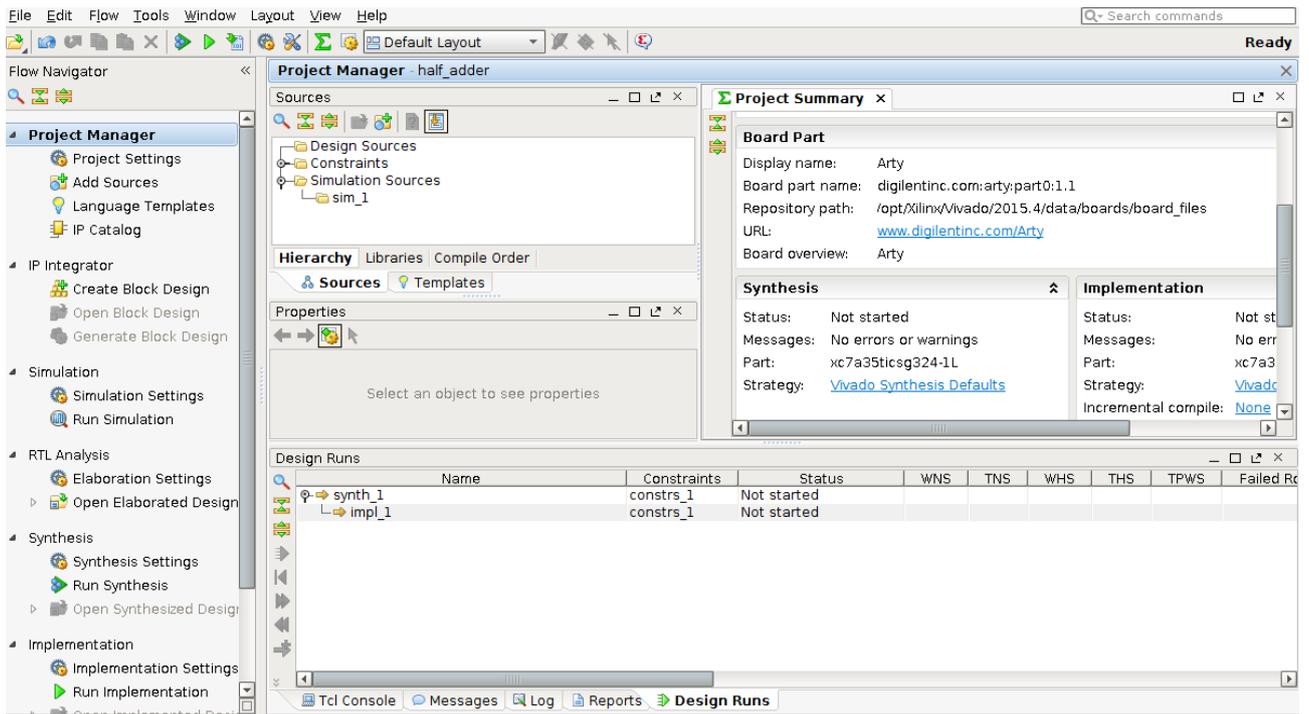
Board Rev:

Search:

Display Name	Vendor	Board Rev	Part	I/O Pin Count	File Version	Block RAMs	DSPs	FlipFlops	GTPE2 Transceivers	Gb Transceivers
Arty Z7-20	digilentinc.com	A.0	xc7z020clg400-1	400	1.0	140	220	106400	0	0
Arty	digilentinc.com	C.0	xc7a35ticsg324-1L	324	1.1	50	90	41600	0	0
Basys3	digilentinc.com	C.0	xc7a35tccpg236-1	236	1.1	50	90	41600	2	2
Cmod A7-15t	digilentinc.com	B.0	xc7a15tccpg236-1	236	1.1	25	45	20800	2	2
Cmod A7-35t	digilentinc.com	B.0	xc7a35tccpg236-1	236	1.1	50	90	41600	2	2
Genesys2	digilentinc.com	H	xc7k325ffg900-2	900	1.1	445	840	407600	0	16
Nexys4	digilentinc.com	B.1	xc7a100tccg324-1	324	1.1	135	240	126800	0	0
Nexys4 DDR	digilentinc.com	C.1	xc7a100tccg324-1	324	1.1	135	240	126800	0	0
Nexys Video	digilentinc.com	A.0	xc7a200tcbg484-1	484	1.1	365	740	269200	4	4
Zedboard	digilentinc.com	D.3	xc7z020clg484-1	484	1.0	140	220	106400	0	0
Zybo	digilentinc.com	B.3	xc7z010clg400-1	400	1.0	60	80	35200	0	0
ZedBoard Zynq Evaluation and Development Kit	em.avnet.com	d	xc7z020clg484-1	484	1.3	140	220	106400	0	0
Artix-7 AC701 Evaluation Platform	xilinx.com	1.1	xc7a200tcbg676-2	676	1.2	365	740	269200	8	8
Kintex-7 KC705 Evaluation Platform	xilinx.com	1.1	xc7k325ffg900-2	900	1.2	445	840	407600	0	16
Virtex-7 VC707 Evaluation Platform	xilinx.com	1.1	xc7vx485tffg1761-2 1,761	1,761	1.2	1030	2800	607200	0	28
Virtex-7 VC709 Evaluation Platform	xilinx.com	1.0	xc7vx690tffg1761-2 1,761	1,761	1.7	1470	3600	866400	0	36
ZYNQ-7 ZC702 Evaluation Board	xilinx.com	1.0	xc7z020clg484-1	484	1.2	140	220	106400	0	0
ZYNQ-7 ZC706 Evaluation Board	xilinx.com	1.1	xc7z045ffg900-2	900	1.2	545	900	437200	0	16

Review the settings and click finish.

Now the following screen will appear:



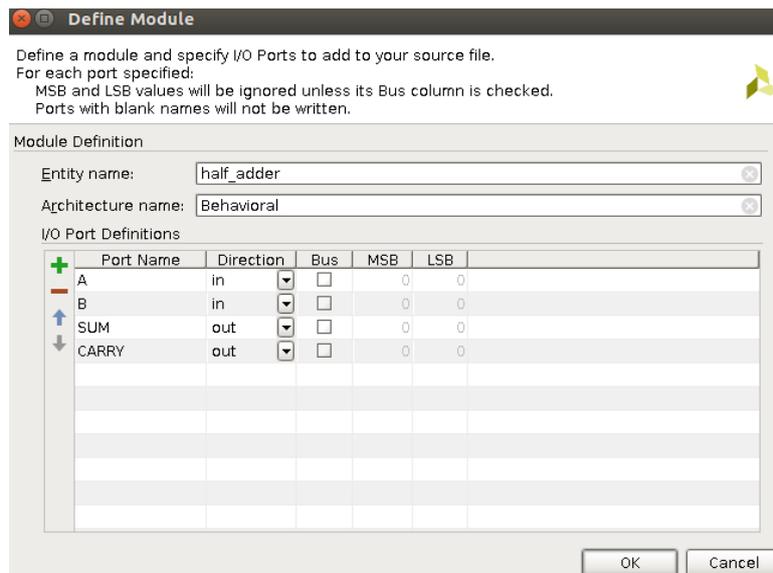
The screen is divided in various different areas:

- *Source Window*: it is located at the top of the environment by default, and it contains all the sources associated to the project. For example, *Design Sources* represent HDL files (that can be VHDL or Verilog), while *Constraints Sources* represent constraints files
- *Flow Navigator Pane*: it is located at the left of the environment by default, and it contains the main steps that can be performed in a design with Vivado (such as *Simulation*, *Synthesis*, etc.)

The project just created does not contain source files. Add one by right clicking the mouse into source window and selecting *Add Sources*.



Select *Add or create design sources* and click *Next*. Click on the green cross at the top left and select *Create File*. Specify the name (e.g. *half_adder*, pay attention to not insert spaces in the file name) and click *OK*. Then click *Finish*. A wizard will be opened: it allows to define module parameters (i.e. entity name and architecture name).



After filling various fields keeping in mind the half adder structure, click *OK* to complete the source file creation. Open the new generated source file by double clicking on it in the source window. Internal to this file you can see that the structure of the VHDL code has been automatically set up. It is possible to note how it is necessary only to write the architecture section (i.e. the functionality of the system). Complete the description as seen during the course lesson:

```
ARCHITECTURE Behavioral OF half_adder IS

BEGIN

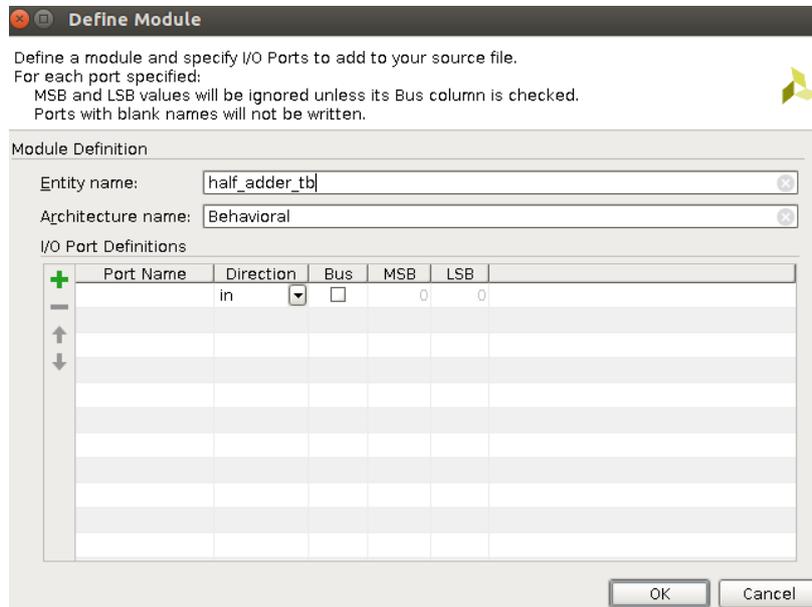
    SUM <= A xor B;
    CARRY <= A and B;

END Behavioral;
```

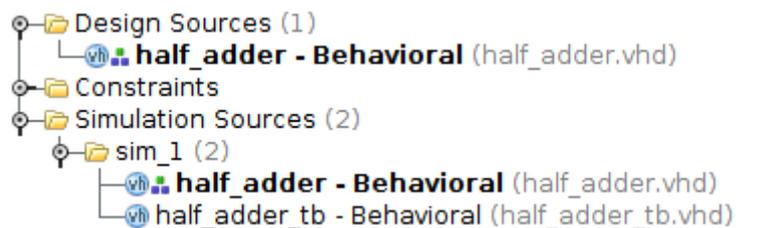
At this point, the description of half adder module is complete. Save the modified source file.

1.2 - Simulation

Verify that the behavior of the block is the one expected: i.e. perform a *Behavioral* simulation. For this it is necessary to create a *testbench* to simulate the described circuit. Add to the project another source file, but now select *Add or create simulation sources*. Then create a new file from the top-left green arrow and select *Finish*. Do not insert ports to the entity (it is a *testbench*):



From source window, select the *testbench* from *Simulation Sources*:



Consider the *testbench* contained in the homelab folder. In the code section it can be noted that a constant *period* has been defined: this one can be used to define time interval variations of input signals. By means of the process statement and the wait instruction, it is possible to model the desired signal input evolutions. Examine, for example, the following piece of code and draw the equivalent timing diagrams. Then, add these stimuli to the right section of the provided *testbench* file:

```
A_stim_proc: process
begin
    wait for 100 ns;
    SIG_A <='0';
    wait for period;
    SIG_A <='1';
    wait for period;

end process;

B_stim_proc: process
begin
    wait for 100 ns;
    SIG_B <='0';
```

```

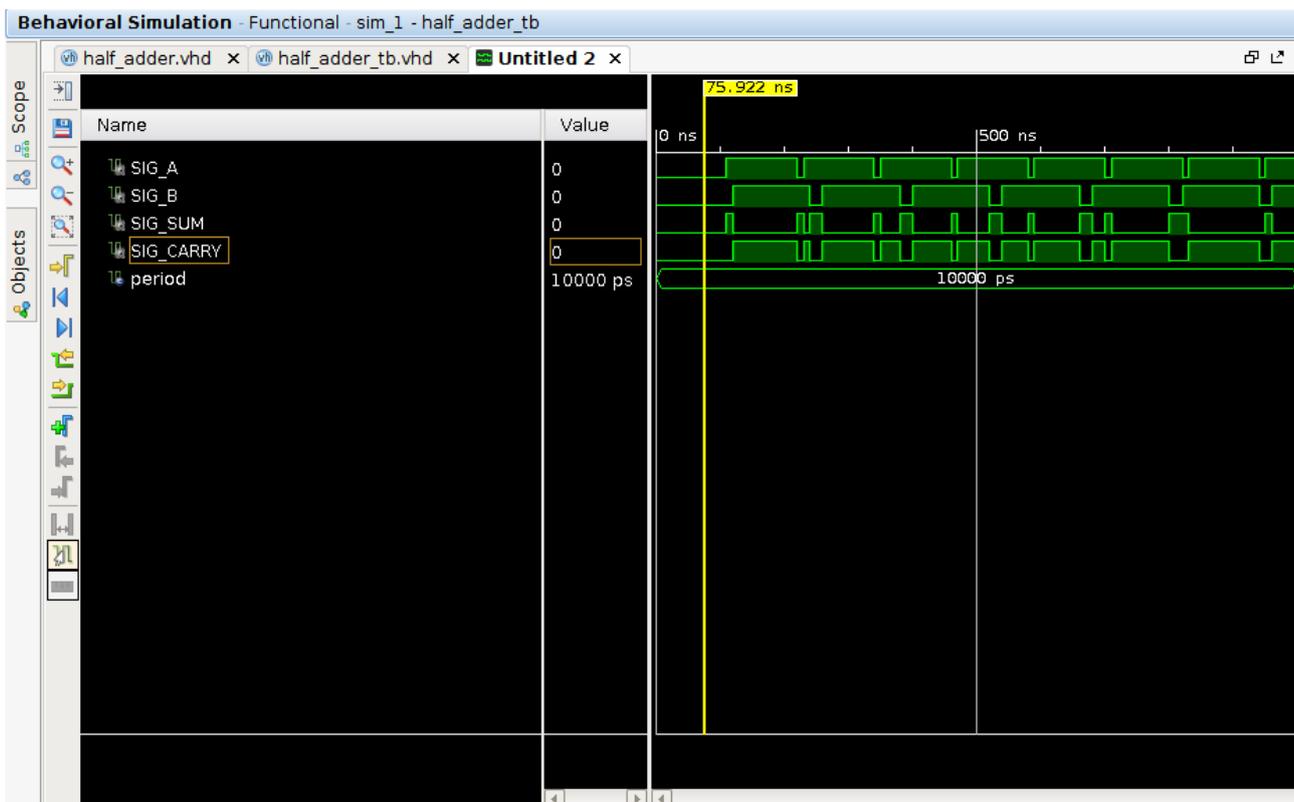
        wait for 2*period;
        SIG_B <='1';
        wait for 2*period;

    end process;

```

Once defined input signals evolution, it is possible to run the simulation: in the flow navigator pane, select *Simulation Settings* and check if Vivado Simulator is set and if the top module is the *testbench* that was already written. Then, in the *Simulation Tab*, set the *Simulation Runtime*. At the end, select *OK* and select *Run Simulation* and *Run Behavioral Simulation* in the flow navigator pane.

This last step will launch the *XSim* simulator, and this one will show automatically the timing diagrams of the simulation process (click to various buttons to show waveforms with different levels of detail).



Check simulation results by verifying that half adder outputs behave good with respect to the inputs.

1.3 -Synthesis

Close *XSim*, go back to source window, select the half adder module and click on *Run Synthesis* in the flow navigator pane. At the end, it is possible to check reports, to have a look on synthesized design or continue by running *Implementation*.

1.3 – To Do

Try to develop VHDL description of the following combinatorial circuits:

- Multiplexer
- Demultiplexer

Verify that the circuit behavior is correct by means of behavioral simulation. Keep in mind that the description is not uniquely and that if the circuit is simulatable, it does not imply that it is synthesizable. Be aware only to the simulation results, not the implementation.

Section 2: Sequential circuits, I/O and constraints

In a development board, like ARTY, the FPGA is surrounded with hardware components (e.g. UART -> USB converters, VGA, PMODs, LEDs, switch buttons, push buttons, etc.). For the use of this components, it must be possible to connect the implemented blocks into FPGA with external hardware.

In this section, a first sequential circuit into FPGA will be implemented, and how to show the outputs by means of LEDs placed on the board will be illustrated [5].

2.1 – Counter

Create a new project in Vivado (e.g. named *homelab_counter*), still targeting the ARTY board, and add a new source file having the entity named *hl_counter*. The possible structure of the counter is reported in the *homelab* folder.

It is possible to note that it is necessary to generate a “slow clock”, otherwise the frequency of the counting would be too fast and it will not be visible to human eye through a LED.

Once defined the architecture of the counter, it is possible to go to next phase.

At the middle of work, it is possible to perform a behavioral simulation. For this sake, a *testbench* has to be defined, in the same manner of the last exercise (i.e. the half adder), and the counter has to be instantiated into this one and properly excited.

Pay attention to the fact that, after simulation launch, by default one is able to look at the behavior of input and output of the top-level section of the system, that is the entity: remember that a slow clock has been used internally to the counter, in order to have the ability to show the LED blinking on the board. From the simulation point of view, this could require a very long simulation time. In order to check the functionality with smaller simulation time, one can either have a look to the internal signals of the counter module, or rewrite the architecture of the VHDL module removing the process that slows down the counter evolution. This can be done by navigating into *XSim* through *Object Tab* and *Scope Tab*, and by dragging and drop necessary signals. Note that the simulation should be re-launched in order to include these signals.

Going toward the implementation phase, remember that after the downloading of the *bitstream* on the FPGA, the counter outputs has to drive directly LEDs sited on the board. In order to do this, some constraints have to be defined for the implementation by operating on *Xilinx Design Constraints (XDC)*. An XDC file is a list of functions written in *TCL* language [\[link\]](#) that allow to make some operations targeting constraints in the Vivado Design Suite. Constraints can be of different types: board constraints, timing constraints, etc. In this context, the interest is on board constraints. The way (not the unique) to operate with constraints, in this context, is to start from a single XDC file that contains all the board constraints, related to the ARTY board, initially commented out. In order to use this XDC file, it must be downloaded from the *Digilent* repository [\[link\]](#). Have a look on the internal of the file, and note a list of TCL commands. The ARTY board has various inputs and outputs elements, as indicated in RM. Considering that the counter has 1 clock input, 1 reset input, 1 counter output (composed of 4 bits), there is the need of 1 input clock (e.g. provided by an oscillator), 1 input reset (e.g. a push-button) and 4 output counter (e.g. 4 LEDs). After checking that these elements are available on the board, the XDC file can be imported into the project:

- Going in the source pane and adding a source
- Selecting *Add or Create Constraints*
- Adding the XDC file using the top-left green cross (set the *Copy constraints files into the project*)

- Clicking *Finish*

Now the XDC file can be modified, in order to make required board connections. In particular, the following lines can be commented out:

- Connect the clock:

```
set_property -dict { PACKAGE_PIN E3  IOSTANDARD LVCMOS33 } [get_ports { clk }]
```

- Connect the output:

```
set_property -dict { PACKAGE_PIN H5  IOSTANDARD LVCMOS33 } [get_ports { count_out[0]
}]
```

```
set_property -dict { PACKAGE_PIN J5  IOSTANDARD LVCMOS33 } [get_ports { count_out[1] }]
```

```
set_property -dict { PACKAGE_PIN T9  IOSTANDARD LVCMOS33 } [get_ports { count_out[2] }]
```

```
set_property -dict { PACKAGE_PIN T10 IOSTANDARD LVCMOS33 } [get_ports { count_out[3]
}]
```

- Connect the reset:

```
set_property -dict { PACKAGE_PIN D9  IOSTANDARD LVCMOS33 } [get_ports { reset }]
```

The synthesis process and *bitstream* generation can be completed now, by clicking on *Generate Bitstream* in the flow navigator pane. At the end, board can be connected to the PC and to the Virtual Machine. The FPGA configuration can be done by using *Hardware Manager*, in particular:

- selecting *Open Hardware Manager* in the screen at the end of *bitstream* generation or in the flow navigator pane
- clicking on *Open Target* and asking for *auto-connection*
- Clicking on *Program Device* and selecting the *bitstream* generated file (located in the *runs/impl_1* folder of the project tree).

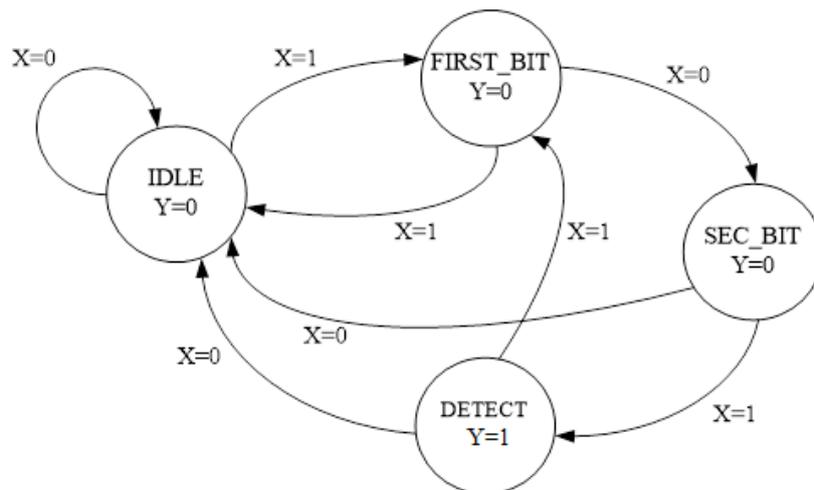
At the end of operation, the LEDs 0-3 on the board should illuminate according to counter evolution. It can be tested also the reset by pushing the BTN0 push-button.

2.2 - Sequence detector (Finite State Machines - FSM)

In this section, it will be illustrated how to implement a circuit able to detect an exact binary sequence on the input line. Suppose that a synchronous detector has to be implemented, therefore data input line (X) and clock signal are expected as inputs: data will then be read at the rising edge of the clock. The output (Y) will be equal to 1 only when the sequence is detected.

Using the HDL for system description, XDC for constraints and Vivado Design Suite for implementation, the task is to implement a system that solves the problem and to verify the functionality using two switches to emulate, respectively, the clock signal and the input signal.

The following state diagram describes a sequence detector in which the detected word on the serial port (101) is “hardcoded”.



A draft of VHDL code for a Finite State Machine (Moore type, i.e. the output is determined only by the current machine state, and the next machine state is determined by the input and the current machine state) that implements the behavior of the previous diagram is provided within the homelab folder. The definition of an enumerated data type is convenient for the state representation. It is left as exercise the completion of the project (by assigning XDC to specify the right connection between sequence detector, board LEDs and board switches and to perform the FPGA configuration in the same way of the previous exercise).

2.3 - To Do

Simulate the GCD single purpose processor: do it for every kind of hardware description, i.e. behavioral, RTL with FSM and RTL with FSM and datapath. You already have the files from the extra folder from “Introduction to VHDL” lessons.

Section 3: MicroBlaze

MicroBlaze is Xilinx 32-bit RISC Harvard architecture soft processor core with a rich instruction set optimized for embedded applications. The MicroBlaze soft processor solution delivers complete flexibility to select the combination of peripheral, memory and interface features that will give you the exact system you need at a reduced cost on a single FPGA.

Following the tutorial at the following link:

<https://reference.digilentinc.com/vivado/getting-started-with-ipi/start>

build a MicroBlaze on ARTY board and program it.

Section 5: References and useful readings

[1] ARTY Reference Manual [\[link\]](#)

[2] Pong P. Chu – FPGA Prototyping by VHDL Examples - Wiley

[3] VHDL Tutorial – Learn by examples [\[link\]](#)

Acknowledgements

We would like to thank Fabio Federici, Walter Tiberti, Gianni Rea, Federico Angeloni, Mattia Micozzi, Celestino De Crescente Pinti, Marco Mirabilio and Gabriella D'Andrea for the support to solve some issues.