University of Ioannina

Department of Electronics Engineering
Computers & Information Technology

Testing and Reliability of Electronic Systems

Spring Semester 2024

JTAG 1149.1

Christos Dimitresis **4351** **cs04351@uoi.gr**

Contents

JTAG Building Blocks 1149.1

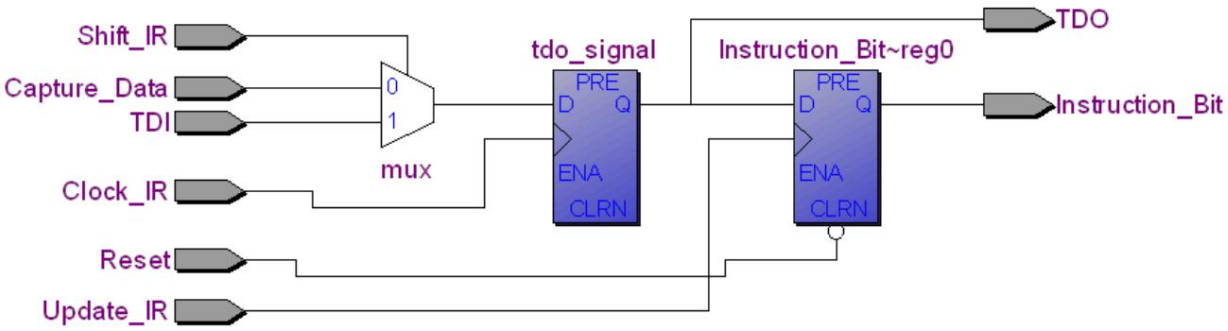
Below are the building blocks for implementing a Boundary Scan chain according to the **JTAG 1149.1** standard . Due to the simplicity of the structural elements, their correctness is verified by synthesizing them in Quartus and visualizing them at the RTL level.

** Code files in VHDL language that implement the following building blocks accompany the reference in the delivery file.

** The structural elements have been implemented from the beginning in relation to exercise 3 while more comprehensible signal names have also been used.

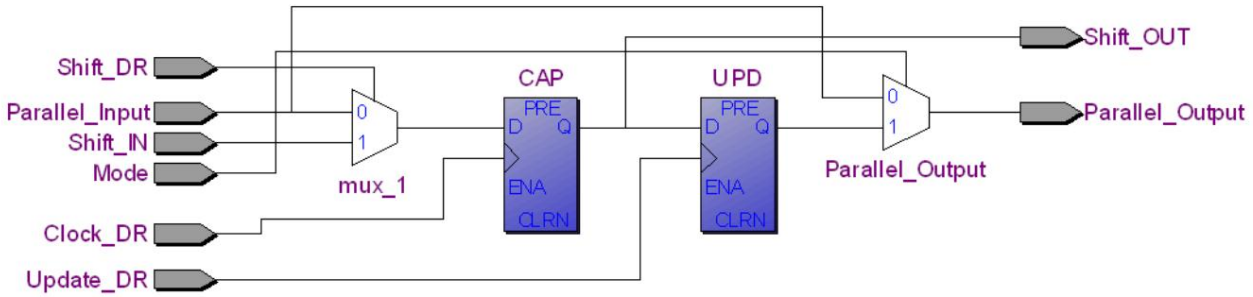
Instruction Register (IR)

File name that implements the Instruction Register (IR) building block: ***IR_REG_JTAG.vhdl***



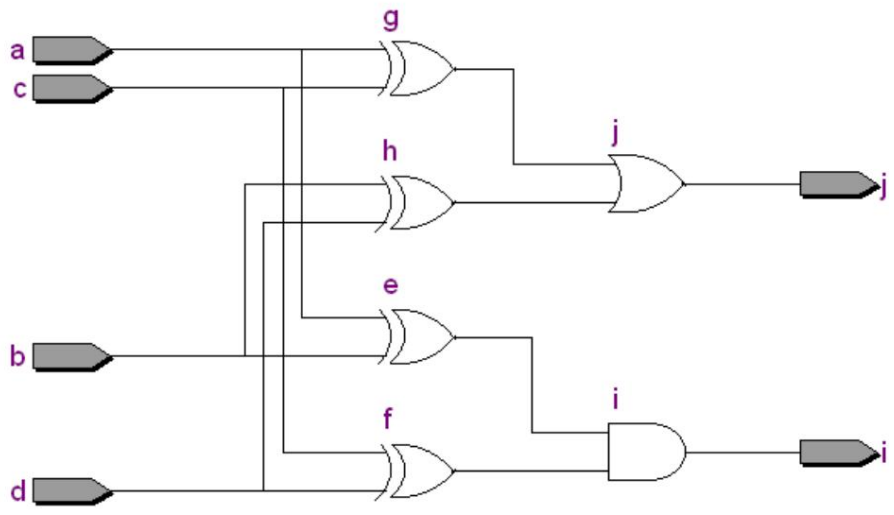
Boundary Register Cell (BRC)

Filename implementing the Boundary Register Cell (BRC) building block: ***BR_CELL_JTAG.vhdl***



Circuit Under Test (CUT)

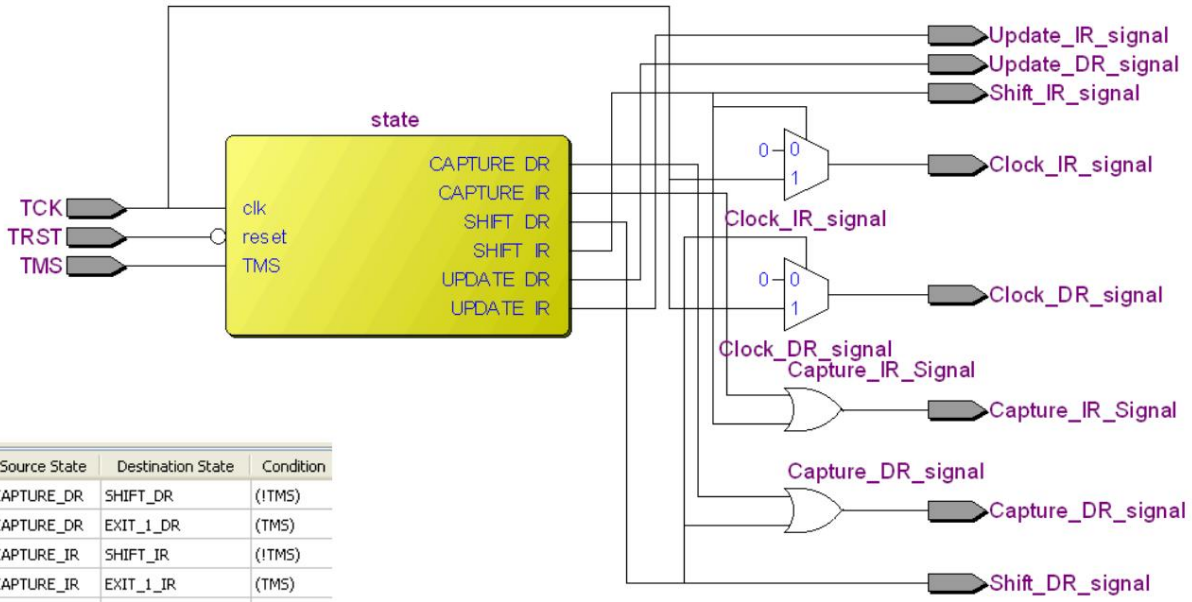
File name that implements the Circuit Under Test (CUT) building block: ***Circuit_Under_Test.vhdl***



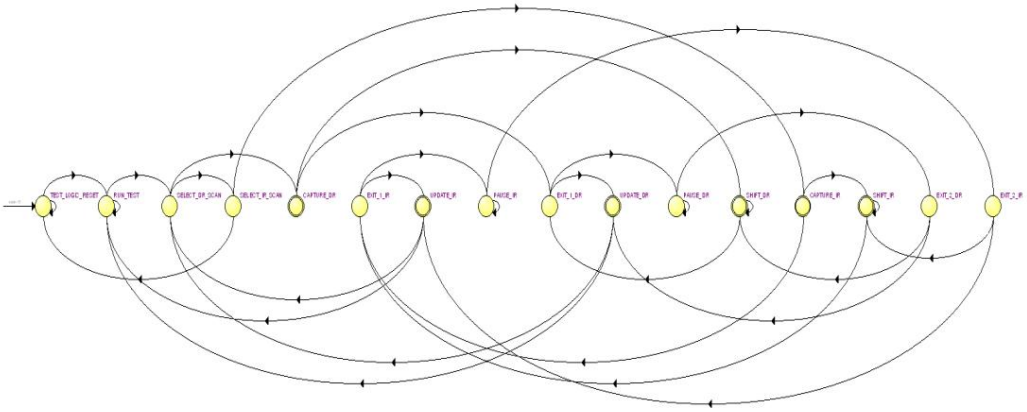
Tap Controller FSM (TAP)

File name that implements the Tap Controller FSM (TAP) building block: ***tap_controler_FSM.vhdl***

** The **Declare.vhdl** file is also necessary because in it the **state type is defined,** with the use of which we assign a label in natural language for each state of the FSM.



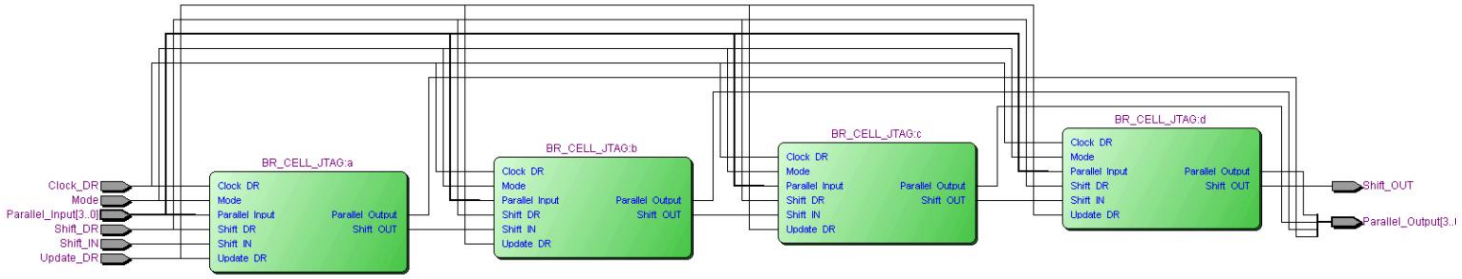
	Source State	Destination State	Condition
1	CAPTURE_DR	SHIFT_DR	(!TMS)
2	CAPTURE_DR	EXIT_1_DR	(TMS)
3	CAPTURE_IR	SHIFT_IR	(!TMS)
4	CAPTURE_IR	EXIT_1_IR	(TMS)
5	EXIT_1_DR	UPDATE_DR	(TMS)
6	EXIT_1_DR	PAUSE_DR	(!TMS)
7	EXIT_1_IR	UPDATE_IR	(TMS)
8	EXIT_1_IR	PAUSE_IR	(!TMS)
9	EXIT_2_DR	UPDATE_DR	(TMS)
10	EXIT_2_DR	SHIFT_DR	(!TMS)
11	EXIT_2_IR	UPDATE_IR	(TMS)
12	EXIT_2_IR	SHIFT_IR	(!TMS)
13	PAUSE_DR	PAUSE_DR	(!TMS)
14	PAUSE_DR	EXIT_2_DR	(TMS)
15	PAUSE_IR	PAUSE_IR	(!TMS)
16	PAUSE_IR	EXIT_2_IR	(TMS)
17	RUN_TEST	SELECT_DR_SCAN	(TMS)
18	RUN_TEST	RUN_TEST	(!TMS)
19	SELECT_DR_...	SELECT_IR_SCAN	(TMS)
20	SELECT_DR_...	CAPTURE_DR	(!TMS)
21	SELECT_IR_...	TEST_LOGIC_RESET	(TMS)
22	SELECT_IR_...	CAPTURE_IR	(!TMS)
23	SHIFT_DR	SHIFT_DR	(!TMS)
24	SHIFT_DR	EXIT_1_DR	(TMS)
25	SHIFT_IR	SHIFT_IR	(!TMS)
26	SHIFT_IR	EXIT_1_IR	(TMS)
27	TEST_LOGIC...	TEST_LOGIC_RESET	(TMS)
28	TEST_LOGIC...	RUN_TEST	(!TMS)
29	UPDATE_DR	SELECT_DR_SCAN	(TMS)
30	UPDATE_DR	RUN_TEST	(!TMS)
31	UPDATE_IR	SELECT_DR_SCAN	(TMS)
32	UPDATE_IR	RUN_TEST	(!TMS)

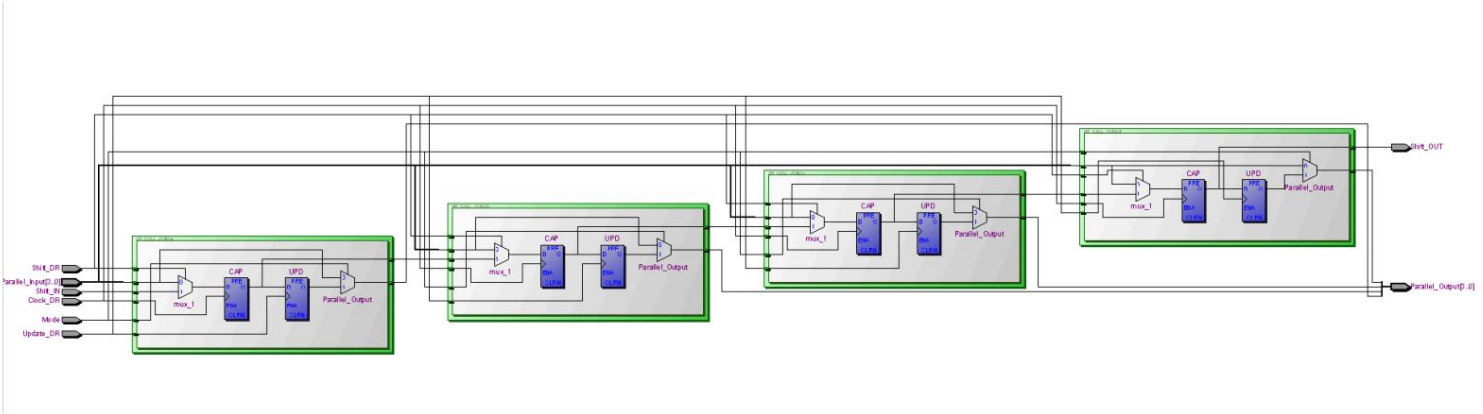


Boundary Register Cells 4 Bits (BRC_4BITS)

The register connected to the CUT inputs is created by 4 BRC cells connected appropriately.

Filename implementing the Boundary Register Cells 4 Bits (BRC_4BITS) building block: ***BRCell_4bits_JTAG.vhdl***

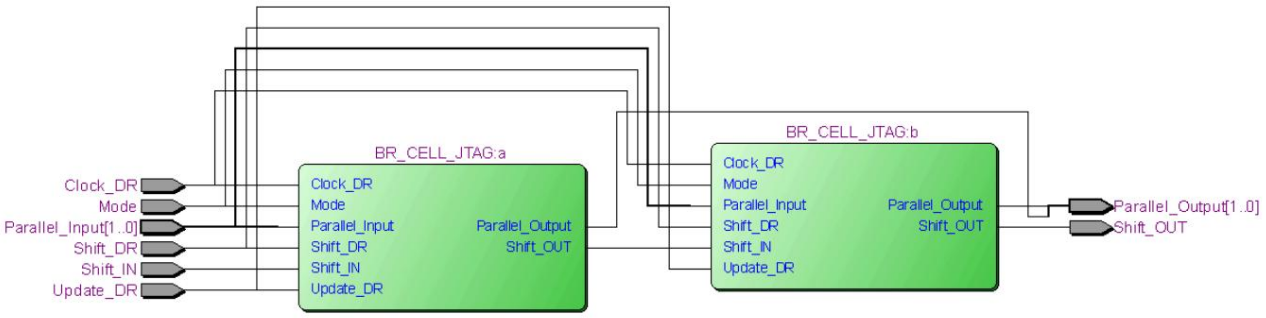


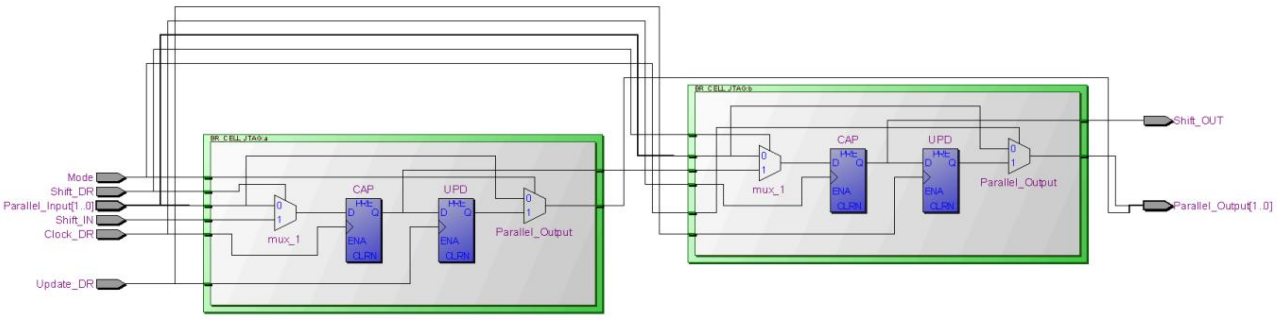


Boundary Register Cells 2 Bits (BRC_2BITS)

The register connected to the outputs of the CUT is created by 2 BRC cells connected appropriately.

Filename implementing the Boundary Register Cells 2 Bits (BRC_2BITS) building block: ***BRCell_2bits_JTAG.vhdl***





Instruction Register 2 BITS (IR_2BITS)

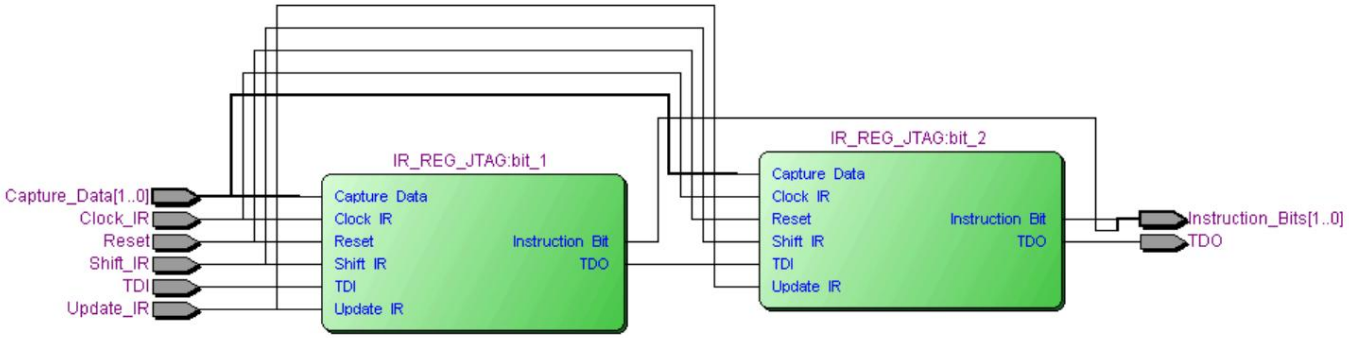
The instruction register that we will use in our implementation consists of 2 registers of the **IR** building block shown earlier. 2 bits is the minimum instruction register size we can use to implement the 3 mandatory instructions required by the JTAG 1149.1 standard.

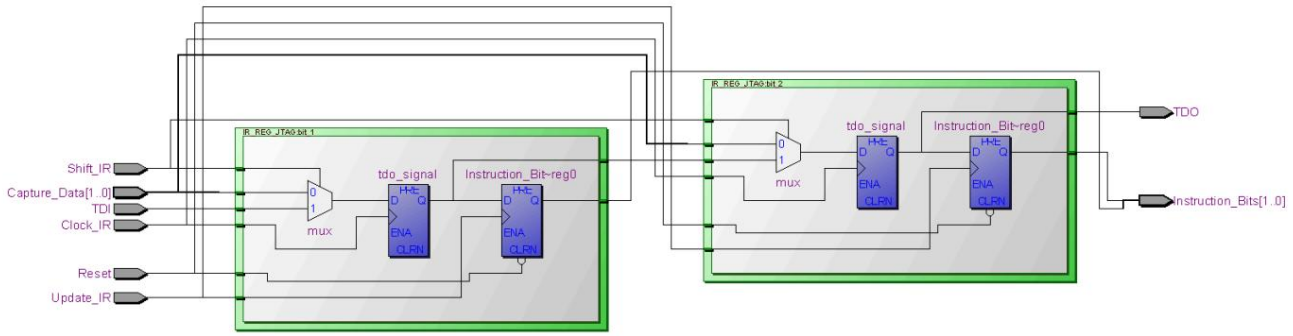
Specifically, these 3 orders are the following:

1) BYPASS
2) SAMPLE/PRELOAD
3) EXTEST

and examples of their implementation in our implementation will be presented below.

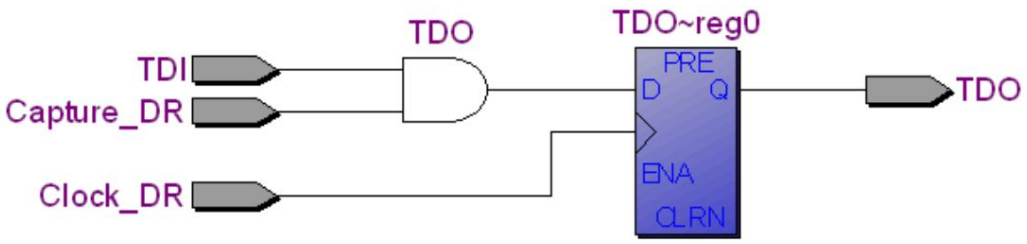
File name implementing the Instruction Register 2 BITS (IR_2BITS) building block:
IR_REG_2bits_JTAG.vhdl





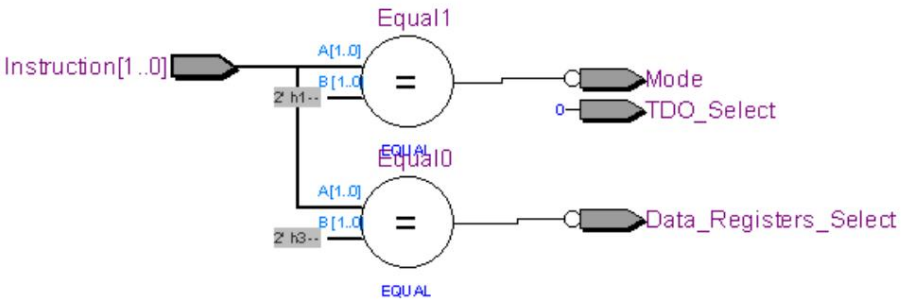
Bypass Register (BYPASS_REG)

File name that implements the Bypass Register (BYPASS_REG) building block:
Bypass_REG_JTAG.vhdl

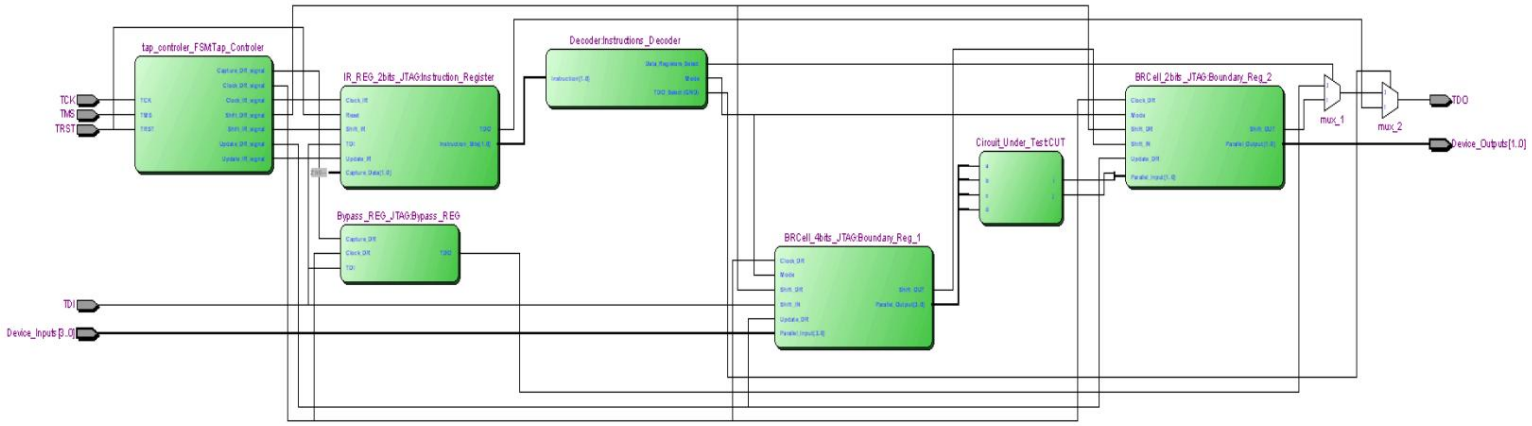


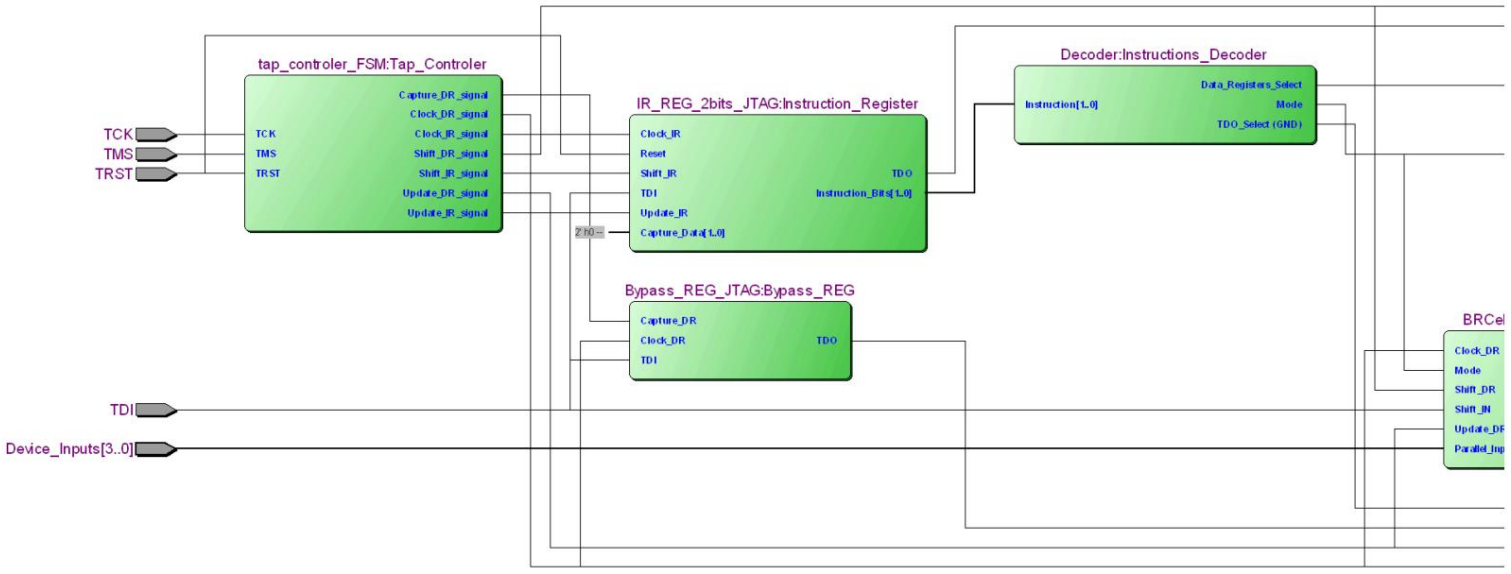
Decoder (DEC)

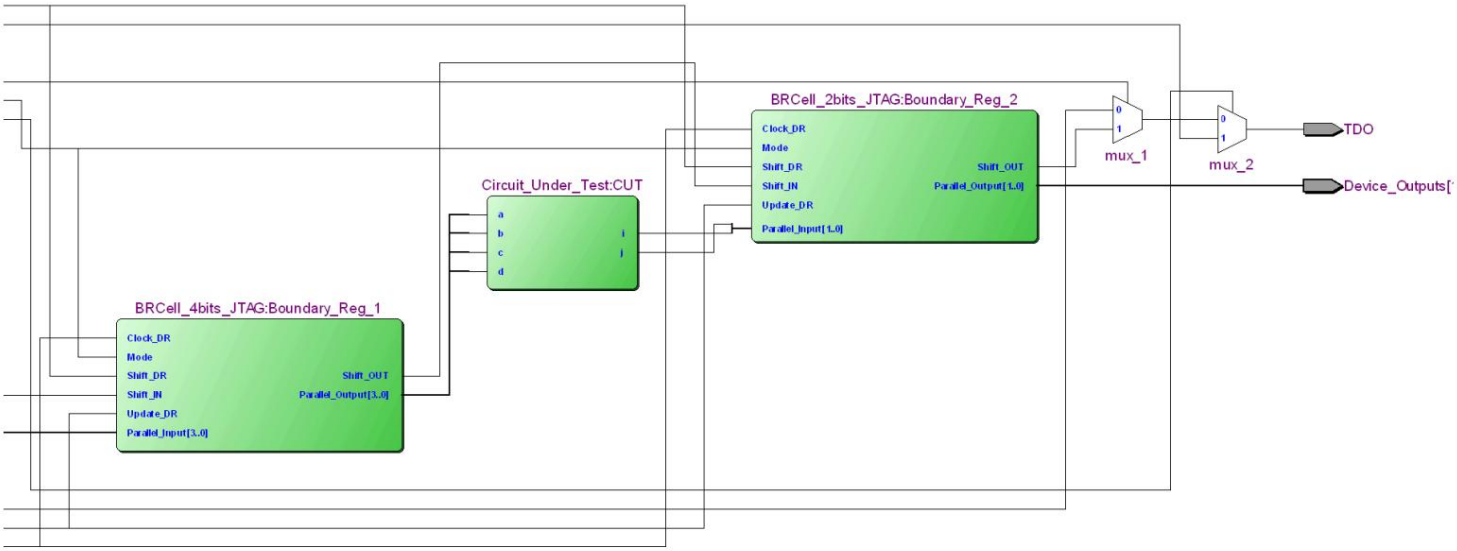
Filename implementing the Bypass Register Decoder (DEC) building block: ***Decoder.vhdl***



Connecting Components - JTAG Implementation



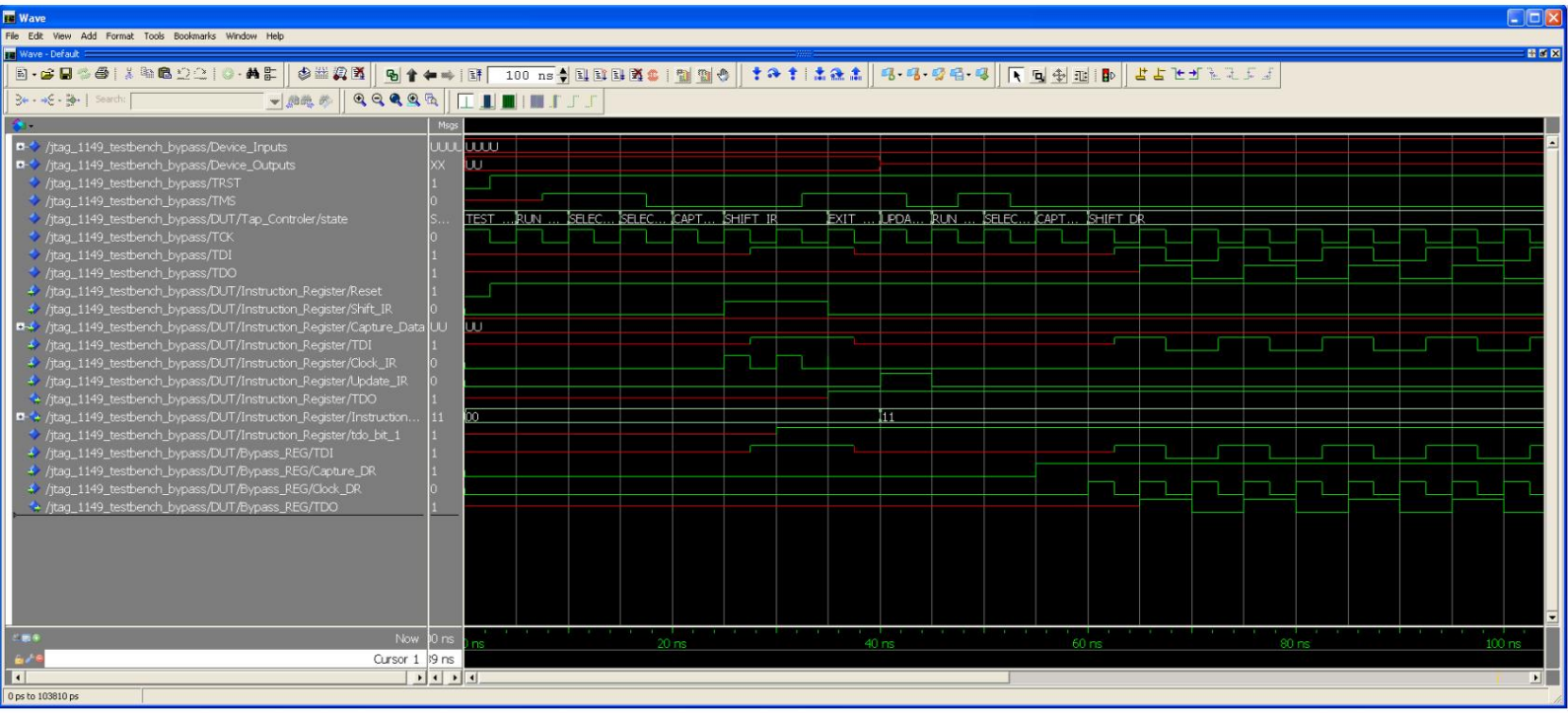


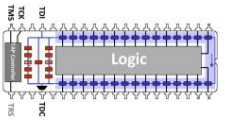


File name in which JTAG 1149.1 is implemented by properly connecting the previously mentioned building blocks together : ***JTAG_1149.vhdl***

Implementation of Mandatory Commands of the standard

BYPASS





TestBench Explained

The simulation starts by setting the TRST signal to logic 0 for 2.5 ns.

Because of this the state machine of the Tap Controller is initialized in the state **TEST_LOGIC_RESET** while all the structural elements that are registered are also initialized to zero because in our implementation we have added the RESET signal to the IRs , BRC.

Due to the above initialization we see the command register containing the value “00” at the beginning of the simulation even though we have not loaded a command into it yet.

Then we load the command for the bypass operation, i.e. the bits "11" through the TDI serial input of the JTAG in an appropriate way.

The command is loaded according to the template, looping through the states in the TAP_CONTROLLER.

The command, as seen in the simulation, has been loaded at time t = 40ns.

With the loading of the command and due to the Decoder, the multiplexers that connect the output of the bypass register with the TDO output of the JTAG, have made the desired output path.

After we have loaded the command, we go through the states of the TAP controller again in order to enter its execution process.

Arriving in the CAPTURE_DR state of the TAP_CONTROLLER, the Capture_DR_Signal signal is set to logical '1', which is necessary so that whatever value the TDI has, passes through the AND gate of the bypass reg.

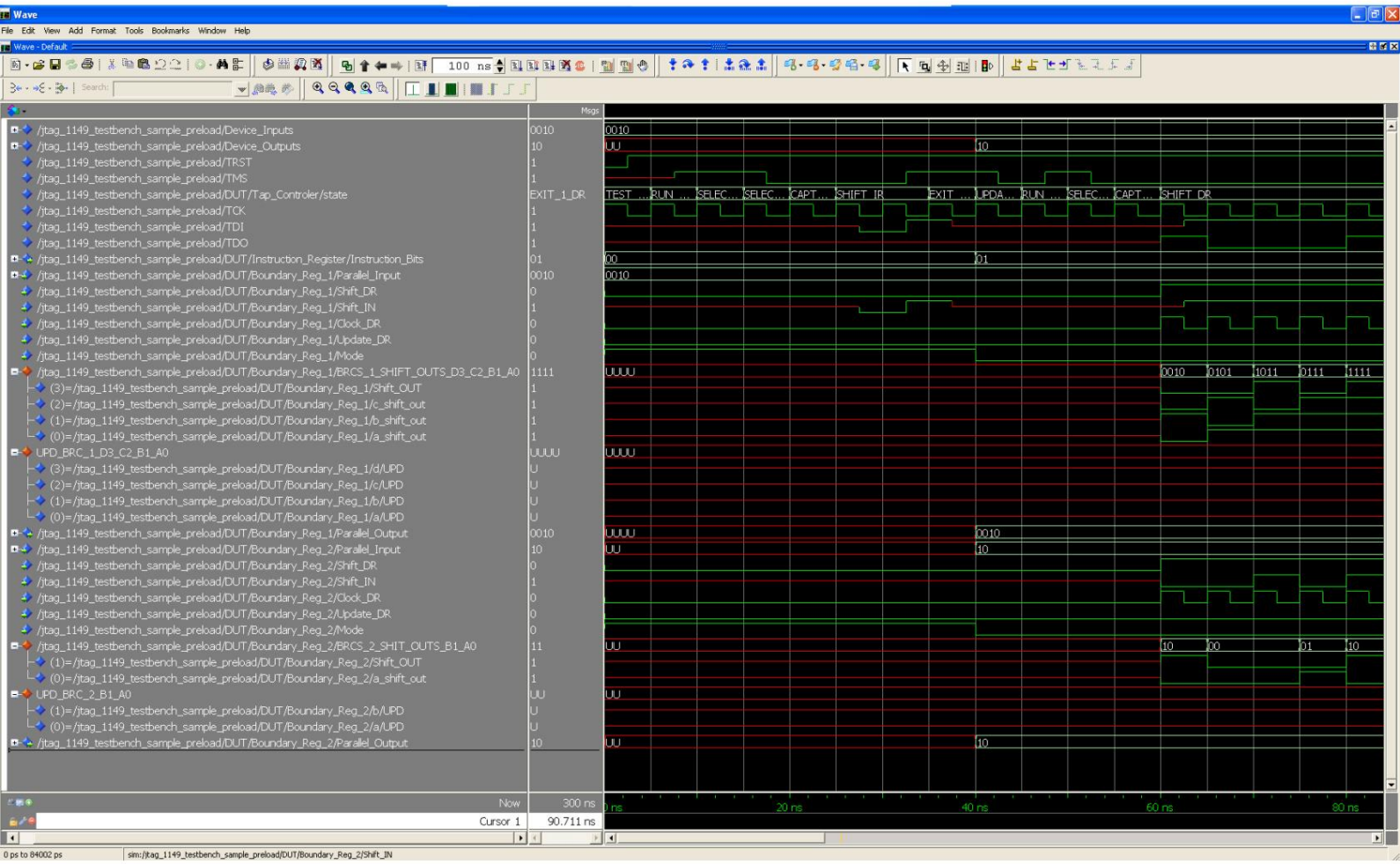
Then going to the SHIFT_DR state and staying there, CLOKC_DR follows the TCK , Capture_DR remains at logic '1' and thus passes into the BYPASS REG the value of TDI and finally we have created the appropriate path from the bypassreg output to the JTAG TDO output.

As long as we are in the shift state on each positive edge of TCK (which equals ME CLOCK_DR in that state) the TDO output will follow the TDI input

The above can also be seen in our simulation from the time point 65 ns onwards.

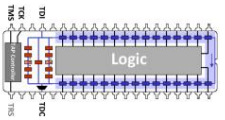
Sample/Preload

Testbench Explanation



- We load the code "10" which corresponds to the encoding of the command SAMPLE/PRELOAD into the INSTRUCTION REGISTER following exactly the same steps as for loading the bypass instruction. Because it is exactly the same process and to reduce complexity, the internal signals of the Instruction Register have been omitted and only the signal showing the bits stored in the Instruction Register has been kept displayed, in order to confirm that the instruction was loaded correctly.

- At time t = 40 ns the command has been loaded successfully.



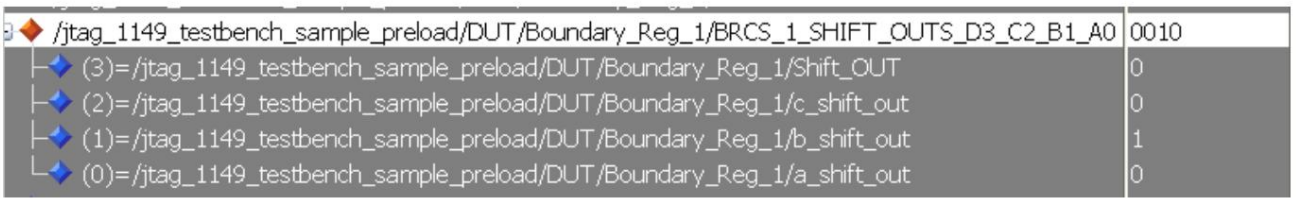
- Then, after the command has been loaded correctly, we run through its states Tap_Controler appropriately in order to reach the Capture_DR state and perform the Sample function.

- It is emphasized that we have taken care at the beginning of the simulation that the parallel loading of the JTAG has been loaded with the vector “0010” which is also visible in the simulation. Parallel Load JTAG = DEVICE_INPUTS

- At the time t = 60 ns we capture in boundary_Reg_1 the values that the CUT accepts as input (from the JTAG device_inputs) and in boundary_Reg_2 we capture the responses of the CUT to these input values.

- The response of the CUT stored in Boundary_Reg_Cell_2 is the expected one given the input vector.

- The correctness of the procedure is confirmed by the signals



and

The following correspond to the shift_outs of each Boundary_Cell contained in Boundary_Reg_1 and Boundary_Reg_2 respectively.

As can be seen in the screenshots, they have been grouped **in reverse** of the order they are in reality, i.e. in reverse of the time the shift_out is done between them.

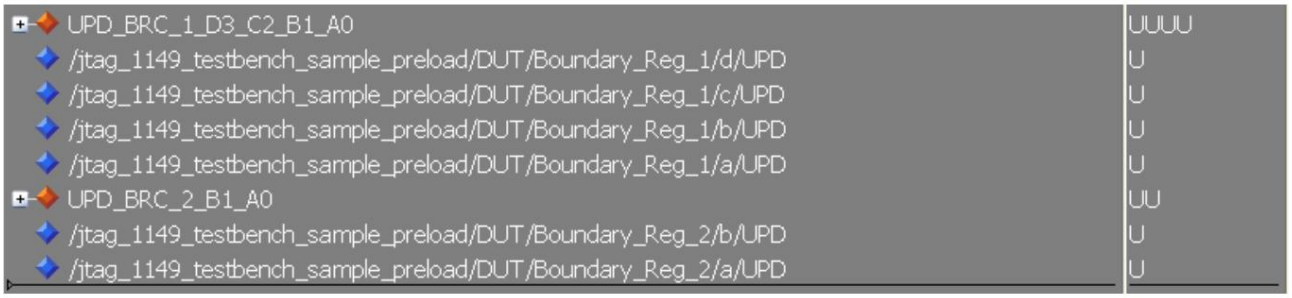
This was done in order to easily visually verify the successful capture with the input value (We keep the most significant bit on the left)

So, be careful, when we shift, then shift from right , in the grouped signal the bits will be seen to left.

Then from time t = 60 ns and for (4+2)*5 = 30 ns we remain in the SHIFT_DR state in order to slide out of the JTAG via TDO serially, the response of the CUT followed by the vector applied to the CUT . At the same time, we make sure that when we output the above serially, the vector "1111" is loaded serially through the TDO with appropriate signals to the TDI.

At the time t= 95 ns, the vector “1111” that we loaded serially into the BRC_1 but also of the “11” that has been serially loaded into BRC_2 and are transferred from the CAP to the UPD register.

The above is evident in the signs



which are grouped in exactly the same logic followed for shift_out